Wesleyan University The Honors College

Languages with Potential: Types & Recurrences for Formal Amortized Analysis

by

Joseph W. Cutler

A thesis submitted to the
faculty of Wesleyan University
in partial fulfillment of the requirements for the
Degree of Bachelor of Arts
with Departmental Honors in Mathematics and Computer Science

Middletown, Connecticut April 2021

Classical watches display time, but can hardly do anything else. This limitation is artificial: for instance, several people confessed to be often in want of mustard... and what is the point of knowing time if you cannot get mustard?

Y.J. Ringard [67]

Acknowledgements

Firstly and most importantly, I must thank my outstanding research advisor, Prof. Dan Licata. Dan took me under his wing when I was only a freshman, and spent countless hours of his valuable time gently teaching me the ways of programming language research. In the years since, he's become an amazing mentor, a tremendously skillful collaborator, and a great friend. For all this, I will be eternally grateful.

I am also incredibly lucky to have been simultaneously mentored by the wonderful Prof. Norman Danner. On top of being the spiritual leader of the Wesleyan PL cost-analysis group and the driving force in shaping my research interests, Norman is an absolutely stellar professor from whom I had the pleasure of taking multiple courses and seminars.

I would like to express my gratitude to my readers, Dan, Norman, and Robert Rose, for taking the time to read this tome.

A huge amount of thanks also goes out to Prof. Deepak Garg, who, across a high-top table at the hotel bar at POPL 2020, offered to host me at MPI-SWS for the internship that became Chapter 2 of this thesis. I also appreciate his generosity in agreeing to work with me virtually when a global pandemic decided to upend my summer-in-Saarbrucken plans.

I am very grateful for the joint friend/mentor-ship offered by Joomy Korkut, Prof. Alex Kavvos, and Mitchell Riley, all of whom seamlessly and rapidly alternate between being outstanding friends, kind mentors, and non-judgmental shoulders to cry on.

I would be remiss not to mention the rotating cast of characters with whom I've shared ESC341 and ESC345, my two "offices" during my time at Wesleyan: Yulia, Pi, Rocco, Elliott, Vabuk, and Ed, you all made it worth trudging through the CT cold to come to work in Exley each day. Similarly, I would like to thank the many denizens of the 6th floor math lounge for providing yet another comfortable and fun working environment.

Of course, as is true of all such projects, this thesis would not have been possible without the emotional support of my wonderful housemates and friends. Sam, Shea, Rachel, the gang from $(\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N})$, and the folks from the PL-twitterverse have all been instrumental in keeping me sane during the madness that has been 2020 and 2021.

Finally, I must acknowledge the tremendous support my family has provided me through college, and during the past year especially. Mom, Dad, and Nathaniel: I got to spent more time with you than I was expecting to this year, and I'm thankful for it every day. Cristian and Vyana: your astounding hospitality made it possible for me to weather the storm of our rocky COVID summer, and emerge on the other end with a head start on the contents of this thesis.

Abstract

While decades of research into formal verification have brought provable correctness guarantees closer to being a part of developers' everyday reality, provable guarantees about algorithmic complexity are harder to come by. This is not for lack of effort. Algorithmic complexity and program cost don't play nicely with abstraction, and so they can prove a difficult target for the kinds of compositional analysis that formal tools handle best. However, the classical algorithm analysis technique known as amortized analysis [75] is promising in this regard: it allows one to selectively break abstraction barriers to precisely and compositionally calculate program costs. In this thesis, we leverage amortized analysis to make two contributions which make some progress towards our goal of provable cost guarantees for the masses.

We begin by taking a cue from the interactive theorem proving school of software verification. We develop a functional language called `LambdaAmor`, which provides a rich refinement type system for in-language amortized cost analysis. We begin with a previously-developed core calculus called λ-Amor [64], transform it to an *algorithmic* type system which is amenable to implementation, and subsequently implement the language in OCaml.

Next, we move to considering a more lightweight method of analyzing the cost of programs, namely the extract-and-solve method of recurrences. This technique is already used (explicitly or otherwise) by practitioners of functional languages, and regularly included in introductory CS curricula. However, the technique is informal, error-prone, and not immediately applicable to amortized cost analysis. Following on work by Danner et al. [19], we formalize the process of amortized analysis by recurrence extraction as a language-to-language translation, and use this to prove the technique's correctness.

Contents

CHAPTER 1

Introduction

As the importance of ubiquity of modern software increases, so too does its complexity. The burden of this complexity blowup lands squarely on the shoulders of software developers, who are asked to create increasingly intricate systems, with little extra help. In response to this, many developers have turned to languages and tooling to help ease the burden: this is exemplified by the rise of Rust, TypeScript, and other and modern strongly-typed languages [78] [77] [52], along with the explosion of interest in formally-verified software by way of theorem proving or other formal methods techniques [68]. While these practices go a long way to improve developer experience and confidence, they are limited in the domains of understanding that they improve. Notably, there are very few existing tools which help developers reason about the resource usage, algorithmic complexity, or performance of their software. The days when resource usage could be easily discerned from source code by eye are long gone, and yet very few techniques have stepped in to fill the void. Moreover, the techniques that software developers use to analyze algorithmic complexity in the absence of formal procedures, languages, or tooling are informal, fragile, and ad-hoc.

The long-term goal of the field of language-based resource analysis is to create languages, tools, and methods which fill this gap by providing programmers with the capability to statically reason about the resource cost of the programs they write. In this thesis, we will restrict our focus along two axes. First, we will only consider a particular resource: run-time cost. Second, we will only consider typed functional programs: the compositional nature of functional programs and the structure provided by types lend themselves greatly to the approaches to cost analysis we consider. Additionally, the techniques will all have the flavor of *amortized* analysis [75], a technique which we discuss in depth in Section 1 below.

We begin in Chapter 2 by exploring a project which serves as a tool for engineers to prove cost properties of their programs. In particular, we consider `LambdaAmor`, a functional language for amortized cost analysis with an immensely expressive type system. The original creators of the core calculus on which `LambdaAmor` is based [64] considered it primarily as a unifying calculus for multiple resource-aware type systems. In this chapter, we consider it instead from a language designer's perspective and investigate what it takes to *implement* the language. `LambdaAmor` fills a point in the design space of resource aware languages analogous to that of dependently-typed languages like Agda [56], Idris [8], and F* [74], or refinement-typed languages like Liquid Haskell

[80] in the space of verification techniques. `LambdaAmor`'s highly expressive type system allows for intrinsic verification [43] of program costs, similarly to how dependent or refinement types allow for intrinsic verification of functional correctness. However, the same expressiveness that allows interesting amortized bounds to be verified in `LambdaAmor` makes this implementation task no easy feat, and so we draw on a plethora of type system algorithmization techniques to eventually arrive at a calculus which is amenable to implementation. Finally, we present and evaluate an implementation of `LambdaAmor` in OCaml.

Then, in Chapter 3, we move to considering a technique that functional programming practitioners *already* use in analyzing the cost of their programs, namely the method of extracting and solving recurrences. As taught in computer science courses, the technique is easily understandable but fraught with potential for error and is entirely informal: there is no proven connection. Previous work by Danner et al. [18] [19] has put recurrence extraction on on firmer ground by formalizing the procedure and proving it correct: we refer to this as *formal recurrence extraction*. With our formalization, there is no doubt that the extracted recurrences are meaningful, and give sound upper bounds on program cost. Formal recurrence extraction has been considered for a large and growing class of functional languages [40] [16]. In this chapter, we extend it to handle recurrences for amortized cost.

0.0.1. *Attributions and Funding.* Chapter 2 is based on work started during the author's Summer 2020 internship at The Max Planck Institute for Software Systems, and completed at Wesleyan University. The project was undertaken under the supervision of Prof. Deepak Garg. This work is unpublished.

Chapter 3 was published and presented at the ACM SIGPLAN International Conference of Functional Programming, 2020. The work was conducted at Wesleyan University under the supervision of Profs. Daniel R. Licata and Norman Danner. This material is based upon work supported by the National Science Foundation under Grant Number CCF-1618203, the Air Force Office of Scientific Research under award number FA9550-16-1-0292, and the United States Air Force Research Laboratory under agreement number FA9550-15-1-0053.

$$\begin{array}{lcl@{\qquad}lcl}
\texttt{inc} & : & \texttt{bit list} \to \texttt{bit list} & \texttt{set} & : & \texttt{nat} \to \texttt{bit list} \\
\texttt{inc}\,[\,] & = & [1] & \texttt{set}\,0 & = & [\,] \\
\texttt{inc}\,(0 :: bs) & = & 1 :: bs & \texttt{set}\,(S\,n) & = & \texttt{inc}(\texttt{set}\,n) \\
\texttt{inc}\,(1 :: bs) & = & 0 :: \texttt{inc}\,bs & & &
\end{array}$$

Figure 1. Binary Counter Data Structure

1. Amortized Analysis Primer

The intuition behind both developments in this thesis is rooted in amortized analysis, the classical algorithm analysis technique first presented by Tarjan [75]. As such, we will provide a brief introduction to the technique here, in addition to presenting a few examples of its utility, one of which we will use as a running example.

Amortized analysis was initially conceived of as a technique for analyzing the worst-case cost of a sequence of operations on a data structure. Without amortization, such cost analyses can be very imprecise. Naïvely, the worst-case cost of a sequence of operations is bounded by the sum of the worst-case cost for each operation. However, this usually fails to take into account internal data structure invariants which make it impossible (or not always true) that each operation in the sequence executes with its worst-case complexity. In short, amortized analysis allows us to peel back some of the abstraction barrier of a datatype in order to more closely analyze the cost of its operations in context. For a concrete example of this phenomenon, consider the (contrived, yet pedagogically useful) example of a binary counter shown in Figure 1.

The type `bit` is either 0 or 1, and `bit list` represents a binary counter with the least significant bit at the head. The `inc` operation increments a counter by one, and the `set` operation applies to n iterates `inc` n times, starting from the zero (empty) counter. To illustrate where a standard analysis overapproximates the cost, we will naïvely analyze the cost of `set`. For simplicity, the only costly operations are cons (`::`) operations, which cost one unit of time each.

Given a counter of length k, `inc` performs at most $k + 1$ cons operations — at worst, the counter is all ones, and `inc` must walk down the entire list flipping ones to zeroes, finishing by cons-ing a one onto the end. It's easy to see that after i calls to `inc`, the counter has length bounded by $\lceil \log_2 i \rceil$, the number of needed for the binary representation of i. Thus, the total

$$\begin{tikzcd} \ldots \arrow[r] & s_{i-1} \arrow[r, "f_i"] & s_i \arrow[r, "f_{i+1}"] & s_{i+1} \arrow[r] & \ldots \end{tikzcd}$$

FIGURE 2. The Generic Amortized Analysis Setup

cost of `set n` is

$$\begin{aligned}\sum_{i=1}^{n} \lceil \log_2 i \rceil + 1 &\leq \sum_{i=1}^{n} \log_2 i + 2 \\ &\leq 2n + \sum_{i=1}^{n} \log_2 i \\ &\leq 2n + \log_2(n!) \\ &\leq 2n + n \log_2 n\end{aligned}$$

While it may be useful for some applications, this bound is not tight. To see why, consider the case where the counter is set to the value of 15_{10}, or 1111_2. A call to increment on this counter costs the full 5 cons operations, leaving the counter at 10000_2. However, a subsequent call to `inc` only costs 1 to flip the first bit. Indeed, very few calls to `inc` traverse the whole list — the vast majority only flip one or two bits. This example illustrates the tension of doing these naive analysis, and provides the primary insight for amortized analysis: while one data structure operation may be expensive, it may also restructure the data structure in such a way which makes *subsequent* operations cheaper than the worst case[1].

1.1. Physicist's Method. The most common method for operationalizing this insight is by the physicist's method of amortized analysis. This method proceeds by associating a data structure with a real-valued "potential function" $\Phi : S \to \mathbb{R}$ on its states. The only restriction on potential functions is that they be nonnegative everywhere. Then, for any sequence of data structure operations f_i with costs c_i and intermediate states s_i (pictured in Figure 2, with s_0 initial and $s_i = f_i(s_{i-1})$), we may define the *amortized cost* of each operation as:

$$a_i = c_i + \Phi(s_i) - \Phi(s_{i-1})$$

[1] This is the genesis of the name *amortized* analysis: expensive function operations effectively pay for subsequent ones to be cheaper, evoking *amortization* from accounting.

The amortized cost of an operation is the actual cost, plus the change in potential across it. When we sum the amortized cost of all the operations across the sequence, the sum telescopes:

$$\begin{aligned}\sum_{i=1}^{n} a_i &= \sum_{i=1}^{n} c_i + \Phi(s_i) - \Phi(s_{i-1}) \\ &= \Phi(s_n) - \Phi(s_0) + \sum_{i=1}^{n} c_i\end{aligned}$$

Then, if $\Phi(s_0) = 0$, we get that the total amortized cost is an upper bound on the total actual cost.

$$\sum_{i=1}^{n} a_i \geq \sum_{i=1}^{n} c_i$$

A good intuition for potential functions Φ is that $\Phi(s)$ represents the amount of work that subsequent operations have to do in order to modify the data structure from state s. In practice, one usually picks potential functions such that when an expensive operation f_i runs, $\Phi(s_{i-1})$ is very large, and $\Phi(s_i)$ is very small so that subsequent operations are cheap.

Returning to the binary counter example, the traditional choice of potential function is to take $\Phi(\texttt{xs})$ to be the number of 1-bits in `xs`. For simplicity, we will denote the actual cost of a call to `inc xs` by $C(\texttt{xs})$, and its amortized cost by $A(\texttt{xs}) = C(\texttt{xs}) + \Phi(\texttt{inc xs}) - \Phi(\texttt{xs})$.

Theorem 1.1. *For all* `xs:counter`*,* $A(\texttt{xs}) = 2$

Proof. By a straightforward structural induction on `xs`.

- (`xs = []`): `inc []` does 1 cons operation, $\Phi(\texttt{[]}) = 0$ and $\Phi(\texttt{[1]}) = 1$, so the amortized cost is 2.
- (`xs = y::ys`): If `y = 0`, then the `inc xs` does 1 cons operation. Since $\Phi(\texttt{1::ys}) - \Phi(\texttt{y::ys}) = (1 + \Phi(\texttt{ys})) - \Phi(\texttt{ys}) = 1$, we again have that the amortized cost of `inc xs` is 2. Now, suppose `y = 1`. Then `inc xs` incurs 1 cost from the cons operation, plus the cost of `inc ys`. So, we may compute:

$$\begin{aligned}A(\texttt{y::ys}) &= C(\texttt{y::ys}) + \Phi(\texttt{inc (y::ys)}) - \Phi(\texttt{y::ys}) \\ &= 1 + C(\texttt{ys}) + \Phi(\texttt{inc ys}) - (1 + \Phi(\texttt{ys})) \\ &= C(\texttt{ys}) + \Phi(\texttt{inc ys}) - \Phi(\texttt{ys}) \\ &= A(\texttt{ys})\end{aligned}$$

But by the inductive hypothesis, $A(\texttt{ys}) = 2$, which completes the proof.

□

An immediate corollary of this fact is that the amortized cost of `set n` is bounded by $2n$, by the same telescoping argument as before. Most importantly, since the binary counter `[0]` has potential 0, the amortized cost of `set n` is an upper bound on the *actual* cost of `set n`. This last step is crucial. A priori, amortized costs give no information about the actual costs of program execution, and are only a bound on the actual cost if the final potential is greater than the initial.

1.1.1. *Single Function and Gas Tank Analyses.* Often, amortized analysis is applied to individual functions, rather than a sequence. This may be thought of as the length-one case of amortized analysis. Given a function `f:a->b` and potential functions $\Phi_{\mathtt{a}} : \mathtt{a} \to \mathbb{R}$ and $\Phi_{\mathtt{b}} : \mathtt{b} \to \mathbb{R}$, we may define the amortized cost of `f` as $A_{\mathtt{f}}(\mathtt{x}) = C_{\mathtt{f}}(\mathtt{x}) + \Phi_{\mathtt{b}}(\mathtt{f\ x}) - \Phi_{\mathtt{a}}(\mathtt{x})$, where $C_{\mathtt{f}}(\mathtt{x})$ is the cost of `f`.

A common variation on this concept is to pick the potential functions such that the amortized cost is *zero*. Then, the actual cost $C_{\mathtt{f}}(\mathtt{x})$ is exactly $\Phi_{\mathtt{a}}(\mathtt{x}) - \Phi_{\mathtt{b}}(\mathtt{f\ x})$. The usual intuition here is to think of the available potential as a sort of "gas tank" which the function must siphon from to do work. Then, the total gas used, $\Phi_{\mathtt{a}}(\mathtt{x}) - \Phi_{\mathtt{b}}(\mathtt{f\ x})$, is an upper bound on the amount of work done by the function.

1.2. Banker's Method. While the physicist's method is powerful, some situations call for a more fine-grained analysis. In this case, we employ the so-called banker's method of amortized analysis. The banker's method works by introducing imaginary "credits" to a data structure, which may be "attached" to the values in a program. These credits are thought of to interact with the cost model of the language in a special way: credits can always be created from thin air at a cost of one unit of time, and then they can subsequently be discarded or "spent" to decrease execution cost by one unit. In the setting of the banker's method, this is what we mean when we refer to "amortized cost" — the real cost of an operation, plus the cost of creating and spending credits along the way. Crucially, if an operation begins with no credits available, then the amortized cost must be an upper bound on the actual cost since credits must be created (incurring a cost of 1) before they can be spent (decreasing the cost by 1). All of this is best illustrated by returning to the binary counter example.

We begin by enforcing a credit invariant on values of type `counter`: every `1` bit must have a credit attached. It's worth confirming that the increment function is able to maintain this invariant: if the counter is empty, we spawn a credit, attach it to a `1` bit, and cons it to the front of the list. If the least significant bit is `0`, we again spawn a credit, flip the bit to `1`, and attach the credit. Finally, if the least significant bit is `1`, then we detatch its credit, spend it, recurse down the tail, and finish by cons-ing a `0` onto the front. Of course, none of this is manifest in the code [2]. Just like with potential in the physicist's method, these proofs must happen off to the side on paper.

Finally, we may perform the analysis itself.

Theorem 1.2. *For all `xs:counter` satisfying the credit invariant, the amortized cost of `inc xs` is 2.*

Proof. We proceed by structural induction on `xs`.

- (`xs = []`): `inc []` does 1 cons operation and spawns one credit, for an amortized cost of 2.
- (`xs = y::ys`): If `y = 0`, then `inc xs` does one cons operation and spawns a single credit, for an amortized cost of 2. Finally, if `y = 1`, then `inc xs` makes a single recursive call `inc ys`, which has amortized cost 2, by inductive hypothesis. But then, the function spends the credit attached to `y`, which cancels out the cost 1 incurred by cons-ing a `0` onto the result of the recursive call. In total, this case has 2 amortized cost, as required.

□

1.3. Comparisons. The reader may note that the proof of Theorem 1.2 was remarkably similar to the proof of Theorem 1.1. This is, unsurprisingly, not by coincidence. The reason for this similarity is that the banker's method can be thought of as a concretization of the physicist's method. Rather than "globally" assigning potential to the states of a data structure, the banker's method "localizes" the potential, thought of as discrete credits, on specific values

[2] Readers familiar with concurrent separation logic might find this idea familiar: credits are a form of ghost state.

in the state. Because of this, analyses with the banker's method have an "operational" feel to them, while analyses using the physicist's method have a more "calculational" flavor.

On the whole, the two methods of amortized analysis are essentially equivalent in power. Given a banker's method analysis, we may turn it into a physicist's method analysis by taking the potential function to be the total number of credits. Conversely, physicist's method analyses can be converted to use the banker's method by maintaining the invariant that $\lceil \Phi(\mathtt{s}) \rceil$ credits be kept on the value $\mathtt{s}$.

Proofs using the banker's method are often more tedious and traditionally less formal. In Chapter 3, we present a formalization of the banker's method by way of recurrence extraction. Chapter 2 presents λ-Amor, which is based primarily on the physicist's method.

2. Substructural and Modal Types Primer

While Chapter 3 is primarily concerned with approaching cost analysis from a recurrence extraction angle, it will, like Chapter 2, involve the creation of a type system. The two type systems under consideration, those of λ^A and λ-Amor, share a major feature which is common to most developments in resource analysis: they are both *affine substructural* type systems. Most type systems have three "non-logical" rules governing the behavior of their typing context. The rule that allows contexts to be freely considered up to permutations is called *exchange*, the rule allowing variables to be dropped from the context is called *weakening*, and the rule which allows variables to be duplicated is referred to as *contraction*. The three rules are sometimes explicitly included in the list of rules generating a typing judgment, but more often than not they are omitted and derived as admissible rules.

Substructural type systems are ones in which some or all of the traditional structural rules governing typing contexts are disallowed. Disallowing all of them yields ordered types, and allowing only exchange yields linear types. Most importantly for this thesis however, is the combination of exchange and weakening (but not contraction), which yields an *affine* type system. In an affine type system, contexts are sets, where each variable from the context may be used at most once.

The rules of an affine type system are set up in such a way that the contraction rule is inadmissible. In particular, every multi-premise rule "splits" the context so that a variable cannot be used by more than one sub-term. For instance, consider the usual product introduction rule of a fully-structural type system:

$$\frac{\Gamma \vdash e_1 : A \qquad \Gamma \vdash e_2 : B}{\Gamma \vdash (e_1, e_2) : A \times B}$$

is transformed into the following rule to support affine types:

$$\frac{\Gamma_1 \vdash e_1 : A \qquad \Gamma_2 \vdash e_2 : B}{\Gamma_1, \Gamma_2 \vdash (e_1, e_2) : A \otimes B}$$

The context must be divided into two disjoint parts Γ_1 and Γ_2 to be given to the two premises[3].

Affine types are a natural object of study in the resource-usage literature, since they naturally enforce a resource-usage restriction: using a variable "consumes" it, which corresponds to the consumption of the resources the variable holds. In the type systems we will study for the remainder of this thesis, the notion of resource which our affine type systems will track will be the credits and potentials of amortized analysis. Indeed, non-duplication of variables in an affine type system can be leveraged to enforce the non-duplication requirement of credits and potential in the banker's and physicist's method of amortized analysis.

Of course, this restriction on the usage of variables is very strong from a programming perspective. Even incredibly simple programs often require the re-use of a single variable. The traditional solution to this is the introduction of a new type $!A$, which represents values of type A that can be used arbitrarily many times. This $!$, usually referred to as the *exponential modality* is our first example of a modality, or a unary operator on types. To "implement" this modality in the type system, one usually adds a second context to the typing judgment of so-called exponential variables, which can be used multiple times. This context is fully structural — it is not split in multi-premise rules. For instance, the product introduction rule becomes something like:

$$\frac{\Omega; \Gamma_1 \vdash e_1 : A \qquad \Omega; \Gamma_2 \vdash e_2 : B}{\Omega; \Gamma_1, \Gamma_2 \vdash (e_1, e_2) : A \otimes B}$$

[3] The name of the connective changes also — in a fully structural type system, positive and negative products are isomorphic, and thus written $\times$. Passing to a substructural type system disentangles the two notions.

and we also add an exponential variable rule to use variables from the Ω context.

$$\frac{}{\Omega, x : A; \Gamma \vdash x : A}$$

The modality itself is governed by a pair of simple introduction and elimination rules. A term can be made exponential with the introduction rule, so long as it does not depend on any affine resources:

$$\frac{\Omega; \cdot \vdash e : A}{\Omega; \Gamma \vdash e : !A}$$

The elimination rule allows for a term of type $!A$ to be bound as one of type A in the exponential context, and thus be used many times:

$$\frac{\Omega; \Gamma_1 \vdash e : !A \qquad \Omega, x :: A; \Gamma_2 \vdash e' : B}{\Omega; \Gamma_1, \Gamma_2 \vdash \texttt{let}\,!x\ \texttt{=}\ e\,\texttt{in}\,e' : B}$$

Both λ^A and λ-Amor make use of versions of this modality, but λ^A generalizes it to also allow n-affine types, whose values may be used at most n times. In addition, each type system uses a *graded* modality to quantify the usage of potential and credits. While the presentation (and terminology) is slightly different, the $!_r A$ modality of λ^A and the $[I]\,A$ modality of λ-Amor can be thought of as essentially one and the same.

CHAPTER 2

Amortized Analysis with Type Systems

1. Introduction

As anyone who's ever tried it knows, writing correct software is hard. Fortunately, decades of work in verification and interactive theorem proving for program correctness have brought forth a world of possibilities for future programmers to harness in their quest to build robust, correct, and extensible modern software [68]. The vast majority of work in this area is about *functional* or *extensional* correctness: proving that a program's input/output behavior matches the programmer's intended specification. Much less well studied is the correctness of programs with respect to *intensional* properties: those which refer to *how* a program runs, rather than simply what it computes. Of particular interest to this thesis is the intensional property of resource usage. While some intensional properties such as information flow can be rephrased in an extensional manner[1], resource usage is inherently intensional. More specifically, as Chapter 1 suggests, we will restrict our view to a particular resource: that of *cost*.

One particularly promising verification method based in interactive theorem proving is *intrinsic* verification [43], wherein the program being verified and the proof of its correctness are packaged together. In this approach, expressive (usually dependent) type systems are used to encode invariants in the types of the program, in such a way that a certificate that the program is type-correct is also a certificate of its functional correctness.

In this work, we apply the approach of intrinsic verification to proving cost bounds of functional programs. Unfortunately, the settings in which intrinsic verification of extensional properties is traditionally performed — dependently-typed proof assistants such as Coq [76], Agda [56], or F* [74], to name a few — are not optimal settings for verifying cost bounds of programs. While some work [50, 14, 27] has attempted to forge ahead and perform intrinsic cost analyses in dependently-typed languages, it is all limited in various ways by the fact that cost and potential is not a first-class notion in any of their logics[2].

Instead, the primary goal of this work is the development of `LambdaAmor`, a domain-specific functional language for cost verification which combines a first class notion of cost with a rich

[1]For instance, noninterference [26] is an extensional (hyper-)property which soundly under-approximates information flow control policies

[2] Some dependent type theories such as Cost-Aware Type Theory (CATT) [54], *do* have first-class notions of cost, but there are no proof assistants built on top of them.

refinement type system to statically verify time bounds of the programs written in it. By analogy to the behavioral invariants in the types of of traditional intrinsic analyses, programs written in λ-Amor have types which enforce cost invariants. For instance, a function in `LambdaAmor` could have a type like "function from `nat` to `nat` which runs in no more than five steps". To expand the class of possible cost analyses, `LambdaAmor` supports (as the name implies) amortized analysis. This is enabled by adding types which classify values carrying certain amounts of potential. These cost types and potential types can be combined in nontrivial ways to express nontrivial amortized analyses, which is often required when deriving tight bounds involves breaking data structure abstraction boundaries. Crucially, these cost and potential invariants are statically enforced by the type system: a certificate that a program in `LambdaAmor` is type-correct is also a certificate that it is cost-correct. In this sense, programs written in `LambdaAmor` *are* cost analyses of themselves. This justifies an occasional reference to `LambdaAmor` (or its core calculus) as allowing programmers to "perform a cost analysis" of a program, ostensibly written elsewhere — this amounts to simply re-writing the program in `LambdaAmor`, with types that mirror the corresponding "on-paper" analysis that would have been performed in `LambdaAmor`'s absence.

While other functional languages with amortized cost analysis capabilities (and resource analysis more generally) do exist, `LambdaAmor` sits at a minimally-explored point in the design space. Some languages like Resource Aware ML [31] aim for full automation, at the cost of struggling to handle some common language features. Other languages like TiML [81] or LRT [42] attempt to strike a balance by giving up on a degree of automation and requiring annotations from the programmer in order to make gains in expressiveness. In contrast to these languages[3], `LambdaAmor` emphasizes expressiveness above all else. While this comes at the cost of some automation, `LambdaAmor` is still backed by SMT, and so all of the *quantative* proof obligations are handled automatically.

The core of `LambdaAmor` derives from a type system called λ-Amor [64]. By and large, the creators of λ-Amor were interested in it as a unifying foundational framework in which one could embed *other* cost analysis languages: there are many axes along which one may design a type system for resource analysis, and λ-Amor serves as a calculus in which all sorts can be be emulated. In this work, however, we primarily interest ourselves in λ-Amor's usefulness as

[3] A further comparison between `LambdaAmor` and these languages along with others can be found in Section 10

a core calculus of a programming language which allows its users to prove cost bounds on the programs they write in it. λ-Amor on paper is expressive enough to assign amortized cost bounds for a wide class of functional programs, from the traditional examples of amortized analysis such as functional queues and binary counters, to fully general cost-polymorphic higher-order functions like `map` and `fold`, which aren't well handled by existing resource-analysis languages like Resource Aware ML [31]. This power and flexibility makes λ-Amor a perfect starting point to develop the core calculus of `LambdaAmor`.

The main contribution of this work is the design and theory of a version of λ-Amor called dλ-Amor which is amenable to implementation. This is accomplished by cutting out a fragment of λ-Amor, and restricting its syntax somewhat. In the end, our changes will have been minor. dλ-Amor bears all of the same major features as λ-Amor: a pair of modalities for cost and potential, an affine substructural type system for soundly tracking the potential (values with potential must not be duplicated), and refinement types for encoding potential functions which vary in the sizes of data structures. The major change is the introduction of a construct to selectively restrict the forms of potential functions, borrowed from Automated Amortized Resource Analysis (AARA) [29]. However, despite this work in cutting out an implementable fragment, dλ-Amor does not admit a direct implementation, as its typing rules (just like λ-Amor's) are written in declarative style, from which we cannot immediately construct an algorithm for type-checking or type inference.

The traditional solution to this is to create yet another type system — an *algorithmic* one, from which a type-checker can be easily implemented. For this purpose, we will introduce biλ-Amor, a type system which encodes the same typing relations as dλ-Amor, but is presented in a manner that is trivial to implement. biλ-Amor leverages several techniques from the type systems and type theory literature to algorithmize the dλ-Amor type system. Most notably, biλ-Amor makes use of bidirectional type inference [62], a technique which allows for the implementation of highly expressive type systems, while minimizing the amount of annotation required of the programmer. Additionally, biλ-Amor harnesses normalization, syntax-directedness through admissible rules, and constraint generation to pave the way for an implementation.

Designing biλ-Amor is a nontrivial task, and proving it correct even moreso. To show that dλ-Amor and biλ-Amor are the same type system presented in different ways (the latter being

$$\begin{tikzcd}\boxed{\texttt{LambdaAmor}} \arrow[r, "\text{Implements}"] & \boxed{\text{bi}\lambda\text{-Amor}} \arrow[r, "\text{Algorithmizes}"] & \boxed{\text{d}\lambda\text{-Amor}} \arrow[r, "\text{Subset of}"] & \boxed{\lambda\text{-Amor}}\end{tikzcd}$$

FIGURE 1. Relationship Between Calculi

easily implementable), we must prove a bevy of theorems relating the two. This proof effort makes up the bulk of the technical contribution of this chapter. Finally, once the type system design and proof work is complete, we implement biλ-Amor. Thanks to all of the work done in algorithmization, our implementation is at its core a trivial translation of the rules of biλ-Amor into code.

The outline of the rest of the chapter is as follows.

- We will begin in Section 2 by giving an overview of the concepts λ-Amor draws on. As mentioned previously, λ-Amor includes two modalities for tracking cost and potential. To soundly manage this potential, λ-Amor is based on an affine logic in which every variable may be used at most once so that values with potential cannot be duplicated. To make complex potential functions, λ-Amor uses refinement types in the style of Dependent ML (DML) [84], which we review. Next, we discuss the main obstacle the original type system presents to implementation: constraint solving. Our solution to this problem is based on univariate polynomial potential functions in the style of AARA, which we introduce. This motivates the primary restriction of dλ-Amor compared to λ-Amor: costs and potentials are (with some exception) AARA-style univariate polynomials. Finally, we provide a more foundational account of dλ-Amor's cost and potential modalities, based in linear logic.
- Next, in Section 3, we explore some programs written "on-paper" in the core calculus dλ-Amor. These examples serve to illustrate the kind of cost-correctness proofs enabled by dλ-Amor, and provide a first glimpse of how programs in `LambdaAmor` will be packaged together with their cost-correctness proofs. We present a wide variety of programs, each of which shows off a different facet of of dλ-Amor's type system. These examples provide the beginnings of a comprehensive test suite against which we can evaluate our eventual implementation.

- Then, in Section 4, we give an overview of biλ-Amor, the algorithmic version of dλ-Amor. To help motivate biλ-Amor's creation, we begin by describing the pitfalls which make it impossible to directly implement dλ-Amor. We then move to presenting a high-level overview of the techniques we use to avoid these implementation obstacles.
- In Section 5, we discuss the syntax and type system of dλ-Amor in depth, providing intuition for the each of the judgments, and discussing selected rules from the type system. dλ-Amor's type system is many-layered, with judgments for type formation, type assignment, and a smaller type system for the sub-language which governs the refinement types. We pay special attention to the rules which govern the cost-analysis-specific language features, and describe them in detail.
- In Section 6, we sketch the soundness proof for dλ-Amor, by showing that it may be embedded in λ-Amor, and appealing to its soundness theorem featured in Rajani et al. [64].
- In Section 7, we introduce the formalism for biλ-Amor. While the majority of the syntax is carried over from dλ-Amor, this formalism differs drastically from that of dλ-Amor, and so we take time to explore the ways that the algorithmization features discussed previously in Section 4 are actually applied.
- In Section 8, we prove that biλ-Amor and dλ-Amor are in fact (essentially) the same type system. This fact is a requirement for a good implementation, as it guarantees that our typechecker accurately and soundly types terms. The proof is broken into two parts. A proof of soundness tells us that when a typechecker derived from the algorithmic rules of biλ-Amor confirms that an expression has a given type, our "ground truth" declarative dλ-Amor agrees. Dually, the proof of completeness ensures that every declaratively-derivable typing relationship in dλ-Amor will be found by a typechecker which implements biλ-Amor's algorithm.
- Finally, in Section 9, we discuss `LambdaAmor`, our OCaml implementation of the dλ-Amor. In order to support nontrivial programs, `LambdaAmor` sports a top-level environment with multiple declaration types, on top of the simple typechecking prescribed by biλ-Amor. We discuss these additions to the language, as well as the specific design

choices made while building the artifact. We finish the section by writing the examples from Section 3 in `LambdaAmor`, and benchmarking our implementation.

2. Overview of dλ-Amor

In this section, we will begin by presenting the overarching ideas which make λ-Amor useful as a core calculus for our resource-aware language. Subsequently, we move to discussing its variant, dλ-Amor, which we will focus on for the rest of the chapter.

One of the most basic insights that λ-Amor takes advantage of in its design is that costly computation can be thought of an effect[4]. When a program does work, it has an effect on the world, namely the effect of taking time. In this sense, nearly all "pure" programming languages are impure, as they allow pervasive use of the effect of cost. In contrast to most languages, λ-Amor enforces strict requirements on the use of this effect in particular. While many solutions to controlling effects have been explored in the literature [47] [63], λ-Amor takes the approach of enforcing a monadic [53] discipline on the effect of cost.

2.0.1. *Cost Monad.* However, a simple monad is not enough. We care not only that a term may incur cost, but how much cost it can incur! For this purpose, λ-Amor employs a *graded* monad $\mathbb{M}\, I\, \tau$ to encapsulate the effect of cost [24]. A computation of this type returns a value of type τ, and may incur up to I cost, where I is drawn from the sort of positive real numbers. As a graded modality, this monad's operations interact with the grade in nontrivial ways: for instance, the "pure" computation $\texttt{ret}(e)$ has type $\mathbb{M}\, 0\, \tau$ when $e : \tau$. This allows any pure term to be lifted to a monadic computation which incurs no cost. Most importantly, given a costly computation $e_1 : \mathbb{M}\, I_1\, \tau_1$ and a continuation $x : \tau_2 \vdash e_2 : \mathbb{M}\, I_2\, \tau_2$, the two can be sequenced into a computation $\texttt{bind}\, x = e_1\, \texttt{in}\, e_2 : \mathbb{M}\, (I_1 + I_2)\, \tau_2$. Note that the costs add: a computation which may take up to I_1 units of time followed by a computation which takes up to I_2 units takes at most $I_1 + I_2$ units. However, neither `ret` nor `bind` incurs any nontrivial cost: any program written using only `ret`s and `bind`s will have type $\mathbb{M}\, 0\, \tau$. For this, λ-Amor includes a term $\texttt{tick}[I]$ of type $M\, I\, 1$, which incurs cost I (and 1 is the unit type). This is the only construct in λ-Amor which incurs any "extra cost", the idea being that programmers insert `tick`s in front

[4] In fact, cost can also be thought of as a *coeffect* [25], and one of the major breakthroughs of λ-Amor is the unification of both styles of resource tracking in a single calculus.

of the operations their specific cost model dictates are costly. This technique is widely used in the cost analysis literature [14], and so λ-Amor also adopts it for simplicity.

But of course, this cost monad can only be half the story. In a language which seeks to provide types for amortized analysis, a mechanism for handling potential is required.

2.0.2. *Potential Modality and Affine Types.* In addition to the cost monad, λ-Amor includes another graded modality for tracking potential. A term of type $[I]\ \tau$ can be thought of a term of type τ which stores I potential[5], where I is again drawn from a sort of positive real numbers. The most important operation associated with the potential modality is the ability to use potential to offset the cost of a computation. Concretely, given a term $e_1 : [I]\ \tau_1$ and a monadic continuation $x : \tau_1 \vdash e_2 : \mathbb{M}\ (I + J)\ \tau_2$, we can form the computation $\texttt{release}\, x = e_1 \,\texttt{in}\, e_2 : \mathbb{M}\ J\ \tau_2$. The crucial aspect of this construction is the fact that the resulting computation requires at most J units of time to run, while the initial computation e_2 required $I + J$. Intuitively, we think of this as the I units of potential "paying for" I steps of computation.

Potential may also be created and attached to values. In λ-Amor, these two operations are handled by the same construct. For terms $e : \tau$, we may form $\texttt{store}[I](e) : \mathbb{M}\ I\ ([I]\ \tau)$, which is a computation which runs for at most I units of time, and returns a τ with I potential attached. The fact that `store` incurs this cost is what justifies the term `release` — the program has paid an "extra" cost of I to create $[I]\ \tau$, and thus can exercise this option to reduce the cost of a subsequent computation with `release`.

This dynamic between `store` and `release` forces a restriction on the type system: variables can only be used at most once. Our argument for the soundness of `release` relies on an the assumption that the potential we are releasing has not already been released elsewhere, and so duplication of variables must be disallowed. This kind of restriction is very common, as discussed in Section 2: λ-Amor is an affine type system.

2.0.3. *Refinement Types and Index Terms.* The situation we've described so far would only allow types with *constant* amounts of potential. For nontrivial analyses, this is wholly insufficient, as the potential of a data structure must be able to depend on the size or other numerical

[5] In some senses, potential in λ-Amor behaves more like the credits of the banker's method discussed in Chapter 1 — it can be created and attached to specific values. To avoid confusion, we follow Rajani et al. [64] with the terminology of "potential"

parameters of that data structure. For this purpose, λ-Amor includes *refinement types* in the style of Dependent ML [84]. Concretely, λ-Amor supports length-refined lists. A value of type $L^I\tau$ is a list of length I, where I is a term in a small language of arithmetic expressions over a set of variables, which we call an *index term*. These index terms may also appear in potentials. For example, $\left[I^2\right]\ (L^I\tau)$ is the type of lists of length I with potential I^2.

2.0.4. *Index Term Quantifiers, Indexed Types, and Constraint Types.* To make good use of these refinements, λ-Amor supports more refinement-related types. While not strictly part of the resource-analysis "core" of λ-Amor, these are required for practical use. First and foremost is the inclusion of universal and existential quantifiers over index terms, which allow for types like $(\tau \multimap \sigma) \multimap \forall n : \mathbb{N}.\,(L^n\,\tau \multimap L^n\,\sigma)$, a possible type for a map function which can operate on lists of any length. λ-Amor also includes a syntax for constraints over index terms, which take the form $I = J$, $I \leq J$, $I < J$ along with conjunctions, disjunctions, and implications thereof. These constraints are used in *constraint types*. The conjunction constraint type $(n \geq 1)\&\tau$ classifies values of type τ with an attached (irrelevant) proof of $n \geq 1$, while terms of the implication constraint type $(m + n = 1) \implies \sigma$ have type σ when $m + n = 1$ is true. Finally, λ-Amor sports indexed types, which can be thought of as type-level functions from sorts to types: $\lambda i : \mathbb{N}.L^i\tau$ is a function which, given a natural number i, produces the type $L^i\tau$. Note that this is not the same as $\forall i : \mathbb{N}.L^i\tau$, as they have different "kinds": the first has kind $\mathbb{N} \to \star$ (a function which returns types), while the second has kind $\star$, the kind of types of terms.

2.1. Potential Vectors and AARA. The story we've just told about λ-Amor is loyal to the original presentation in [64], but somewhat inadequate for implementation purposes. As we will discuss in Section 4, efficient subtyping is necessary for implementation of λ-Amor. However, the inclusion of the potential and cost modalities presents a challenge. For $[I]\ \tau_1$ to be a subtype of $[J]\ \tau_2$, it must be that $\tau_1 <: \tau_2$, and that $J \leq I$. But as discussed above, I and J are index terms, and may be polynomials in a set of index variables. Ideally, we would like to discharge these inequalities generated by subtyping by constraint solver, but even the most advanced SMT solvers struggle to handle polynomial inequalities.

To solve this problem, we borrow a key idea from AARA [29] which will allow us to generate only linear constraints over index variables, while still allowing univariate polynomial potentials and cost. The main idea is to fix a clever "basis" for the space of polynomials, and then represent polynomials as a vector of their coefficients with respect to that basis. The basis in question is chosen to satisfy one key property: if a real polynomial $f(n)$ is written in terms of the basis, then the coefficients of $f(n-1)$ may be efficiently determined from those of $f(n)$. This property gives rise to the ability to easily analyze list algorithms in λ-Amor: when writing a function $([f(n)]\,(L^n\,\tau)) \multimap \sigma$, it is simple to pattern match on the argument and determine the type of the tail $[f(n-1)]\,(L^{n-1}\,\tau)$ to pass to a recursive call.

In dλ-Amor, we will syntactically restrict potential functions to be of this form, with some exception. It is intuitively clear that all AARA-style potential functions are expressible in the index term language of the original λ-Amor[6]. This restricted potential form is also sufficiently expressive for practical purposes, as the examples we present in Section 3 show.

The reader accustomed to the literature surrounding AARA or Resource-Aware ML is likely to be familiar with the following presentation of AARA-style polynomial potential. The less familiar reader is encouraged to consult Hoffmann's Thesis [28] for a more in-depth exposition of the technique.

Definition 2.1 (Potential Vector). *For a fixed k, we call a vector of nonnegative reals $(a_0, \dots, a_k)$ a potential vector.*

Definition 2.2 (ϕ Function). *For fixed k, we define $\phi : \mathbb{N} \times \mathbb{R}^k_{\geq 0} \to \mathbb{R}_{\geq 0}$ to be*

$$\phi\,(n, (p_0, \dots, p_k)) = \sum_{i=0}^{k} p_i \binom{n}{i}$$

where $\binom{n}{r}$ is the binomial coefficient. We refer to the first argument of ϕ as the "base", and the second argument as the "potential".

With ϕ in hand, we redefine the cost and potential modalities. In dλ-Amor, the cost modality is written as $M\,(I, \vec{p})\,\tau$ and the potential modality is $[I|\vec{p}]\,\tau$. These two types classify values of type τ which cost up to $\phi(I, \vec{p})$ units of time and posess $\phi(I, \vec{p})$ potential, respectively.

Theorem 2.1 (Monotonicity and Additivity of Φ). *Let $\vec{p}$ and $\vec{q}$ be potential vectors.*

(1) If $\vec{p} \leq \vec{q}$ componentwise, then $\phi(n, \vec{p}) \leq \phi(n, \vec{q})$.

(2) $\phi(n, \vec{p} + \vec{q}) = \phi(n, \vec{p}) + \phi(n, \vec{q})$

[6] While we do not prove this fact, a version of the requisite translation can be found in the code of the constraint elaboration pass described in Section 9.

This theorem has two main consequences for the new cost and potential modalities. First, the fact that ϕ is monotone in its second argument allows us to reduce the problematic subtyping rule for potentials (and costs) to generating linear inequalities and equalities[7] $[I|\vec{p}]\,\tau_1$ is a subtype of $[J|\vec{q}]$ when $I = J$ and $\vec{q} \leq \vec{p}$ componentwise. Second, the additivity of ϕ also allows us to pass from addition of polynomial index terms in the `bind` and `release` rules to componentwise (linear) addition on potential vectors. Given a computation $e_1 : \mathbb{M}\,(I, \vec{p})\,\tau_1$ and a continuation $x : \tau_1 \vdash e_2 : \mathbb{M}\,(I, \vec{q})\,\tau_2$, we perform the computations in sequence with $\mathtt{bind}\,x = e_1\,\mathtt{in}\,e_2 : \mathbb{M}\,(I, \vec{p} + \vec{q})\,\tau_2$.

For convenience, it is sometimes useful to consider a restricted version of the orignal λ-Amor cost monad in dλ-Amor: for this we will sometimes write $\mathbb{M}\,\vec{p}\,\tau$ to mean $\forall j : \mathbb{N}.\mathbb{M}\,(j, \vec{p})\,\tau$. Intuitively, this ought to be considered the same as $\mathbb{M}\,(0, \vec{p})\,\tau$, since a computation that costs at most $\phi(j, \vec{p})$ for any j must be bounded above by $\phi(0, \vec{p})$ by monotonicity of ϕ.[8] Rather than writing this type as such, we instead use the universally-quantified type $\forall j : \mathbb{N}.\mathbb{M}\,(j, \vec{p})\,\tau$ to ensure that it can be composed with any other computation: the `bind` rule requires that the base of the term and the base of the continuation be equal. It is easy to compute that $\phi(0, \langle p_0, \dots, p_k \rangle) = p_0$, and so it is reasonable to think of a term of type $\mathbb{M}\,\langle p_0, \dots, p_k \rangle\,\tau \equiv \forall j : \mathbb{N}.\mathbb{M}\,(j, \langle p_0, \dots p_k \rangle)\,\tau$ as being a computation of a τ which takes p_0 time: throwing away the higher-order terms of the cost allows us to emulate the original λ-Amor's "constant" cost monad. Practical concerns necessitate the addition of one more construct into the mix: the constant potential vector $\mathtt{const}(I)$, where I is of sort $\mathbb{R}^+$. Intuitively, we think of this as being the potential vector $\langle I, 0, \dots, 0 \rangle$, such that $\phi(n, \mathtt{const}(I)) = I$, for any n.

The final ingredient of this new version of the cost and potential modalities is the ability to change base. To illustrate, consider the process of writing a function $L^n\tau \multimap \mathbb{M}\,(n, \vec{p})\,\sigma$. The recursive call on the tail of the input list will have type $\mathbb{M}\,(n-1, \vec{p})\,\sigma$, but the function expects a return value of type $\mathbb{M}\,(n, \vec{p})\,\sigma$. Since the `bind` requires that the argument and the continuation have the same base, the recursive call cannot be used in this context, rendering it useless. To fix this, we include a term `shift` in dλ-Amor which "promotes" a computation of type $\mathbb{M}\,(n-1, \vec{p})\,\sigma$

[7] This is somewhat inaccurate: the presence of a sum construct in the language of index terms breaks linearity, but this is rarely a problem in practice.

[8] The $\forall j : \mathbb{N}...$ being *irrelevant* is crucial here: since one cannot pattern match on the value of j, the cost of the computation must be uniform in j.

to one of type $\mathbb{M}\,(n, \vec{q})\,\sigma$, for a specific $\vec{q}$ determined by $\vec{p}$. This concept is likely familar to the reader familiar with AARA: in Resource Aware ML (an implementation of OCaml based on AARA) this construct is baked into the pattern match rule, while we make it explicit.

Definition 2.3 (Additive Shift). *For $\vec{p} = (a_0, \ldots, a_{k-1}, a_k)$ a potential vector, we define $\triangleleft\, \vec{p} = (a_0 + a_1, \ldots, a_{k-1} + a_k, a_k)$*

Theorem 2.2. *For $n \geq 1$ and $\vec{p}$ a potential vector, $\phi(n, \vec{p}) = \phi(n-1, \triangleleft\, \vec{p})$*

Proof. Follows from the fact that $\binom{n-1}{i} + \binom{n-1}{i+1} = \binom{n}{i+1}$, and unfolding definitions. $\square$

The shift operator allows us to define the proper type of the `shift` operator: `shift`(e) has type $\mathbb{M}(n, \vec{p})\,\tau$ when e has type $\mathbb{M}(n-1, \triangleleft\, \vec{p})$. This shift in perspective will be of critical importance when performing AARA-style analyses: when required to provide a term of type $\mathbb{M}(n, \vec{p})\,\tau$, we will often find ourselves in posession of only costly computations with base $n-1$, and so it we will be required to `shift` our perspective.

2.2. Cost and Potential Foundations. The more logically-inclined reader is likely to be unsatisfied with the presentation of the potential and cost modalities thusfar. Luckily, the two modalities are far from ad-hoc. In fact, they can easily be explained as being user-optimized instances of a more basic phenomenon. Consider affine types with an additional atomic type P which encodes a single unit of potential. The potential type $[I|\vec{p}]\,\tau$ can be encoded as $P^{\phi(I,\vec{p})} \otimes \tau$, where P^k is the k-fold tensor of P with itself[9]. This is in line with our intuitive understand of the potential type as carrying I potential along with a value of type τ. On the other hand, the cost type $M\,(I, \vec{p})\,\tau$ is encoded as $P^{\phi(I,\vec{p})} \multimap \tau$. This presents a monadic computation which costs up to $\phi(I, \vec{p})$ as a function *requiring* that much potential to be provided in order to produce the result.

Under this framing, all of the operations on costs and potentials in dλ-Amor can be thought of in this way by considering their denotations in this model. For example, the `store` operation, which has type $\tau \multimap \mathbb{M}\,(I, \vec{p})\,([I|\vec{p}]\,\tau)$ can be thought of as having type $\tau \multimap P^{\phi(I,\vec{p})} \multimap P^{\phi(I,\vec{p})} \otimes \tau$, which is plainly the pairing function. Similarly, the `release` function of type $[I|\vec{p}]\,\tau \multimap (\tau \multimap$

[9] This analogy breaks down slightly when $\phi(I, \vec{p})$ is not an integer, but the principle stands.

$\mathbb{M}\,(I, \vec{p} + \vec{q})\,\sigma) \multimap \mathbb{M}\,(I, \vec{q})\,\sigma$ can instead be thought of a function of type:

$$P^{\phi(I,\vec{p})} \otimes \tau \multimap (\tau \multimap P^{\phi(I,\vec{p}+\vec{q})} \multimap \sigma) \multimap P^{\phi(I,\vec{q})} \multimap \sigma$$

which operates by using Theorem 2.1 to equate the types $P^{\phi(I,\vec{p})} \otimes P^{\phi(I,\vec{q})} = P^{\phi(I,\vec{p})+\phi(I,\vec{q})} = P^{\phi(I,\vec{p}+\vec{q})}$.

This "model" of dλ-Amor also serves to motivate two of the fundamental strictures of programming in it. Since we have described no introduction forms for the type P, it is impossible to construct a closed term of that type. This is in line with the intended mental model of potentials in dλ-Amor and amortized analysis more generally: potentials are unobservable, and thus should never "escape" a program. The first consequence of this fact is that there are no closed terms of type $[I]\,\tau$: potential can only occur under a monadic computation. Second, given a function $P \multimap \tau$, one should not be able to recover a τ in a closed context. Correspondingly, a programmer in dλ-Amor can never internally "run" a monadic computation of type $M\,(I, \vec{p})\,\tau$ to get at the underlying τ. This stricture is similar in spirit to Haskell's IO monad, whose terms cannot be evaluated except at the interactive top level.

Next, considering the two modalities in this model makes plain their potentially confusing subtyping rules. While P is not a part of dλ-Amor, its subtyping is the primary source of confusion, as the the actual subtyping rules in λ-Amor are inherited from this model. Intuitively, we think of P as being a subtype of the unit type 1, as one can always discard "an atom of potential". From this, it follows that $P^k <: P^\ell$ when $\ell \leq k$. More potential can always be used in place of less[10]. Note the contravariance in this subtyping rule: the ordering in amounts of potential is the opposite of the ordering on numbers. From this intuitive understanding of the subtyping in this atomic potential model, we can derive the subtyping rules of the cost and potential modalities in dλ-Amor. We begin by supposing that $\vec{q} \leq \vec{p}$. By Theorem 2.1, we have $\phi(I, \vec{q}) \leq \phi(I, \vec{p})$, which implies that $P^{\phi(I,\vec{p})} <: P^{\phi(I,\vec{q})}$ as discussed. For the subtyping rule for potentials, we use the $\otimes$ subtyping rule to get $P^{\phi(I,\vec{p})} \otimes \tau <: P^{\phi(I,\vec{q})} \otimes \sigma$, when $\tau <: \sigma$. However, these types are $[I|\vec{p}]\,\tau$ and $[I|\vec{q}]\,\sigma$, respectively, which justifies our subtyping rule: to show that $[I|\vec{p}]\,\tau <: [I|\vec{q}]\,\sigma$, it suffices to have $\vec{q} \leq \vec{p}$ and $\tau <: \sigma$. For costs, we use the $\multimap$ subtyping rule to get $P^{\phi(I,\vec{q})} \multimap \tau <: P^{\phi(I,\vec{p})} \multimap \sigma$, when $\tau <: \sigma$. This time, the types are $\mathbb{M}\,(I, \vec{q})\,\tau$ and $\mathbb{M}\,(I, \vec{p})\,\sigma$,

[10] Alternatively, this can be thought of as being analogous to width subtyping for records.

which again justifies our eventual rule: to show $\mathbb{M}\,(I,\vec{q})\,\tau <: \mathbb{M}\,(I,\vec{p})\,\sigma$, it suffices to have $\vec{q} \leq \vec{p}$ and $\tau <: \sigma$.

Finally, while this model of the cost and potential modalities is useful, it is by no means complete. In other words, there are types which are inhabited in the model whose counterparts are uninhabited in dλ-Amor. A key example is the lack of a term corresponding to potential "application". In the model, the type $P^k \otimes (P^k \multimap \tau) \multimap \tau$ is inhabited by the λ-term which performs the application. Meanwhile in dλ-Amor, the corresponding type $[I|\vec{p}]\,(\mathbb{M}\,(I,\vec{p})\,\tau) \multimap \tau$ is uninhabited. The only way to use a term with potential is to `release` the potential into a monadic computation, which the target of this function is not. Another example is the lack of a potential fusion law. Since potentials only occur under the cost monad, the potential modality is not itself a monad: there is no term inhabiting $[I|\vec{p}]([I|\vec{q}]\,\tau) \multimap [I|\vec{p}+\vec{q}]\,\tau$. However, the potential join can be written in an ambient monadic context. In other words, there is a term of type $[I|\vec{p}]([I|\vec{q}]\,\tau) \multimap \mathbb{M}\,(I,\vec{0})\,([I|\vec{p}+\vec{q}]\,\tau)$.

3. Examples of Programs in dλ-Amor

In this section, we will present a number of examples of programs written in dλ-Amor, each of which exemplifies a different component of its cost analysis features. While we have not formally introduced the syntax of dλ-Amor yet, we provide a simple term for the first example in Figure 2 to illustrate the way programs are intertwined with their proofs. These examples will loosely follow the presentation of Section 3 of Rajani et al. [64], where more in-depth discussion can be found.

3.0.1. *Add One.* We begin with a (very) simple example to demonstrate the utility of dλ-Amor's AARA-style costs. Consider writing a function `addOne`, which adds one to each integer in a list. If we assume the cost model that natural number addition costs one unit of time, the function would have type $\forall n : \mathbb{N}.\, L^n(\texttt{nat}) \multimap \mathbb{M}\,(n,\langle 0,1\rangle)\,(L^n(\texttt{nat}))$. Recalling the intended meaning of the AARA-style cost functions, this means that `addOne` costs $\phi(n,\langle 0,1\rangle) = n$ in total, where n is the length of the input list (and also the output). This makes sense, as each entry in the list incurs a single cost to add one to it. Pseudocode of the term for this type can be found in Figure 2. The operational aspects of the program are exactly what one expects from an instance of map. More interesting are the cost-related aspects of the

```
fix(addOne.$\Lambda n.\lambda xs.$match($xs$, ret([]), $y.ys.$
    shift(
      bind $zs$ = addOne [$n-1$] $ys$ in
      bind _ = tick[$n-1|\langle 1,0\rangle$] in
      ret(($y+1$) :: $zs$)
    )
))
```

FIGURE 2. `addOne` function in dλ-Amor

code. In the cons branch, we immediately `shift`. This allows us to provide a term of type $\mathbb{M}\,(n-1,\langle 1,1\rangle)\;(L^n(\texttt{nat}))$ in place of the expected type $\mathbb{M}\,(n,\langle 0,1\rangle)\;(L^n(\texttt{nat}))$. Although this is guaranteed to be by Theorem 2.2, we can check that this is sound by computing that $\phi(n-1,\langle 1,1\rangle) = (n-1)+1 = n = \phi(n,\langle 0,1\rangle)$. This shift is required to perform the recursive call on the tail: `addOne` $[n-1]\,ys$ has type $\mathbb{M}\,(n-1,\langle 0,1\rangle)\;\big(L^{n-1}(\texttt{nat})\big)$, which can only be bound into a continuation which results in something of type $\mathbb{M}\,(n-1,\_)\,\_$. Further, the shift "exposes" the one constant cost, which is incurred by the tick (which we attribute to the addition). This raises a crucial point: a "hole" in a program expecting $\mathbb{M}\,(n,\langle 0,1\rangle)\,\tau$ cannot accept a term of type $\mathbb{M}\,(n,\langle 1,0\rangle)\,\tau$ for any n, despite this being semantically sound for $n \geq 1$.

This example can also be performed using potentials, rather than costs. Instead of a function which incurs n cost, we can instead think of `addOne` as a free-to-execute function which expects n potential. One possible choice for this function's type is:

$$\forall n:\mathbb{N}.\,[n|\langle 0,1\rangle]\,1 \multimap L^n(\texttt{nat}) \multimap \mathbb{M}\,(n,\langle 0,0\rangle)\;(L^n(\texttt{nat}))$$

This style is reminiscent of the "gas-cost" analyses from Chapter 1, as we expect n gas up front to run, and spend it all towards performing the additions. For technical reasons relating to expressivity[11], we often use this style (preferring the type $[I]\,\tau \multimap \mathbb{M}\,0\,\sigma$ over the type $\tau \multimap \mathbb{M}\,I\,\sigma$) even in cost analyses which are not amortized.

Another option is a type which attaches a single potential to each element of the input list, in a style indicative of the Banker's method:

$$\forall n:\mathbb{N}.\,L^n\,([1]\,\texttt{nat}) \multimap \mathbb{M}\,(n,\langle 0,0\rangle)\;(L^n(\texttt{nat}))$$

[11] In short, coeffect-style analyses require the use of potentials.

This cost analysis is tight and fairly uninteresting: it requires no "real" amortized analysis. To illustrate how dλ-Amor handles describing cost analyses for programs where amortization is required for tight bounds, we show how a few classic examples of amortized analysis can be written in dλ-Amor.

3.0.2. *Insertion Sort.* Our second example will illustrate how the AARA-style costs of dλ-Amor will allow us to verify quadratic-and-higher cost bounds, while only ever solving linear constraints. Insertion sort is a good example of this class of program, since its cost analysis is very understandable (nested loops, nothing fancy), while still being an interesting function.

For insertion sort, we will assume the traditional cost metric for sorting algorithms: all comparisons cost one unit. The insertion sort we will write will be monomorphic, and assume a comparison operator of type

$$\texttt{leq} : \tau \otimes \tau \multimap \mathbb{M}\,\langle 1 \rangle\, 2$$

where 2 is defined to be $1 \oplus 1$.

Since elements of the list will have to be compared multiple times, we will use the exponential modality ($!\tau$ in Chapter 1) to have insertion sort operate over lists of infinite-use τs. With this in mind, we give the insertion function the following type:

$$\texttt{insert} : \forall n : \mathbb{N}.\, !\tau \multimap L^{n}\,(!\tau) \multimap \mathbb{M}\,(n, \langle 0, 1\rangle)\,\left(L^{n+1}\,(!\tau)\right)$$

The type of this function should be intuitive: at worst, we scan the list once, incurring $n = \phi(n, \langle 0, 1\rangle)$ cost. Folding this function over a list yields the insertion sort algorithm, which has the type shown below.

$$\texttt{ins\_sort} : \forall n : \mathbb{N}.\, L^{n}\,(!\tau) \multimap \mathbb{M}\,(n, \langle 0, 0, 1\rangle)\,(L^{n}\,(!\tau))$$

We can compute that $\phi(n, \langle 0, 0, 1\rangle) = \binom{n}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$, which gives `ins_sort` the requisite quadratic cost bound. This cost analysis is fairly elementary, but it's important to note the design of dλ-Amor (via AARA) is what makes this analysis possible. Because the constraints on polynomial functions are only ever coefficient-wise and linear, checking the corresponding terms (found in Appendix A) is easy for SMT solvers, despite the cost bound being quadratic.

3.0.3. *Functional Queue.* The first example of amortized analysis is the traditional functional queue [58]. Here, a queue is represented as a pair of lists, l_f and l_r, which we refer to as the front and rear lists, respectively. To enqueue an element, we cons it to the head of the front list, and to dequeue, an element is removed from the head of the rear list. If the rear list is empty when a dequeue operation is issued, the front list is reversed into the rear.

If we assume that cons operations are the only costly operation, and that they each incur one cost, this dequeue operation has worst-case complexity $O(n)$ where n is the size of the queue (the sum of the sizes of l_f and l_r). While "most" calls to dequeue will be $O(1)$, the worst case is $O(n)$ since the function needs to reverse the entire front list whenever the rear list is emppty. However, by employing the banker's method, we may enforce the invariant that each element of the front list carries two credits to be used to pay for its eventual reversal. Under this scheme, both enqueue and dequeue are constant amortized time.

This entire informal analysis is captured formally by the types of the enqueue and dequeue operations in dλ-Amor. To encode this analysis, we define a queue to be of type $L^n([2]\,\tau) \otimes L^m\,\tau$: a pair of τ-lists, where the front has 2 potential on each of its n elements.

The enqueue function has the following type.

$$\texttt{enq} \;:\; \forall n, m : \mathbb{N}.\, [3]\,1 \multimap \tau \multimap L^n([2]\,\tau) \otimes L^m\,\tau \multimap \mathbb{M}\,\langle 0\rangle\, \left(L^{n+1}([2]\,\tau) \otimes L^m\,\tau\right)$$

From a queue and three extra potential, we may enqueue a single element, resulting in queue with one more element on its front list, for no cost. The term implementing `enc` can be found in Appendix A. The type of dequeue is somewhat more involved, since the sizes of the output lists are not a simple function of this inputs. In addition, the function has a precondition: the queue cannot be empty. These two numerical restrictions provide a nice illustration of dλ-Amor's refinement types.

$$\begin{aligned}\texttt{deq} \;:\; \forall m, n : \mathbb{N}.(m + n > 0) \implies\; & L^n([2]\,\tau) \otimes L^m\,\tau \multimap \\ & \mathbb{M}\langle 0\rangle \left(\exists n', m' : \mathbb{N}.(n' + m' + 1 = n + m) \,\&\, \left(L^{n'}([2]\,\tau) \otimes L^{m'}\,\tau\right)\right)\end{aligned}$$

`deq` takes a nonempty queue, and produces another queue which has one element removed. The implementation of `deq` relies on a function `move`, which reverses the rear list into the front. The terms for all functions involved can be found in Appendix A.

3.0.4. *Cost-Parametric Map.* While many existing languages and type systems for (amortized) resource analysis also support higher-order functions, the allowable analyses with higher-order functions are limited. One such limitation is that function arguments to higher-order functions are usually assumed to be constant-cost: for instance, in the cost analysis of a map, each application of the mapping function is assumed to incur the same amount of cost.

To improve on this, we employ a cost family $C : \mathbb{N} \to \mathbb{R}^+$ to encode the costs of each application of the function: the i-th call to the function is thought to incur $C(i)$ cost. Then in total, the map function incurs $\sum_{0 \leq i < n} C(i)$ cost. This analysis is reified in the type of map:

$$\begin{aligned}\texttt{map} \,:\, & \forall \alpha, \beta : \star . \forall C : \mathbb{N} \to \mathbb{R}^+ . \forall n : \mathbb{N}. \\ & !\,(\forall i : \mathbb{N}.[C\, i]\, 1 \multimap \texttt{Nat}(i) \multimap \alpha \multimap \mathbb{M}\,\langle 0 \rangle\, \beta) \multimap \\ & !\texttt{Nat}(n) \multimap \\ & L^n\, \alpha \multimap \\ & \mathbb{M}\,\langle \texttt{const}\,(\textstyle\sum_{0 \leq i < n} C(i)) \rangle\; (L^n\, \beta)\end{aligned}$$

Most importantly, the mapping function has type $!\,(\forall i : \mathbb{N}.[C\, i]\, 1 \multimap \texttt{Nat}(i) \multimap \alpha \multimap \mathbb{M}\,\langle 0 \rangle\, \beta)$. Since it must be applied to each element of the list, its type is !-ed to ensure it may be duplicated. The function is parameterized by the index i on which it operates. To ensure that the mapping function at i is actually only ever used at index i, the mapping function takes an additional argument of type $\texttt{Nat}(i)$, which is the singleton type of natural numbers equal to i[12]. Finally, the mapping function requires $C\, i$ potential to run, and incurs no amortized cost, which ensures that its actual cost is bounded by $C\, i$.

Given the mapping function, the function `map` then transforms an $L^n\, \alpha$ into a monadic computation of an $L^n\, \beta$, incurring $\sum_{0 \leq i < n} C(i)$ amortized cost. As usual, the term implementing map can be found in Appendix A

4. Overview of biλ-Amor

For dλ-Amor to be useful as a programming language, it must be implementable! While a declarative type system on paper is useful for modeling and proving purposes, it has limited utility from a language engineering standpoint. dλ-Amor is far more implementation-ready than its predecessor λ-Amor, but the rules of its type system do not provide us with an obvious implementation method. Traditionally, one hopes to implement a type system in a manner similar to implementing a definitional interpreter [66]. For each judgment of the type system, the programmer writes a function which essentially runs a non-backtracking proof search for that judgment, with the type position as an input or output depending on if the function is type checking or inference.

[12] This is simply an alias for $L^i\, 1$.

Unfortunately, many of the judgments of dλ-Amor— which we will see when the formalism is presented in Section 5 — do not have straightforward implementations. The difficulties of coming up with implementations stems from five critical challenges which must be overcome before we can implement dλ-Amor. Our eventual solutions to these five challenges form the basis of biλ-Amor, the *algorithmic* version of dλ-Amor which we will subsequently implement.

(1) The main type-assignment judgment of dλ-Amor yields an ambiguous proof search method. It is not at all clear which rule to apply at any given step of building a derivation, since there are some rules (subtyping, weakening) that can always be tried at each stage. Indeed, one could always implement proof search for dλ-Amor using backtracking, but it is preferable to avoid this if possible. Instead, we would like our implementation-ready calculus biλ-Amor to be *syntax-directed* in the sense that the outermost syntax of the current term informs us which typing rule must be applied next to build a successful derivation.

(2) dλ-Amor includes full System F impredicative polymorphism, but a well-known result of Wells [83] states that type inference for System F is undecidable. Hence, we will not be able to design a type inference algorithm for dλ-Amor. A natural second option is to shoot for implementing a type checker. Unfortunately, this too has its limitations. To implement proper type checking, the syntax of dλ-Amor would have to be changed such that every variable binder includes a type annotation. This is a heavy burden on the programmer: annotating binders with types is tedious, error prone, and generally uninteresting[13]. Instead, biλ-Amor adopts *bidirectional type checking*, a technique pioneered by Pierce and Turner [62] which trades off some of the generality of full type inference for added ergonomics over standard type checking.

(3) dλ-Amor's subtyping relation provides a challenge which should be familiar to the reader who is versed in the implementation of dependent type theories. The inclusion of indexed types means that the deciding the subtyping relation requires (essentially) deciding β equality at the type level: for instance, establishing the subtyping relation $(\lambda i : \mathbb{N}.L^i\,\tau)\,3 <: L^3\,\tau$ requires a step of β-reduction. Luckily, the equational theory of types is simpler than that of a simply-typed lambda calculus, since the type-level

[13] As Pierce [61] notes: "The more interesting your types get, the less fun it is to write them down!"

lambda in dλ-Amor $\lambda i : S.\tau$ ranges over index terms, not types. This allows for a very simple single-pass normalization procedure which allows us to subsequently decide the subtyping relation.

(4) Many of the crucial rules of the dλ-Amor subtyping relation include constraint validity premises: for instance, the rule for deriving subtyping of potential types $[I|\vec{p}]\,\tau <: [I|\vec{q}]\,\sigma$ has $\vec{q} \leq \vec{p}$ as a premise. These premises will need to be discharged by an SMT solver. However, repeatedly pausing the subtyping algorithm to send constraints to a solver each time the premise of a rule requires one be verified is inefficient. Instead, we would prefer to do one pass of typechecking, followed by a single call to the solver. To achieve this, the judgments of biλ-Amor "output" constraints. The intended meaning of this is that when the constraints are valid, the declarative version of the same judgment is derivable.

(5) The final barrier to implementation comes not from the refinement type or cost analysis features of dλ-Amor, but simply from the fact that it is an affine type system. Multi-premise rules require the context to the type-checker be split into disjoint parts which can be used by each premise. This choice is nondeterministic: there is no way to know a priori what allocation of resources to which premise until later. To solve this, we employ a classical technique for implementing substructural type systems, which we refer to as the IO method [11].

In the rest of the section, we present the solutions to these five problems that we choose to adopt. All five solutions are well-known techniques, but to our knowledge biλ-Amor is the language to show that they may all be simultaneously integrated into a single system. Since the five techniques are orthogonal, we present each feature of biλ-Amor in isolation for a significantly simpler language. In Section 7, we will present the formalism for biλ-Amor, which applies the below-presented techniques to the declarative calculus dλ-Amor.

4.0.1. *Bidirectional Type Systems.* Bidirectional type inference, also known as "local type inference" is a type system algorithmization technique pioneered by Pierce and Turner [62]. The technique works by separating the typing judgment $\Gamma \vdash e : \tau$ of a declarative type system into two algorithmic judgments: $\Gamma \vdash e \downarrow \tau$ and $\Gamma \vdash e \uparrow \tau$, which are read "$e$ checks against τ" and "e infers τ" (sometimes "synthesizes"), respectively. These two judgments are mutually-recursively defined in a specific manner. The process of turning a declarative type system into a bidirectional algorithmic one is straightforward to the point of mechanical: Dunfield and Pfenning [22] provide a simple-to-follow recipe for this conversion, which extends from the simple type system they consider all the way to dλ-Amor.

Syntax-directed algorithmic type systems presented in a bidirectional style are trivially implementable: the implementation strategy is built into the structure of the rules. To implement a bidirectional type system, one writes two mutually-recursive functions `check:ctx->tm->typ->unit` and `infer:ctx->tm->typ` by recursion on the term input: the recursive calls are guided by the premises of each rule. Note that the types of these functions indicate the intended *modes* of the three positions of the judgment, in the sense of logic programming. In the checking judgment, all positions are imagined to be *inputs*, while the inference judgment indicates that the type position is an *output* of the judgment.

As alluded to earlier, the "inference" of the judgment $\Gamma \vdash e \uparrow \tau$ is not full inference, but merely "local" inference: this judgment is derivable when enough information is present in the form of e to determine its type. This is in contrast to full type inference, where the type of a term may not be fully known until its typing constraints are considered in the context of those from the larger term in which it sits. For this reason, every syntactic form in the language has either an inference or checking rule: if requiring one of the premises to be inference gathers enough information to determine the type of the conclusion, then that conclusion will be an inference judgment. Otherwise, the judgment will be checking.

To mediate between the two judgments, bidirectional type systems include two special rules. First, is the rule which is traditionally referred to as "subsumption": to show that $e \downarrow \tau$, it suffices to show that $e \uparrow \tau$. In other words, if e can infer a type, then it checks against that type. This rule is usually strengthened by subtyping:

$$\frac{\Gamma \vdash e \uparrow \tau' \quad \tau' <: \tau}{\Gamma \vdash e \downarrow \tau}$$

For e to check against τ, it suffices for e to synthesize a more precise type τ'.

Going in the other direction from a checking premise to an infering conclusion is somewhat more involved. In general, the desired converse rule is not true: there will always be terms such that $e \downarrow \tau$ but it is not the case that $e \uparrow \tau$. To remedy this, bidirectional type systems introduce a new piece of syntax to the declarative language on which they're based: annotations. When e checks against τ, the annotated term $(e : \tau)$ infers the type τ:

$$\frac{\Gamma \vdash e \downarrow \tau}{\Gamma \vdash (e : \tau) \uparrow \tau}$$

These annotations must be manually added to terms by the programmer as they write the program. However, the only place where annotations are truly required are at the sites of *bare β-redexes*. For example, to check the term $(\lambda x.e)\, e'$, it must be annotated as $(\lambda x.e : \tau \to \sigma)\, e'$. Since most programs only contain bare β-redexes in the form of let-bindings, this requirement is both predictable and fairly ergonomic. In fact, the only type that truly must be annotated is the so-called "cut type", the τ in $(\lambda x.e : \tau \to \sigma)\, e'$. This is because when checking $(\lambda x.e)\, e'$ against σ, the type of e *must* be σ, and the only unknown type is type of e' (and x). In the eventual implementation, we include an annotated let-binding construct `let x :  t = e in e'` for exactly this reason.

It is important to remember that these annotations are *not* present in a declarative syntax. It will eventually be useful (when discussing the relation between biλ-Amor and dλ-Amor) to have the ability to talk about the "underlying" declarative term of an algorithmic term, which is achieved by simply removing all type annotations $(e : \tau)$ from a term. We usually denote this $|e|$, when e is an algorithmic term, and sometimes refer to it as the *erasure* of a term. The erasure can be trivially defined by recursion on raw terms, with the critical case being $|(e : \tau)| = |e|$.

As of yet, the relationship between a declarative calculus and its bidirectional algorithmic counterpart has been left unstated. However, the point of the bidirectional calculus is to be able to algorithmically generate declarative derivations! To this end, one always requires that the bidirectional type system be *sound* for the declarative one.

THEOREM 4.1 (Bidirectional Soundness). *If $\Gamma \vdash e \downarrow \tau$, then $\Gamma \vdash e : \tau$*

In other words, successfully running `check`(Γ, e, τ) is sufficient to show that e in fact has type τ in context Γ.

Conversely, completeness is also desirable, but not strictly necessary for bidirectional type systems. The most obvious statement of completeness ($\Gamma \vdash e : \tau$ implies $\Gamma \vdash e \downarrow \tau$) does not, in fact, hold. If the term e contains un-annotated explicit β-redexes, the algorithmic system will fail to infer function types in those positions. For this reason, the following slightly weaker theorem is used as the completeness result for bidirectional type systems.

THEOREM 4.2 (Bidirectional Completeness). *If $\Gamma \vdash e : \tau$, then there exists e' such that $\Gamma \vdash e' \downarrow \tau$, and $|e'| = e$, where $|e'|$ is the annotation-erasure of e'.*

When proven constructively, this completeness result encodes an algorithm which inserts annotations into the term e so that the resulting term checks against τ. When Theorem 4.2 is proven directly by induction, the algorithm it encodes introduces far more annotations than is often strictly necessary. We improve on this with our completeness proof of biλ-Amor in Section 8 by proving an equivalent statement whose constructive proof inserts fewer annotations than the standard theorem.

4.0.2. *Algorithmic Subtyping and Normalization.* To implement the subsumption rule mentioned above, a decision procedure for the subtyping relation $\tau <: \tau'$ is required. However, dλ-Amor's subtyping is not immediately implementable for two important reasons.

Firstly, like dλ-Amor's typing relation, it is not syntax directed: the relation includes two rules (reflexivity and transitivity) that can be used at any step of a derivation. To avoid a backtracking implementation, it will be necessary to design an algorithmic subtyping relation for biλ-Amor which includes neither of these rules. Of course, the algorithmic subtyping will need to be sound and complete for the declarative one. This requirement means that the algorithmic subtyping relation will need to have reflexivity and transitivity as admissible rules: in effect, we will need to prove identity and cut elimination.

The second (and more pernicious) problem is the inclusion of indexed types. While many refinement type systems (including DML [84], on which λ-Amor's refinement types are based) include indexed types [85], they are usually implemented only as types of the form $\forall i : S.\tau$. While useful, these indexed types are limited: their abstraction and application is controlled by term-level introduction and elimination rules. Instead, dλ-Amor includes indexed types of the form $\lambda i : S.\tau$, which operate entirely at the type level, without programmer input required. The inclusion of type-level abstractions and applications does require the subtyping relation to include β equalities for these indexed type families: without them, the subtyping relation would not be able to judge relations like $(\lambda i : \mathbb{N}. L^i\,\tau)\,3 <: L^3\,\tau$, where the subtying relation holds up to β equality.

The inclusion of β-conversion makes a simple algorithmic subtyping relation unlikely, since any way of deciding declarative subtyping must also decide β-equality of this small λ-calculus at the type level. However, the situation is sufficiently simple that we can get away with a fairly low-powered solution. To this end, biλ-Amor's subtyping relation is split into two phases. First, both types are evaluated (or *normalized*) to normal forms, and then judged for subtyping by a relation which only contains the congruence rules. These normal forms are only normal insofar as they contain no type-level β-redexes. Since index terms only become important at constraint-solving time, we do not require that the index terms appearing in a normal form type be normal in any sense. This notion of normal form has a major benefit: since abstractions $\lambda i : S.\tau$ range over index terms and not types, a β reduct has strictly fewer type connectives than its redex. For this reason, the normalization can be implemented in a single pass: substituting an index term for a free variable in a type in normal form yields another type in normal form. The two-phase algorithmic subtyping relation, as well as the normalization proof, are discussed in detail in Section 7.1

4.0.3. *Constraint Generation.* As motivated in Section 2, most of the changes to λ-Amor that result in dλ-Amor are there for the purpose of simplifying the constraint-solving process that arises as a part of subtyping. Efficiently handling these constraints is crucial to an efficient implementation. For this reason, it is useful to defer the discharging of inference rules' constraint validity premises until *after* the typechecking pass has finished.

We operationalize this in biλ-Amor by designing each judgment to "output" a constraint: we replace declarative judgments $\mathcal{J}$ with algorithmic ones $\mathcal{J} \Rightarrow \Phi$, where Φ is a constraint, thought of as an output of the judgment. For instance, a constraint-emitting algorithmic version of a declarative type-assignment judgment $\Gamma \vdash e : \tau$ might look like $\Gamma \vdash e : \tau \Rightarrow \Phi$. The intended meaning of this (and the shape of the soundness theorem for an algorithmic judgment with a constraint output) is that if we can derive the algorithmic judgment $\mathcal{J} \Rightarrow \Phi$ and Φ holds, then the declarative judgment $\mathcal{J}$ is derivable.

This scheme is pervasive. Since every judgment in dλ-Amor either has a rule with a constraint validity premise or depends on one that does, every judgment in biλ-Amor must emit constraints. The pattern in transforming a declarative judgment to an algorithmic one which emits constraints is fairly uniform: the output constraint of a rule is essentially the conjunction of the constraints output by its premises. One must also ensure that implications and quantifiers are inserted for constraints and index variables bound in premises: the logical structure of the output constraint mirrors the structure of the premises.

The constraints Φ that each judgment emits are drawn from a syntax of logical formulae, which is outlined in Figure 3 in Section 5. This syntax is very general, allowing unrestricted universal and existential quantification, the usual truth-functional logical connectives, as well as equalities and inequalities between index terms. While this theory is (almost certainly) not decidable, SMT solvers handle it well enough for practical purposes, as we will see in Section 9.

4.0.4. *I/O Method.* On top of the implementation challenges created by the fancier aspects of dλ-Amor's type system, its affine-ness presents a well-understood barrier to implementation. To illustrate, consider writing the following case of the `check:ctx->tm->typ->unit` function from earlier.[14]

```
check gamma Pair(e1,e2) Tensor(t1,t2) = ??
```

This case corresponds to the introduction rule for tensor,

$$\frac{\Gamma_1 \vdash e_1 : \tau_1 \quad \Gamma_2 \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \vdash (e_1, e_2) : \tau_1 \otimes \tau_2}$$

It is not at all clear how to proceed in this case. The tensor introduction rule prescribes that we make two recursive calls `check gamma1 e1 t1` and `check gamma2 e2 t2`, but provides no direction how to obtain `gamma1` and `gamma2` from `gamma`: the rule is presented in the standard way so that the two halves of the context are given at the outset.

This problem has two naive solutions. Firstly, one could analyze the structure of `e1` and `e2` to determine the variables they each use, and partition the context accordingly. This approach is very inefficient: even if done with a pre-processing step, it adds at least one pass through the term. Secondly, one could split the context *symbolically*, and generate yet more constraints to unify at the end of the typechecking process.

[14] To simplify some of the presentation of this sub-section, we will specialize to the non-bidirectional setting, and work in a simply-typed language where binders are fully annotated.

Instead of either of these, we adopt a more principled approach based on the work of Cervesato et al. [11]. In short, we extend the main typing judgment with yet another output — this time a second context, which contains the variables which were unused in typing the term. A simplified version of the typing judgment takes the form $\Gamma \vdash e : \tau \Rightarrow \Gamma'$, where Γ is the *input context*, and Γ' is the *output context*. The key idea of this setup (known sometimes as the I/O method) is that we may thread the contexts through the premises of a rule as follows:

$$\frac{\Gamma \vdash e_1 : \tau_1 \Rightarrow \Gamma_1 \quad \Gamma_1 \vdash e_2 : \tau_2 \Rightarrow \Gamma_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \otimes \tau_2 \Rightarrow \Gamma_2}$$

The first premise (the first component of the pair) has access to the entire input context, and it outputs Γ_1, the variables in Γ which were unused in typing e_1. This context is then used as the *input* context for checking e_2. Since affine variables may be used at most once, the only variables which e_2 may access are those unused by e_1. This property is enforced in declarative systems "in parallel" by splitting the context up front, but it may similarly be enforced "sequentially" by lazily deciding which premises may use which variables in this algorithmic style.

The key rule in designing an algorithmic type system which uses the I/O method is the affine variable rule. When a variable is used, it must be removed from the output context:

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau \Rightarrow \Gamma \setminus \{x\}}$$

Moreover, this I/O method will be trivial to implement. We simply change the type of `check` and `infer` to output a context as well. Since these functions both receive and output a context, one can think of typechecking with the I/O method as happening inside a state monad of contexts, as opposed to the usual reader monad. While this solution is clearly preferable to the naive ones efficiency-wise, it is not at all obvious that this way of algorithmizing an affine type system is sound, much less complete, for the standard presentation of the rules.

Writing $\Gamma \vdash e : \tau$ (with no output context) as the declarative typing relation, a first cut at a soundness theorem for this calculus is the following:

THEOREM 4.3 (First Cut at Soundness of the I/O Method). *If* $\Gamma \vdash e : \tau \Rightarrow \Gamma'$, *then* $\Gamma \vdash e : \tau$

While true, the statement of this theorem isn't strong enough to be proven by a direct induction. Intuitively, Γ contains variables that are unused in the typing of e, namely those in

Γ'. Thus, it makes sense to strengthen the conclusion to type e in a context with only the free variables it mentions, namely $\Gamma \setminus \Gamma'$.

THEOREM 4.4 (Soundness of the I/O Method). *If* $\Gamma \vdash e : \tau \Rightarrow \Gamma'$*, then* $\Gamma \setminus \Gamma' \vdash e : \tau$

Note that $\Gamma \setminus \Gamma'$ is well-defined because $\Gamma' \subseteq \Gamma$, a fact which must be proven by induction over the algorithmic rules. The completeness theorem is simpler to state, but harder to prove.

THEOREM 4.5 (Completeness of the I/O Method). *If* $\Gamma \vdash e : \tau$*, then there is some* Γ' *such that* $\Gamma \vdash e : \tau \Rightarrow \Gamma'$

The proof of this theorem relies on the fact that weakening is also admissible for an algorithmic type system using the I/O method. When new variables are added to the input context, they simply "flow through" the judgment to the output context, and are left unused.

THEOREM 4.6 (Admissibility of Weakening for the I/O Method). *If* $\Gamma \vdash e : \tau \Rightarrow \Gamma'$*, then for all* Γ''*, we have that* $\Gamma, \Gamma'' \vdash e : \tau \Rightarrow \Gamma', \Gamma''$

5. Syntax and Type System of dλ-Amor

In this section, we will finally rip off the band-aid and begin to discuss the formal system that makes up dλ-Amor. All of the main motivating concepts for the design of this formalism have been discussed in the previous sections. When the cost monad and potential modalities with their AARA-style representations, refinements, and polymorphism are combined, we have a recipe for some potentially convoluted inference rules. The puzzled reader is encouraged to refer to Xi [84] for an account of similar material, without any of the added burden of resource-tracking features.

Since dλ-Amor is only a minor revision of λ-Amor, the structure of its formalism closely follows that of the original found in Rajani et al. [64]. We will begin by presenting its syntax, which differs from λ-Amor only in its added term-level index and type variables, which are added for syntax-directedness purposes. Next, we discuss dλ-Amor's type system, which mirrors that of λ-Amor judgment-for-judgment. The only major difference between the systems is that we take some extra care to pin down the invariants and presuppositions of each judgment: this extra level of precision and formality is required to prove some of the proof-theoretic properties of dλ-Amor with which we conclude the section.

Base Sort	bS	::=	$\mathbb{N} \mid \mathbb{R}^+ \mid \vec{\mathbb{R}^+}$
Sort	S	::=	$bS \mid bS \to S$
Kinds	K	::=	$\star \mid S \to K$
Constants	c	::=	$n \in \mathbb{N} \mid r \in \mathbb{R}^+ \mid (c_0, \ldots, c_k) \in \vec{\mathbb{R}^+}$
Index Term	$I, J, \vec{p}, \vec{q}$	::=	$0, 1, \ldots \mid i, j \mid I + J \mid I - J \mid k \cdot I \mid \lambda i : bS.I \mid I\ J$
			$\mid \texttt{const}(I) \mid \sum_{i=I_0}^{I_1} J$
Constraint	Φ	::=	$\top \mid \bot \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \to \Phi_2 \mid \forall i : S.\Phi \mid \exists i : S.\Phi$
			$\mid I = J \mid I \leq J$
Type	τ	::=	$1 \mid \alpha \mid \tau_1 \multimap \tau_2 \mid \tau_1 \otimes \tau_2 \mid \tau_1 \& \tau_2 \mid\ !\tau \mid \tau_1 \oplus \tau_2 \mid \forall i : S.\tau$
			$\mid \exists i : S.\tau \mid \forall \alpha : K.\tau \mid L^I \tau \mid \Phi \implies \tau \mid \Phi \& \tau \mid \mathbb{M}(I, \vec{p})\ \tau$
			$\mid [I \vert \vec{p}]\ \tau \mid [I]\ \tau \mid \lambda i : S.\tau \mid \tau\ I$
Expression	e	::=	$0, 1, \ldots \mid x \mid \lambda x.e \mid e_1\ e_2 \mid (e : \tau) \mid \langle\langle e_1, e_2 \rangle\rangle$
			$\mid \texttt{let}\ \langle\langle x, y \rangle\rangle = e\ \texttt{in}\ e' \mid (e_1, e_2) \mid \texttt{fst}(e) \mid \texttt{snd}(e) \mid\ !e$
			$\mid \texttt{let}\ !x = e\ \texttt{in}\ e' \mid \texttt{inl}(e) \mid \texttt{inr}(e) \mid \texttt{case}(e, x.e_1, y.e_2)$
			$\mid \texttt{case}(e, e_1, x.e_2) \mid \texttt{fix}(x.e) \mid \Lambda i.e \mid e\ [I] \mid \Lambda \alpha.e \mid e\ [\tau]$
			$\mid \texttt{nil} \mid e_1 :: e_2 \mid \texttt{match}(e, e_1, x.y.e_2) \mid \texttt{pack}[I](e)$
			$\mid \texttt{unpack}\ (i, x) = e\ \texttt{in}\ e' \mid \Lambda.e \mid e\ \{\} \mid\ < e >$
			$\mid \texttt{clet}\ x = e\ \texttt{in}\ e' \mid \texttt{ret}(e) \mid \texttt{bind}\ x = e\ \texttt{in}\ e' \mid \texttt{tick}[I \vert \vec{p}]$
			$\mid \texttt{store}[I \vert \vec{p}](e) \mid \texttt{store}[I](e) \mid \texttt{release}\ x = e\ \texttt{in}\ e'$
			$\mid \texttt{shift}(e)$
Type Variable Context	Ψ	::=	$\cdot \mid \Psi, \tau : K$
Index Variable Context	Θ	::=	$\cdot \mid \Theta, i : S$
Constraint Context	Δ	::=	$\cdot \mid \Delta, \Phi$
Term Variable Context	Γ, Ω	::=	$\cdot \mid \Gamma, x : \tau$

FIGURE 3. Syntax of dλ-Amor

5.1. Syntax of dλ-Amor. In preparation to discuss dλ-Amor's type system, we present its syntax in Figure 3.

5.1.1. *Index Terms, Sorts, Kinds, and Constraints.* dλ-Amor's refinement types are modeled in the style of DML, which takes the form of a two-level type system. As discussed in Section 2, these refinements allow the user to assign potential to types which depends on the sizes of data structures, such as the lengths of lists. These numerical values are denoted by *index terms* (I, J) which decorate some of the types and surface syntax of dλ-Amor. Index terms may be of three possible numerical *base sorts*: natural numbers $\mathbb{N}$, positive real numbers $\mathbb{R}^+$, and potential vectors of some fixed length k, $\vec{\mathbb{R}^+}$. Additionally, dλ-Amor also includes first-order sort-level functions.

The syntax of index terms themselves is generated by the standard arithmetic operations, along with constants, variables, and application/abstraction forms for the sort-level functions. Of special note are the `const` and Σ constructs. For an index term I of sort $\mathbb{R}^+$, the term $\texttt{const}(I)$ is of potential vector sort, and may be thought of as the "constant" potential vector $(I, 0, \ldots, 0)$, such that for all $n\mathbb{N}$, $\phi(n, \texttt{const}(I)) = I$. The Σ construct is as expected, although the upper bound is non-inclusive: the sum $\sum_{i=I_0}^{I_1} J$ sums from $J[I_0/i]$ to $J[(I_1 - 1)/i]$, as long as the range is nonempty, when the sum is of course zero[15].

dλ-Amor also supports full System F-style impredicative polymorphism, as well as sort-indexed types. We denote the kind of types as $\star$. Note that sort-indexed types may have sort-level arrows in negative position, and so sort-function-indexed types are included also.

Finally, dλ-Amor includes constraints over index terms, which can be combined with conjunction, disjunction, implication, and both kinds of quantification. We will not provide a proof system for these constraints. Instead, we will only ever interact with constraints via an abstract validity relation $Theta; \Delta \vDash \Phi$, and all the proofs of soundness and completeness in Section 8 will be relative to a decision procedure/oracle for this relation $\vDash$. In the implementation of `LambdaAmor`, we think of this relation parameter of the type system as being instantiated by the SMT solver backend, which does a good enough job of deciding the theory for practical purposes.

[15] Including arbitrary sums in index terms has the consequence that the constraints from subtyping may be nonlinear. In practice this is rarely an issue — sums are only needed in specific analyses such as cost-parametric map/fold and church numerals.

5.1.2. *Types.* dλ-Amor's types include all of the standard connectives from affine logic, namely positive and negative products ($\otimes$ and $\&$), sums ($\oplus$), affine functions ($\multimap$), and the exponential modality $!\tau$. Of course, dλ-Amor also supports a litany of more specialized types for amortized cost analysis.

Chief among these are the cost monad and potential types, A monadic type $M\,(I,\vec{p})\,\tau$ classifies monadic computations of type τ, which may incur up to $\phi(I,\vec{p})$ cost. The type formation rules ensure that I is of sort $\mathbb{N}$, and $\vec{p}$ is of sort $\vec{\mathbb{R}^{+}}$. With the same restrictions on the sorts of its index terms, the potential type $[I|\vec{p}]\,\tau$ classifies values with at least $\phi(I,\vec{p})$ potential. In addition to the AARA-style potential, dλ-Amor also has a "constant potential" modality $[I]\,\tau$, whose values are those of type τ, with $I = \phi(n, \texttt{const}(I))$ potential, for any n. While not strictly necessary for the theoretical development of dλ-Amor, this modality is sometimes useful in practice.

Index variables may be quantified over in types with the $\forall i : S.\tau$ and $\exists i : S.\tau$ types, and polymorphic type variables are quantified over using the $\forall \alpha : K.\tau$ type constructor — we do not support existential types, though there is no metatheoretical barrier to their inclusion.

As previously mentioned, the type of lists $L^{I}\,\tau$ is refined by length — the values of this type all have length I. Next, dλ-Amor also includes two "constraint types", $\Phi \implies \tau$, and $\Phi\&\tau$. Values of the first type are known to have type τ when Φ holds, and values of the second type are values of type τ, along with an (irrelevant) proof of Φ. As λ-Amor has no error handling mechanism, this construct is helpful for statically preventing errors by encoding function pre and post-conditions in a type: for instance, the `head` function may be typed as $\forall n : \mathbb{N}.(n \geq 1) \implies (L^{n}\,\tau \multimap \tau)$

Finally, dλ-Amor's types include abstraction and application forms for indexed types. The abstraction form $\lambda i : S.\tau$ has kind $S \to K$ when τ has kind K, and so term variables will never have type $\lambda i : S.\tau$, as it is a higher-kinded type.

5.1.3. *Terms.* While the original presentation of λ-Amor takes great care to include only a barebones term syntax, dλ-Amor will have to expand this syntax somewhat to ensure that the textual representation of a program is unambiguous for programming purposes. Practically, this means that every logical connective has explicit syntactic introduction and elimination forms, whereas this is handled silently in λ-Amor.

$$\boxed{\Theta \vdash \Delta\,\texttt{wf}} \quad \boxed{\Theta; \Delta \vdash I : S} \quad \boxed{\Theta; \Delta \vdash \Phi\ \texttt{wf}} \quad \boxed{\Psi; \Theta; \Delta \vdash \tau : K}$$

$$\boxed{\Psi; \Theta; \Delta \vdash \tau <: \tau' : K} \quad \boxed{\Psi; \Theta; \Delta \vdash \Gamma\,\texttt{wf}} \quad \boxed{\Psi; \Theta; \Delta \vdash \Gamma \sqsubseteq \Gamma'} \quad \boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau}$$

FIGURE 4. Judgment Forms of the dλ-Amor Type System

The term syntax for all of the standard connectives should be familiar. The two products are distinguished by double angled brackets for positive pairs, and parentheses for negative pairs. All binders are un-annotated to reduce the burden on the programmer. Lists are constructed with nil and cons constructors, and the elimination form is a pattern match. The last standard inclusion is a fixpoint operator `fix`, which allows us to write generally-recursive functions.

The refinement type syntax is similar to that of the logical connectives: universal quantifiers are introduced and eliminated with $\Lambda i.e$ abstraction and application $e\,[I]$, while existentials have the traditional `pack`/`unpack`. The two constraint types are less standard: constraint conjunction is introduced with a pair of angle brackets $< e >$ and eliminated with a `clet`, while constraint implication is introduced and eliminated with a "silent" abstraction $\Lambda.e$, and eliminated with an application $e\,\{\}$.

The syntax associated to the amortized analysis constructs is likely less familiar. The monadic cost type $M\,(I, \vec{p})\,\tau$ has three operations associated with it: $\texttt{ret}(e)$ and $\texttt{bind}\,x = e_1\,\texttt{in}\,e_2$, the unit and bind of the monad, respectively, as well as $\texttt{tick}[I|\vec{p}]$, an atomic operation which incurs a cost of $\phi(I, \vec{p})$. The potential type has introduction form $\texttt{store}[I|\vec{p}](e)$ and elimination form $\texttt{release}\,x = e_1, \texttt{in}\,e_2$. Similarly, the *constant* potential type has introduction form $\texttt{store}[I](e)$, and the same elimination syntax as the AARA-style potential type.

5.2. Type System of dλ-Amor. In Figure 4, we provide a listing of the judgments which make up dλ-Amor's type system. Selected rules are presented in Figure 8, and a listing of all rules can be found in Appendix A.

5.2.1. *Contexts.* Judgments in dλ-Amor have as many as five contexts. Contexts Ψ map type variables to their kinds. Θ is an index variable context, which maps index variables to their sorts. Δ is a list of constraints, which are assumptions of the judgment — constraints in Δ may mention variables in Θ, and so there is a weak form of dependence between the two contexts. The final two contexts Ω and Γ are term variable contexts, which map variables to their types.

The context Ω is referred to as the exponential context, and it contains variables which may be used more than once: i.e. are not subject to the affine restriction [16]. Finally, the context Γ lists the rest of the variables, which may be used at most once.

To avoid questions of exchange, we consider all of the contexts except for Δ up to permutations. Indeed, we will frequently treat contexts like sets, testing membership. Further, it will be useful later on to take intersections, unions, and differences of contexts: these operations will only be defined when both operations involved are subsets of a common superset.

5.2.2. *Index Terms and their Sorts.* The rules that make up the sort system for index terms (prefixed I-) are mostly self-explanatory: we ensure that arguments to arithmetic operators have the same sorts. Since all three base sorts ($\mathbb{N}$, $\mathbb{R}^+$, $\vec{\mathbb{R}^+}$) are nonnegative, the rule for subtraction $I - J$ must ensure that $I \geq J$. As discussed in Section 2, the rule I-ConstVec shows how `const` promotes index terms of sort $\mathbb{R}^+$ to sort $\vec{\mathbb{R}^+}$, while the rule I-Shift types the shift operation on potential vectors (Definition 2.3). Finally, the I-Lam and I-App rules give the introduction and elimination rules for the index-level functions.

5.2.3. *Types and their Kinds.* The type formation rules (prefixed K-) for dλ-Amor are very straightforward. All types have kind $\star$, with the exception of the index type abstraction and elimination forms. The rule K-FamLam ensures that an indexed type $\lambda i : S.\tau$ has kind $S \to K$ when τ has kind K, and the indexed-type application $\tau\, I$ has kind K when τ has kind $S \to K$ and I is of sort S, as seen in K-FamApp.

5.2.4. *Subtyping.* The majority of the rules for subtyping in dλ-Amor (prefixed by S-) are the standard congruences for logical connectives. The rules for the types involved in cost analysis for refinements, however, warrant some discussion.

The rule S-Monad gives the subtyping relation for the cost monad: $\mathbb{M}\,(I, \vec{q})\,\tau_1 <: \mathbb{M}(J, \vec{p})\,\tau_2$ when $\tau_1 <: \tau_2$, $I = J$, and $\vec{q} \leq \vec{p}$ componentwise. The soundness of this rule relies on the fact that $\phi(n, \vec{q}) \leq \phi(n, \vec{p})$ when $\vec{q} \leq \vec{p}$, and the fact that the cost annotations represent *upper bounds* — it is safe to use a computation which incurs less cost in a context which expects one that

[16] One may think of all types in Ω implicitly beginning with !, and imagine the variable rule for exponential variables to be silently inserting the counit $!\tau \multimap \tau$. This dual-context construction is standard in the study of modal types. [39]

I-Var
$$\frac{i : S \in \Theta}{\Theta; \Delta \vdash i : S}$$

I-Plus
$$\frac{\Theta; \Delta \vdash I : bS \qquad \Theta; \Delta \vdash J : bS}{\Theta; \Delta \vdash I + J : bS}$$

I-Minus
$$\frac{\Theta; \Delta \vdash I : bS \qquad \Theta; \Delta \vdash J : bS \qquad \Theta; \Delta \vDash I \geq J}{\Theta; \Delta \vdash I - J : bS}$$

I-Shift
$$\frac{\Theta; \Delta \vdash I : \vec{\mathbb{R}^+}}{\Theta; \Delta \vdash \triangleleft I : \vec{\mathbb{R}^+}}$$

I-ConstVec
$$\frac{\Theta; \Delta \vdash I : \mathbb{R}^+}{\Theta; \Delta \vdash \texttt{const}(I) : \vec{\mathbb{R}^+}}$$

I-Lam
$$\frac{\Theta, i : bS; \Delta \vdash I : S}{\Theta; \Delta \vdash \lambda i : bS.I : bS \to S}$$

I-App
$$\frac{\Theta; \Delta \vdash I : bS \to S \qquad \Theta; \Delta \vdash J : bS}{\Theta; \Delta \vdash I\ J : S}$$

I-Sum
$$\frac{\Theta; \Delta \vdash I_0 : \mathbb{N} \qquad \Theta; \Delta \vdash I_1 : \mathbb{N} \qquad \Theta, i : \mathbb{N}; \Delta, I_0 \leq i < I_1 \vdash J : bS}{\Theta; \Delta \vdash \sum_{i=I_0}^{I_1} J : bS}$$

K-Var
$$\frac{\alpha : K \in \Psi}{\Psi; \Theta; \Delta \vdash \alpha : K}$$

K-Unit
$$\frac{}{\Psi; \Theta; \Delta \vdash 1 : \star}$$

K-Monad
$$\frac{\Theta; \Delta \vdash I : \mathbb{N} \qquad \Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \qquad \Psi; \Theta; \Delta \vdash \tau : \star}{\Psi; \Theta; \Delta \vdash \mathbb{M}(I, \vec{p})\tau : \star}$$

K-Pot
$$\frac{\Theta; \Delta \vdash I : \mathbb{N} \qquad \Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \qquad \Psi; \Theta; \Delta \vdash \tau : \star}{\Psi; \Theta; \Delta \vdash [I|\vec{p}]\tau : \star}$$

K-FamLam
$$\frac{\Psi; \Theta, i : S; \Delta \vdash \tau : K}{\Psi; \Theta; \Delta \vdash \lambda i : S.\tau : S \to K}$$

K-FamApp
$$\frac{\Psi; \Theta; \Delta \vdash \tau : S \to K \qquad \Theta; \Delta \vdash I : S}{\Psi; \Theta; \Delta \vdash \tau\ I : K}$$

C-Conj
$$\frac{\Theta; \Delta \vdash \Phi_1\ \texttt{wf} \qquad \Theta; \Delta \vdash \Phi_2\ \texttt{wf}}{\Theta; \Delta \vdash \Phi_1 \wedge \Phi_2\ \texttt{wf}}$$

C-Eq
$$\frac{\Theta; \Delta \vdash I : bS \qquad \Theta; \Delta \vdash J : bS}{\Theta; \Delta \vdash I = J\ \texttt{wf}}$$

Figure 5. Selected Sort, Kind, and Constraint Rules

$$\textsc{S-Refl}\quad \frac{}{\Psi;\Theta;\Delta\vdash\tau<:\tau:K}\qquad\qquad \textsc{S-Trans}\quad \frac{\Psi;\Theta;\Delta\vdash\tau_1<:\tau_2:K\qquad \Psi;\Theta;\Delta\vdash\tau_2<:\tau_3:K}{\Psi;\Theta;\Delta\vdash\tau_1<:\tau_3:K}$$

$$\textsc{S-Monad}\quad \frac{\Psi;\Theta;\Delta\vdash\tau_1<:\tau_2:\star\qquad \Theta;\Delta\vDash I=J\qquad \Theta;\Delta\vDash\vec{q}\leq\vec{p}}{\Psi;\Theta;\Delta\vdash\mathbb{M}(I,\vec{q})\tau_1<:\mathbb{M}(J,\vec{p})\tau_2:\star}$$

$$\textsc{S-Pot}\quad \frac{\Psi;\Theta;\Delta\vdash\tau_1<:\tau_2:\star\qquad \Theta;\Delta\vDash I=J\qquad \Theta;\Delta\vDash\vec{p}\leq\vec{q}}{\Psi;\Theta;\Delta\vdash[I|\vec{q}]\tau_1<:[J|\vec{p}]\tau_2:\star}$$

$$\textsc{S-FamLam}\quad \frac{\Psi;\Theta,i:S;\Delta\vdash\tau_1<:\tau_2:K}{\Psi;\Theta;\Delta\vdash\lambda i:S.\tau_1<:\lambda i:S.\tau_2:S\to K}\qquad\qquad \textsc{S-FamApp}\quad \frac{\Psi;\Theta;\Delta\vdash\tau_1<:\tau_2:S\to K\qquad \Theta;\Delta\vDash I=J}{\Psi;\Theta;\Delta\vdash\tau_1\ I<:\tau_2\ J:K}$$

$$\textsc{S-Fam-Beta1}\quad \frac{}{\Psi;\Theta;\Delta\vdash(\lambda i:S.\tau)\ J<:\tau[J/i]:K}\qquad\qquad \textsc{S-Fam-Beta2}\quad \frac{}{\Psi;\Theta;\Delta\vdash\tau[J/i]<:(\lambda i:S.\tau)\ J:K}$$

FIGURE 6. Selected Subtyping rules

incurs more. Dually, it is always safe to throw away potential in the subtyping rules for the two potential modalities, S-Pot and S-ConstPot.

In addition to the subtyping rules at base kind, the rules S-FamLam and S-FamApp govern the subtyping of indexed types. The rule S-FamLam states that subtyping at kind $S\to K$ is simply generated by pointwise subtyping in the codomain K, while S-FamApp is a standard congruence rule. Note that S-FamApp requires that the two arguments be equal: since we do not require that indexed types be monotone, this is the strongest possible form of the rule. Finally, the rules S-FamBeta-1 and S-FamBeta-2 serve to include β-equality of type families in the subtyping relation.

5.2.5. *Context Well-Formedness Judgments and Context Subsumption.* The judgment $\Theta;\Delta\vdash\Phi\ \texttt{wf}$ ensures that Φ is a well-formed context: all index terms mentioned in it are sort-correct, and relations are only judged between index terms of the same sort.

$$\textsc{WF-CCtxE}\quad \frac{}{\Theta \vdash \cdot\ \texttt{wf}} \qquad \textsc{WF-CCtxNe}\quad \frac{\Theta \vdash \Delta\ \texttt{wf} \qquad \Theta; \Delta \vdash \Phi\ \texttt{wf}}{\Theta \vdash \Delta, \Phi\ \texttt{wf}}$$

$$\textsc{WF-TCtxE}\quad \frac{}{\Psi; \Theta; \Delta \vdash \cdot\ \texttt{wf}} \qquad \textsc{WF-TCtxNE}\quad \frac{\Psi; \Theta; \Delta \vdash \Gamma\ \texttt{wf} \qquad \Psi; \Theta; \Delta \vdash \tau : \star}{\Psi; \Theta; \Delta \vdash \Gamma, x : \tau\ \texttt{wf}}$$

$$\textsc{CS-Emp}\quad \frac{}{\Psi; \Theta; \Delta \vdash \Gamma \sqsubseteq \cdot} \qquad \textsc{CS-Var}\quad \frac{x : \tau' \in \Gamma \qquad \Psi; \Theta; \Delta \vdash \tau' <: \tau \qquad \Psi; \Theta; \Delta \vdash \Gamma \setminus \{x : \tau'\} \sqsubseteq \Gamma'}{\Psi; \Theta; \Delta \vdash \Gamma \sqsubseteq \Gamma', x : \tau}$$

Figure 7. Context Well-formedness rules and Context Subsumption

dλ-Amor also requires two auxiliary context-well-formedness judgments: $\Theta \vdash \Delta\ \texttt{wf}$ and $\Psi; \Theta; \Delta \vdash \Gamma\ \texttt{wf}$. The former ensures that the constraints in the context Δ are well-typed with respect to the context Θ, and the latter ensures that all of the types in Γ have kind $\star$.

Finally, The judgment $\Psi; \Theta; \Delta \vdash \Gamma' \sqsubseteq \Gamma$ determines when we may relax a context Γ' to a weaker one Γ with the T-Weaken rule. Intuitively, this judgment encodes the permission to weaken a context as a kind of record subtyping.

5.2.6. *Terms and their Types.* The typing rules for all of the logical connectives have the standard caveats for an affine type system: affine arrow introduction T-ArrI binds variable $x : A$ in the affine context Γ. Multi-premise rules like tensor introduction (T-TensorI) and sum elimination (T-Case) require splitting the affine context to type the premises. As usual, "parallel" premises such as the two arms of a case may share affine resources, as only one branch will be taken at run-time.

Of greater interest are the rules for the cost analysis and refinement type-related constructs. The return of the cost monad lifts a pure value $e : \tau$ to a monadic computation $\texttt{ret}(e)$ which incurs no cost. So, the rule T-Ret types $\texttt{ret}(e)$ at $\mathbb{M}(I, \vec{0})\,\tau$ for any index term I of sort $\mathbb{N}$, where $\vec{0}$ is the length k vector of 0s. Since $\phi(I, \vec{0}) = 0$ independent of the base I, this rule has the desired effect. Meanwhile, the bind of the cost monad sums the costs of the computation and the continuation. T-Bind operationalizes this by typing $\texttt{bind}\, x = e\, \texttt{in}\, e' \ : \mathbb{M}(I, \vec{p} + \vec{q})\,\tau_2$ when

T-Var-1
$$\frac{x : \tau \in \Gamma}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash x : \tau}$$

T-TensorI
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : \tau_1 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash e_2 : \tau_2}{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \langle\!\langle e_1, e_2 \rangle\!\rangle ; \tau_1 \otimes \tau_2}$$

T-TensorE
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : \tau_1 \otimes \tau_2 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau_1, y : \tau_2 \vdash e' : \tau'}{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{let } \langle\!\langle x, y \rangle\!\rangle = e \texttt{ in } e' : \tau'}$$

T-Nil
$$\frac{}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{nil} : L^0 \tau}$$

T-Cons
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : \tau \qquad \Psi; \Theta; \Omega; \Gamma_2 \vdash e_2 : L^I \tau}{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash e_1 :: e_2 : L^{I+1} \tau}$$

T-Match
$$\frac{\begin{array}{c}\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : L^I \tau \\ \Psi; \Theta; \Delta, I = 0; \Omega; \Gamma_2 \vdash e_1 : \tau' \qquad \Psi; \Theta; \Delta, I \geq 1; \Omega; \Gamma_2, h : \tau, t : L^I \tau \vdash e_2 : \tau'\end{array}}{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{match}(e, e_1, h.t.e_2) : \tau'}$$

T-Ret
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{ret } e : \mathbb{M}\,(I, \vec{0})\, \tau}$$

T-Tick
$$\frac{\Theta; \Delta \vdash I : \mathbb{N} \qquad \Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+}}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{tick}[I|\vec{p}] : \mathbb{M}\,(I, \vec{p})\, 1}$$

T-Bind
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : \mathbb{M}\,(I, \vec{p})\, \tau_1 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau_1 \vdash e_2 : \mathbb{M}\,(I, \vec{q})\, \tau_2}{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{bind } x = e_1 \texttt{ in } e_2 : \mathbb{M}\,(I, \vec{p} + \vec{q})\, \tau_2}$$

T-Release
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : [I|\vec{q}]\tau_1 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau \vdash e_2 : \mathbb{M}\,(I, \vec{p} + \vec{q})\, \tau_2}{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{release } x = e_1 \texttt{ in } e_2 : \mathbb{M}\,(I, \vec{p})\, \tau_2}$$

T-Store
$$\frac{\Theta; \Delta \vdash I : \mathbb{N} \qquad \Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \qquad \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{store}[I|\vec{p}](e) : \mathbb{M}\,(I, \vec{p})\, ([I|\vec{p}]\, \tau)}$$

T-Sub
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau' \qquad \Psi; \Theta; \Delta \vdash \tau' <: \tau}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau}$$

T-Weaken
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \qquad \Theta; \Delta \vdash \Omega' \sqsubseteq \Omega \qquad \Theta; \Delta \vdash \Gamma' \sqsubseteq \Gamma}{\Psi; \Theta; \Delta; \Omega'; \Gamma' \vdash e : \tau}$$

Figure 8. Selected dλ-Amor rules

$e : \mathbb{M}(I, \vec{p})\,\tau_1$ and $x : \tau_1 \vdash e' : \mathbb{M}(I, \vec{q})\,\tau_2$. The soundness of this rule is justified by the linearity of the ϕ function from Theorem 2.1.

The two operations for the potential modality are carefully constructed to work harmoniously with the cost monad. Firstly, given a term $e : \tau$, the rule T-Store allows us to store $\phi(I, \vec{p})$ potential on the term by incurring that amount of cost: this takes the form of assigning the type $\mathbb{M}(I, \vec{p})\ ([I|\vec{p}]\,\tau)$ to the term $\texttt{store}[I|\vec{p}](e)$. Note that to access the underlying potential, one must first `bind` the computation, in effect incurring the requisite $\phi(I, \vec{p})$ cost to have access to the potential. Dually, the rule T-Release gives the typing for using potential. The potential on a term $e : [I|\vec{p}]\,\tau_1$ can be used to pay for a monadic continuation $x : \tau_1 \vdash e' : \mathbb{M}(I, \vec{q} + \vec{p})\,\tau_2$ to get $\texttt{release}\, x = e \,\texttt{in}\, e' : \mathbb{M}(I, \vec{q})\,\tau_2$. The rules for constant potentials follow a similar pattern: the constant store expression $\texttt{store}[J](e)$ has type $\mathbb{M}(I, \texttt{const}(J))\ ([J]\,\tau)$ by T-StoreConst. We note that the type system enforces a discipline that all potential-related activities happen inside the cost monad, which greatly simplifies the type soundness proof found in Rajani et al. [64], when compared to resource-aware type systems with pervasive cost.

The list type $(L^I\,\tau)$ is length-indexed, and so its typing rules are somewhat more involved than the standard ones. To enforce the length refinement, the rules T-Nil and T-Cons specify that the empty list `[]` has type $L^0\tau$, while a cons list $e :: e'$ has type $L^{I+1}\tau$ for $e : \tau$ and $e' :: L^I\tau$. The list elimination rule T-Match is more or less standard, but the two branches are typed under extra constraints in the constraint context Δ. If the scrutinee has type $L^I\tau$, then the nil case of the match is typed under the assumption that $I = 0$. Meanwhile the cons case is given the assumption $I \geq 1$, and the tail of the list is bound as having type $L^{I-1}\tau$.

In addition to the length-refined lists, the refinement type portion of dλ-Amor's type system also includes index term quantifiers in types $(\forall, \exists)$, as well as the two constraint types $(\Phi\&\cdot$, $\Phi \implies \cdot)$. The treatment of the quantifiers is standard: the rules T-ILam and T-ExistE bind index variables in the index context Θ, while the rules T-IApp and T-ExistI substitute in index terms provided by the syntax. The rules for the constraint types operate in a similarly dual fashion.

Finally, dλ-Amor explicitly includes two non-logical rules. The first is a subtyping rule T-Sub, which may be used to downcast the type of a term to a less precise one. This rule has no corresponding syntactic form, and thus may be inserted anywhere in a derivation. The second is T-Weaken, which allows for the weakening of the two term variable contexts, Ω and Γ. As previously mentioned, the weakening relation on which this rule depends includes subtyping, and so a weaker context may include less precise types, and not just fewer available variables.

5.2.7. *Presuppositions.* All of the judgments presented so far are “raw” judgments — one may mechanically derive a proof of one using the inference rules, without regard for whether or not the judgment makes any sense. Traditionally, the requisite assumptions for stating a judgment in a sensical manner are known as *presuppositions.* For example, the sort-checking judgment $\Theta; \Delta \vdash I : S$ requires that the constraint context Δ be well-formed with respect to Θ. There are many ways of handling these, but in this work we choose to make them explicit. Each raw judgment form has an associated judgment form which packages together the requisite well-formedness presuppositions for that judgment. We denote this by a subscript p on the turnstile.

Definition 5.1. *We say that* $\Theta; \Delta \vdash_p I : S$ *when* $\Theta \vdash \Delta$ *wf and* $\Theta; \Delta \vdash I : S$.

Definition 5.2. *We write* $\Psi; \Theta; \Delta \vdash_p \tau : K$ *to mean that* $\Theta \vdash \Delta$ *wf and* $\Psi; \Theta; \Delta \vdash \tau : K$

Definition 5.3. *We write* $\Psi; \Theta; \Delta \vdash_p \tau <: \tau' : K$ *to mean that*

(1) $\Theta \vdash \Delta$ *wf*
(2) $\Psi; \Theta; \Delta \vdash \tau : K$
(3) $\Psi; \Theta; \Delta \vdash \tau' : K$
(4) $\Psi; \Theta; \Delta \vdash \tau <: \tau' : K$

Definition 5.4. *We say* $\Psi; \Theta; \Delta \vdash_p \Gamma \sqsubseteq \Gamma'$ *to mean that*

(1) $\Theta \vdash \Delta$ *wf*
(2) $\Psi; \Theta; \Delta \vdash \Gamma$ *wf*
(3) $\Psi; \Theta; \Delta \vdash \Gamma'$ *wf*
(4) $\Psi; \Theta; \Delta \vdash \Gamma \sqsubseteq \Gamma'$

Definition 5.5. *We say that* $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e : \tau$ *when*

(1) $\Theta \vdash \Delta$ *wf*
(2) $\Psi; \Theta; \Delta \vdash \Omega$ *wf*
(3) $\Psi; \Theta; \Delta \vdash \Gamma$ *wf*

(4) $\Psi; \Theta; \Delta \vdash \tau : \star$

(5) $\Psi; \Theta; \Delta; \Omega \vdash e : \tau$

5.3. Proof Theory of dλ-Amor. Any type system as complex as dλ-Amor's has heaps of syntactic structure to exploit. This structure comes, as is traditional for type theories and type systems, in the form of admissible rules. Each of dλ-Amor's five principal judgments have their own set of admissible rules, corresponding to the structural rules for each context, as well as other judgmental structure. Because of this, the proof theory of dλ-Amor is incredibly rich. We will make no attempt to cover it all here, and instead pick and choose the theorems which will be useful down the line when it comes time to prove the soundness and completeness of biλ-Amor. Below, we present the admissible rules which are used repeatedly in further metatheoretic developments: proofs and intermediate theorems can be found in Appendix A.

First, we prove two statements about the context well-formedness judgments that should be intuitively clear: since term variable contexts are not dependent, subsets of well-formed contexts are themselves well-formed.

THEOREM 5.1. *If* $\Psi; \Theta; \Delta \vdash \Gamma$ *wf and* $\Gamma' \subseteq \Gamma$ *then* $\Psi; \Theta; \Delta \vdash \Gamma'$ *wf*

THEOREM 5.2. *If* $\Theta \vdash \Delta, \Phi$ *wf then* $\Theta \vdash \Delta$ *wf*

Since the type and index variable contexts are fully structural, every judgment that uses them admits weakening and contraction. In practice, we will only ever need to explicitly use the weakening theorem for context subsumption.

THEOREM 5.3. *If* $\Psi; \Theta; \Delta \vdash_p \Gamma \sqsubseteq \Gamma'$ *and* $\Psi' \supseteq \Psi$*,* $\Theta' \supseteq \Theta$*, and* $\Delta' \supseteq \Delta$*, then* $\Psi'; \Theta'; \Delta' \vdash_p \Gamma \sqsubseteq \Gamma'$

Substitution is admissible for all contexts in all judgments — this is essentially a requirement of a type system! However, we will primarily be concerned with substitution for types, as it will become critical once we discuss normalization for types in Section 7.1.

THEOREM 5.4. *If* $\Psi; \Theta, i : S; \Delta \vdash_p \tau : K$ *and* $\Theta; \Delta \vdash_p I : S$ *then* $\Psi; \Theta; \Delta \vdash_p \tau[I/i] : K$

The strengthening theorem below only becomes relevant when passing back and forth between dλ-Amor and our eventual algorithmic system. Intuitively, the concept is clear: if a subtyping relation holds between two types which share a common set of free variables, the subtyping can be derived while mentioning only those variables.

THEOREM 5.5. *Suppose* $\Psi;\Theta;\Delta \vdash \tau : K$ *and* $\Psi;\Theta;\Delta \vdash \tau' : K$*. If* $\Psi';\Theta';\Delta \vdash_p \tau <: \tau' : K$ *with* $\Theta' \supseteq \Theta$ *and* $\Psi' \supseteq \Psi$*, then* $\Psi;\Theta;\Delta \vdash_p \tau <: \tau' : K$

The next theorem is a convenient equivalent characterization of the context subsumption judgment, which formalizes our intuition of this judgment as being essentially record subtyping: the inclusion of Γ' into Γ acts as record width subtyping, while pointwise subtyping on the intersection of the two contexts allows for record depth subtyping.

THEOREM 5.6. $\Psi;\Theta;\Delta \vdash \Gamma \sqsubseteq \Gamma'$ *if and only if for all* $x : \tau' \in \Gamma'$*, there is some* τ *so that* $x : \tau \in \Gamma$ *and* $\Psi;\Theta;\Delta \vdash \tau <: \tau' : \star$.

Finally, we prove a compatability lemma about context subsumption that is similar in flavor to Theorem 5.5, which allows us to strengthen the assumptions of the judgment in a specific (but common) situation.

THEOREM 5.7. *Suppose that*

(1) $\Psi;\Theta;\Delta \vdash_p \Gamma_1 \sqsubseteq \Gamma_1'$

(2) $\Psi';\Theta';\Delta' \vdash_p \Gamma_2 \sqsubseteq \Gamma_2'$

(3) $\Gamma_1 \supseteq \Gamma_2$ *and* $\Gamma_1' \supseteq \Gamma_2'$

Then, $\Psi;\Theta;\Delta \vdash_p \Gamma_2 \sqsubseteq \Gamma_2'$.

6. Semantics and Soundness of dλ-Amor

For dλ-Amor to be useful, its type system must be *sound.* In this context, soundness means that the static amortized execution costs from the types given to programs are in fact actual upper bounds on the programs' real dynamic execution cost. To prove that dλ-Amor's type system is sound in this way, we will appeal to a version of the soundness proof of λ-Amor. As discussed in Section 2, λ-Amor differs from dλ-Amor mainly in its treatment of potentials and costs. In fact the two languages are sufficiently similar (by design, of course) that there is a straightforward embedding of dλ-Amor into λ-Amor. This embedding is cost-preserving, and so the soundness of dλ-Amor follows immediately from the soundness of λ-Amor. Formally, we do not present a true embedding into λ-Amor, as it does not have a sort of potential vectors. However, potential vectors can be trivially added to λ-Amor: the kripke logical relation which forms the basis for its soundness proof never inspects index terms, and conflates index terms with the semantic objects they denote. For this reason, we freely consider λ-Amor as having a sort of potential vectors in the style of dλ-Amor, as well as a sort-level uninterpreted function $\phi : \mathbb{N} \times \vec{\mathbb{R}^{+}} \to \mathbb{R}$ which is additive and monotonic in the second argument.

In Section 6.1, we present the operational semantics for dλ-Amor upon which the soundness theorem is based. This semantics is a big-step cost-indexed operational semantics: the cost indices are the concrete notion of cost that will be bounded by the statically-predicted costs in the soundness theorem.

Then, in Section 6.2, we sketch the embedding of dλ-Amor into λ-Amor, and further sketch proofs that the cost semantics of dλ-Amor coincides with that of λ-Amor under the embedding, as well as the overall soundness theorem of dλ-Amor. We will not present the full details of the translation here. However, the strategy is clear, and we see no barriers to its formalization.

6.1. Operational Semantics of dλ-Amor.

To pin down the exact cost of programs written in dλ-Amor, we provide a *cost semantics* for the language: a big-step operational semantics which is indexed by the cost of evaluation.

Operationally, dλ-Amor behaves like a call-by-name monadic version of PCF. The cost semantics, for which selected rules are presented in Figure 9, consists of two separate judgments. First is a *pure* evaluation relation: $e \Downarrow v$, which evaluates an expression of type τ to a value of the same type. Evaluations in this relation are not thought to incur any cost. The set of values includes all of the monadic actions, which must be subsequently *forced* to be evaluated. This is accomplished with the forcing evaluation relation $e \Downarrow^{\kappa} v'$, which relates monadic values of type $\mathbb{M}\, I\, \tau$ to values of type τ. The pure evaluation judgment is an entirely standard big-step semantics, but selected rules from the forcing judgment can be found in Figure 9.

The rules for the pure evaluation relation are straightforward- as all monadic terms are values, the pure relation simply behaves like a big-step evaluation relation for by-name PCF. The rules for the refinement syntax at term level behave as if the syntax for refinements has been erased at runtime- they contribute nothing meaningful to the operational semantics.

$$\frac{e \Downarrow v}{\texttt{ret}(e) \Downarrow^0 v} \qquad \frac{}{\texttt{tick}[I|\vec{p}] \Downarrow^{\phi(I,\vec{p})} ()} \qquad \frac{e \Downarrow v}{\texttt{store}[I|\vec{p}](e) \Downarrow^{\phi(I,\vec{p})} v}$$

$$\frac{e_1 \downarrow v_1 \qquad v_1 \Downarrow^{\kappa_1} v_1' \qquad e_2[v_1'/x] \Downarrow v_2 \qquad v_2 \Downarrow^{\kappa_2} v}{\texttt{bind}\, x = e_1 \,\texttt{in}\, e_2 \Downarrow^{\kappa_1+\kappa_2} v}$$

$$\frac{e \Downarrow v_1 \qquad e_2[v_1/x] \Downarrow v_2 \qquad v_2 \Downarrow^{\kappa} v_3}{\texttt{release}\, x = e \,\texttt{in}\, e' \Downarrow^{\kappa} v_3}$$

FIGURE 9. Selected Rules of dλ-Amor's Cost Semantics

The rules for the forcing relation warrant some discussion. Since all monadic computations are values, the forcing relation depends on the pure relation to evaluate sub-expressions. For instance, the forcing relation evaluates `ret`(e) to v in 0 steps when $e \Downarrow v$. The pure relation will take some steps of computation by performing β-redexes, but we will not consider these to be *costly* — only `tick`s incur any cost. Pure evaluations steps thus do not need to be accounted for in the forcing relation.

Most importantly, the `tick`$[I|\vec{p}]$ term evaluates with cost $\phi(I,\vec{p})$ to the trivial value $()$. This rule encodes the heretofore intutitive cost behavior of the type $\mathbb{M}(I,\vec{p})\,\tau$, by explicitly assigning the atomic costly operation the cost $\phi(I,\vec{p})$ in our cost semantics. The final cost-monadic term, the `bind`, is assigned cost in a purely compositional way. The evaluation of `bind` proceeds like the evaluation of a let-binding, where the costs of forcing the argument and then the subsequent continuation are added, and given as the total cost.

Finally, the two potential-related operations incur no semantic cost. This may come as a surprise — the statically predicted cost for the `store` operation (for example) is the amount of potential to be allocated. However, this cost is entirely for bookeeping purposes to ensure that potentially is used soundly: it is not truly incurred when the program runs. Similarly, the `release` operation runs identically to `bind`: it is simply a monadic sequencing. This "ghost" nature of potential at runtime is congruent with the way we think about amortized analysis. Recalling the notation of Chapter 1, the operational semantics give the costs $C(f)$, while the static types encode the amortized cost $A(f) + \Delta\Phi$.

6.2. Embedding of dλ-Amor in λ-Amor. The translation of dλ-Amor into λ-Amor requires little insight: we simply compile the costs and potentials written abstractly as a base and potential vector to the ϕ function applied to a pair. Concretely, the meat of the translation on types consists of two rules: the dλ-Amor cost type $\mathbb{M}(I,\vec{p})\,\tau$ is translated to the λ-Amor $\mathbb{M}\,(\phi(I,\vec{p}))\;\tau'$, and the potential type $[I|\vec{p}]\,\tau$ is translated to $[\phi(I,\vec{p})]\;\tau'$, where τ' is the translation of τ. These translations respect rules of the two type systems: the monotonicity and additivity of the ϕ function from Theorem 2.1 justify the translations of the `bind` and `release` operations, as well as the subtyping rule for costs and potentials. The rest of the translation is primarily an erasure. λ-Amor's syntax includes no explicit index terms or types at the term level, and so these are all erased. Finally, the shift operation is erased, a move which is justified by Theorem 2.2.

For the remainder of the section, we will write the embedding on all syntactic forms as $(\cdot)^\circ$. Further, the typing judgments of λ-Amor will be distinguished from those of dλ-Amor by using a $\Vdash$ for their turnstile, while the evaluation relation of λ-Amor will be written with a single arrow $\downarrow$.

THEOREM 6.1. *If* $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e : \tau$ *then* $\Psi^\circ;\Theta^\circ;\Delta^\circ;\Omega^\circ;\Gamma^\circ \Vdash e^\circ : \tau^\circ$

6.2.1. *Statement of Soundness of dλ-Amor.* To prove the soundness of dλ-Amor, we begin by noting that its operational semantics are preserved under the erasure to λ-Amor. This is to be expected: dλ-Amor's cost semantics is simply that of λ-Amor, only written in terms of the abstract costs $\phi(I,\vec{p})$.

THEOREM 6.2. *If* $e \Downarrow^\kappa v$*, then* $e^\circ \downarrow^\kappa v^\circ$

From this, the soundness theorem for dλ-Amor follows immediately: the actual cost of running a closed monadic computation is bounded above by its statically-predicted amortized cost.

THEOREM 6.3. *If* $\cdot \vdash_p e : \mathbb{M}\,(I,\vec{p})\,\tau$ *and* $e \Downarrow^\kappa v$*, then* $\kappa \leq \phi(I,\vec{p})$.

PROOF. By Theorem 6.1 and Theorem 6.2, we have that $\cdot \Vdash e^\circ : \mathbb{M}\,\phi(I,\vec{p})\,\tau^\circ$, and $e^\circ \downarrow^\kappa v^\circ$. Then, by Theorem 1 of Rajani et al. [64], we have that $\kappa \leq \phi(I,\vec{p})$, as required. □

$\boxed{\Theta \vdash \Delta\,\texttt{wf} \Rightarrow \Phi}$ $\boxed{\Theta; \Delta \vdash I : S \Rightarrow \Phi}$ $\boxed{\Theta; \Delta \vdash \Phi\,\texttt{wf} \Rightarrow \Phi'}$ $\boxed{\Psi; \Theta; \Delta \vdash \tau : K \Rightarrow \Phi}$

$\boxed{\Psi; \Theta; \Delta \vdash \tau <: \tau' : K \Rightarrow \Phi}$ $\boxed{\Psi; \Theta; \Delta \vdash \Gamma\,\texttt{wf} \Rightarrow \Phi}$ $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \Rightarrow \Phi, \Gamma'}$

FIGURE 10. Judgment Forms of the biλ-Amor Type System

7. Algorithmic Type System of biλ-Amor

With the formalism of dλ-Amor under control, we move on to presenting the formalism for biλ-Amor. In short, this entails applying all of the type system algorithmization techniques from Section 4 to dλ-Amor at once. As one might expect, this yields a fair bit of complexity in the type system, and so we spend a good deal of space in this section exploring the rules of biλ-Amor in depth, and commenting on some non-obvious design decisions along the way. Additionally, we will formalize the type normalization procedure that is built into biλ-Amor's two-phase subtyping.

7.0.1. *Syntax.* The syntax of biλ-Amor is nearly identical to that of dλ-Amor: this is by design, as biλ-Amor is intended to be an implementable version of dλ-Amor. The only difference is the addition of the type annotation syntax $(e : \tau)$ described in Section 4.0.1. The main change between the two type systems is in the forms of the judgments. Some judgments change in only minor ways: the sort-checking, kind-checking, and constraint well-formedness judgments are all the same as in dλ-Amor, with the exception of the added constraint outputs as described in Section 4.0.3. The subtyping judgment also sports a constraint output, but is also is split into two, with first a "normal form subtyping" relation which judges one type to be a subtype of another when both are in normal form, and then the general algorithmic subtyping relation which relates two types by normalizing them and then relating them via the normal form subtyping relation. Finally, the typing judgment changes the most: it splits into a checking ($\downarrow$) and inferring/synthesis ($\uparrow$) judgment to support bidirectional type inference, with added constraint outputs for solving and unused variable context output for the I/O method. These judgment forms are all shown in Figure 10.

7.0.2. *Sorts, Kinds, and Well-Formed Constraints.* As we will see is true for the majority of the judgments of biλ-Amor, the majority of the rules from dλ-Amor carry over with only minor modification. Although they do form the typing rules for a (small) language embedded

$$\textsc{AI-Var}\quad \frac{i : S \in \Theta}{\Theta; \Delta \vdash i : S \Rightarrow \top} \qquad \textsc{AI-Plus}\quad \frac{\Theta; \Delta \vdash I : bS \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash J : bS \Rightarrow \Phi_2}{\Theta; \Delta \vdash I + J : bS \Rightarrow \Phi_1 \wedge \Phi_2}$$

$$\textsc{AI-Minus}\quad \frac{\Theta; \Delta \vdash I : bS \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash J : bS \Rightarrow \Phi_2}{\Theta; \Delta \vdash I - J : bS \Rightarrow \Phi_1 \wedge \Phi_2 \wedge (I \geq J)}$$

$$\textsc{AI-Sum}\quad \frac{\Theta; \Delta \vdash I_0 : \mathbb{N} \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash I_1 : \mathbb{N} \Rightarrow \Phi_2 \qquad \Theta, i : \mathbb{N}; \Delta, I_0 \leq i < I_1 \vdash J : bS \Rightarrow \Phi_3}{\Theta; \Delta \vdash \sum_{i=I_0}^{I_1} J : bS \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \forall i : \mathbb{N}.(I_0 \leq i < I_1 \to \Phi_3)}$$

$$\textsc{AC-Top}\quad \frac{}{\Theta; \Delta \vdash \top\ \texttt{wf} \Rightarrow \top} \qquad \textsc{AC-Bot}\quad \frac{}{\Theta; \Delta \vdash \bot\ \texttt{wf} \Rightarrow \top} \qquad \textsc{AC-Conj}\quad \frac{\Theta; \Delta \vdash \Phi_1\ \texttt{wf} \Rightarrow \Phi_1' \qquad \Theta; \Delta \vdash \Phi_2\ \texttt{wf} \Rightarrow \Phi_2'}{\Theta; \Delta \vdash \Phi_1 \wedge \Phi_2\ \texttt{wf} \Rightarrow \Phi_1' \wedge \Phi_2'}$$

$$\textsc{AC-Forall}\quad \frac{\Theta, i : S; \Delta \vdash \Phi\ \texttt{wf} \Rightarrow \Phi'}{\Theta; \Delta \vdash \forall i : S.\Phi\ \texttt{wf} \Rightarrow \forall i : S.\Phi'} \qquad \textsc{AC-Eq}\quad \frac{\Theta; \Delta \vdash I : bS \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash J : bS \Rightarrow \Phi_2}{\Theta; \Delta \vdash I = J\ \texttt{wf} \Rightarrow \Phi_1 \wedge \Phi_2}$$

$$\textsc{AK-Unit}\quad \frac{}{\Psi; \Theta; \Delta \vdash 1 : \star \Rightarrow \top} \qquad \textsc{AK-Tensor}\quad \frac{\Psi; \Theta; \Delta \vdash \tau_1 : \star \Rightarrow \Phi_1 \qquad \Psi; \Theta; \Delta \vdash \tau_2 : \star \Rightarrow \Phi_2}{\Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

$$\textsc{AK-IForall}\quad \frac{\Psi; \Theta, i : S; \Delta \vdash \tau : \star \Rightarrow \Phi}{\Psi; \Theta; \Delta \vdash \forall i : S.\tau : \star \Rightarrow \forall i : S.\Phi}$$

$$\textsc{AK-Monad}\quad \frac{\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \Rightarrow \Phi_2 \qquad \Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_3}{\Psi; \Theta; \Delta \vdash \mathbb{M}(I, \vec{p})\tau : \star \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3}$$

FIGURE 11. Selected Algorithmic Sort, Kind, and Constraint Rules

in biλ-Amor, the sort-assignment, kind-assignment and well-formedness judgments for index terms, types, and constraints respectively, do not require a bidirectional treatment. This is because all binders in these three syntactic categories are fully annotated, and so we can easily implement sort/kind inference and checking without any difficulty. Similarly, since the index and type variable contexts are fully structural, there is no need for the I/O method. Hence, the only modification to these three judgments is the addition of the constraint output.

Intuitively, the three judgments all have very simple meanings: for instance, $\Theta; \Delta \vdash I : S \Rightarrow \Phi$ is intended to mean that when Φ is valid, $\Theta; \Delta \vdash I$ holds declaratively, and similarly for the other two judgment forms. This intuition is made formal by the soundness proofs in Section 8.

We present a few selected rules from these judgments in Figure 11. As mentioned earlier, the vast majority of rules are carried over from dλ-Amor: two good examples are AI-Plus and AC-Conj, which follow an identical structure to their declarative counterparts, and simply conjoin the output contexts from the premises in the conclusion.

Some declarative rules have an instance of the constraint validity relation as a premise: for example, the rule I-Minus requires $\Theta; \Delta \vDash I \geq J$ to judge $\Theta; \Delta \vdash I - J : bS$. In the algorithmic judgments of biλ-Amor, these constraints are conjoined onto the output constraint of the conclusion. The algorithmic rule corresponding to I-Minus, AI-Minus, exemplifies this pattern. It has two premises to check that the two subterms I and J are of the proper sort, which emit constraints Φ_1 and Φ_2, respectively. The output constraint is then $\Phi_1 \wedge \Phi_2 \wedge (I \geq J)$. This is constructed in such a way that our eventual soundness theorem will be simple, if the output constraint is valid, then so is $I \geq J$, and we can thus use $\Theta; \Delta \vDash I \geq J$ to construct a declarative proof that $I - J$ is sort-correct.

In a similar manner, in rules where premises bind index variables or assume constraints, the bound variable or constraint must be introduced to the conclusion's output constraint to maintain the well-formedness of output constraints. As an example, consider the rules AK-Forall and AC-Forall. Their premises output constraints Φ which may (and usually do) mention the universal index variable $i : S$, which is bound in the context Θ. Then in the conclusion, this variable is no longer present, and so Φ need not be well-formed. To fix this, we explicitly quantify over the index variable i in the conclusion's output constraint.

$$\text{AWF-CCtxE}\quad \frac{}{\Theta \vdash \cdot\ \texttt{wf} \Rightarrow \top} \qquad \text{AWF-CCtxNe}\quad \frac{\Theta \vdash \Delta\ \texttt{wf} \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash \Phi\ \texttt{wf} \Rightarrow \Phi_2}{\Theta \vdash \Delta, \Phi\ \texttt{wf} \Rightarrow \Phi_1 \wedge (\bigwedge \Delta \to \Phi_2)}$$

$$\text{AWF-TCtxE}\quad \frac{}{\Psi; \Theta; \Delta \vdash \cdot\ \texttt{wf} \Rightarrow \top} \qquad \text{AWF-TCtxNE}\quad \frac{\Psi; \Theta; \Delta \vdash \Gamma\ \texttt{wf} \Rightarrow \Phi_1 \qquad \Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_2}{\Psi; \Theta; \Delta \vdash \Gamma, x : \tau\ \texttt{wf} \Rightarrow \Phi_1 \wedge \Phi_2}$$

FIGURE 12. Algorithmic Context Wellformedness Rules

The assumption context Δ bears a similar requirement, as illustrated by the rule AI-Sum. When the constraint $I_0 \le i \le I_1$ is assumed in a premise which emits a constraint Φ_3, the output constraint is transformed to $I_0 \le i \le I_1 \to \Phi_3$ to preserve the meaning of the judgment.

Finally, it's worth noting a potential confusion about the algorithmic constraint well-formedness judgment, $\Theta; \Delta \vdash \Phi\ \texttt{wf} \Rightarrow \Phi'$. The output constraint Φ' does not encode the truth of Φ. The soundness proof will make this concrete, but knowing that $\Theta; \Delta \vDash \Phi'$ only implies that Φ is well-formed, but it need not be valid.

7.0.3. *Algorithmic Context-Wellformedness.* While these judgments are not a part of the typechecking algorithm we will eventually implement, biλ-Amor includes algorithmic versions of the two context-wellformedness judgments from dλ-Amor to be used in the presuppositions for the rest of the algorithmic judgments. The two judgments, constraint context well-formedness ($\Theta \vdash \Delta\ \texttt{wf} \Rightarrow \Phi$) and term variable context well-formedness ($\Psi; \Theta; \Delta \vdash \Gamma\ \texttt{wf} \Rightarrow \Phi$) have the expected intended meaning: when the output constraint is valid, the declarative version is thought to hold.

These judgments will not need to be implemented since typechecking will begin from empty contexts, where both judgments hold vacuously. Finally, we note that biλ-Amor does not have an algorithmic version of the declarative context subsumption judgment, since it only exists in the declarative theory to be used in the weakening rule, which biλ-Amor does not have. Indeed, weakening of the term context in biλ-Amor is admissible, a fact which will be proven in Section 8.

7.1. Normalization of Types. To circumvent the issue of deciding β-equality of types as a part of biλ-Amor's subtyping routine, we employ a normalization (or evaluation) procedure to eliminate all β-redexes from a type. Once these β-redexes have been eliminated, subtyping only requires congruence rules. The normalization proof that we describe in this section is a normalization relative to the equational theory induced by dλ-Amor's subtyping relation, that is to say: we will eventually prove that a type and its normal form are mutual subtypes of each other *with the subtyping relation of dλ-Amor*.

This normalization procedure computes *normal forms* for types, which should be thought of as canonical representatives of their β-equivalence classes. These normal forms can characterized syntactically: we present a pair of relations $\tau\,\texttt{ne}$ and $\tau\,\texttt{nf}$, which judge a type to be neutral or normal, respectively. Neutral types are those which can be of arrow kind, but will not induce any β-redexes when applied to an argument. Normal types are types which include no β-redexes. The former are required to define the latter: the type $\tau\, I$ is only in normal form when τ is not of the form $\lambda i : S.\tau'$. The rules generating these two relations can be found in Appendix A.

Before we present the normalization function, let us a moment to consider why the solution we are about to present is so simple. Proofs of normalization for most calculi require fairly high-powered techniques such as logical relations, categorical semantics, or hereditary substitution. The inherent complexity of normalization proofs stems from the fact that straightforward induction on terms rarely works, since one would need to induct on substitution instances of lambda terms, which are not subterms of the original term. However, dλ-Amor's type-level lambdas do not range over types, they range over index terms. Because of this, substituting an index term into a type also cannot introduce any new type-level β-redexes, and so any substitution instance of an open type in normal form is also normal.

THEOREM 7.1.

(1) If τ $\texttt{ne}$ then $\tau[I/i]$ $\texttt{ne}$
(2) If τ $\texttt{nf}$ then $\tau[I/i]$ $\texttt{nf}$

Because of these simplifying factors, we can define an evaluation function `eval` defined inductively on the structure of types which computes normal forms. The most important clauses of the definition can be found in Figure 13. For all of the logical connectives, the definition proceeds compositionally — the remaining rules can be found in Appendix A

The most important (and only nontrivial) clause of the definition is the application case. To evaluate $\tau\, I$, we begin by evaluating τ. If its normal form is a lambda, we simply perform the

$$\texttt{eval}(\alpha) = \alpha \qquad\qquad \texttt{eval}(\lambda i : S.\tau) = \lambda i : S.\texttt{eval}(\tau)$$

$$\texttt{eval}(\tau\ I) = \begin{cases} \tau'[I/i] & \texttt{eval}(\tau) = \lambda i : S.\tau' \\ \texttt{eval}(\tau)\ I & \text{otherwise} \end{cases}$$

FIGURE 13. Selected Clauses of the `eval` Function

β-reduction. Note that we do not need to evaluate this substitution instance, as it must already be in normal form by Theorem 7.1, assuming the correctness of the `eval` function. Otherwise, we simply re-apply the index term I.

It is not immediately clear that this function in fact computes what we want! For `eval` function to be a normalization procedure, its image must consist only of types in normal form, and every type must be equivalent to its evaluation. Note that we do not prove the stronger property that equivalence is completely characterized by syntactic equality of normal forms (up to equality of index terms). While almost certainly true, this property requires a bit more work to prove and is not required for the discussion, and so we omit it. Finally, we must also prove that the `eval` function preserves kinds. This proof follows the same inductive structure as the proof of normalization, and so we bundle them together. We present the case for evaluation below, and the remainder of the cases can be found in Appendix A. The Normalization Theorem does depend on a small canonical forms lemma: types of arrow kind in normal form must either be lambdas or neutral.

THEOREM 7.2 (Canonical Forms for $S \to K$). *If $\Psi; \Theta; \Delta \vdash \tau : S \to K$ and τ **nf**, then either:*

*(1) $\tau = \lambda i : S.\tau'$ with τ' **nf***

*(2) τ **ne***

THEOREM 7.3 (Normalization Theorem). *If $\Psi; \Theta; \Delta \vdash_p \tau : K$, then:*

(1) $\Psi; \Theta; \Delta \vdash_p \mathit{eval}(\tau) : K$

(2) $\Psi; \Theta; \Delta \vdash_p \tau \equiv \mathit{eval}(\tau) : K$

*(3) $\mathit{eval}(\tau)$ **nf***

THEOREM 7.4. $\mathit{eval}(\tau[J/i]) = \mathit{eval}(\tau)[J/i]$

AS-Unit

$$\frac{}{\Psi;\Theta;\Delta \vdash 1 <:_{\mathtt{nf}} 1 : \star \Rightarrow \top}$$

AS-Tensor

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_1' : \star \Rightarrow \Phi_1 \qquad \Psi;\Theta;\Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_2' : \star \Rightarrow \Phi_2}{\Psi;\Theta;\Delta \vdash \tau_1 \otimes \tau_2 <:_{\mathtt{nf}} \tau_1' \otimes \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

AS-Monad

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \mathbb{M}(I,\vec{q})\tau_1 <:_{\mathtt{nf}} \mathbb{M}(J,\vec{p})\tau_2 : \star \Rightarrow (I = J) \wedge (\vec{q} \leq \vec{p}) \wedge \Phi}$$

AS-Pot

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash [I|\vec{q}]\tau_1 <:_{\mathtt{nf}} [J|\vec{p}]\tau_2 : \star \Rightarrow (I = J) \wedge (\vec{p} \leq \vec{q}) \wedge \Phi}$$

AS-FamLam

$$\frac{\Psi;\Theta, i : S;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : K \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \lambda i : S.\tau_1 <:_{\mathtt{nf}} \lambda i : S.\tau_2 : S \to K \Rightarrow \forall i : S.\Phi}$$

AS-FamApp

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : S \to K \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \tau_1\ I <:_{\mathtt{nf}} \tau_2\ J : K \Rightarrow (I = J) \wedge \Phi}$$

AS-Normalize

$$\frac{\Psi;\Theta;\Delta \vdash \mathtt{eval}(\tau_1) <:_{\mathtt{nf}} \mathtt{eval}(\tau_2) : K \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : K \Rightarrow \Phi}$$

Figure 14. Selected Algorithmic Subtyping Rules

7.1.1. *Algorithmic Subtyping.* In biλ-Amor, there is not one subtyping judgment, but two. The first, which we will refer to as "normal form" subtyping (denoted $<:_{\mathtt{nf}}$), judges one type to be a subtype of another when both are in normal form. This relation contains all of the congruence rules from dλ-Amor's subtyping relation: these are simply transcriptions of their declarative counterparts. Just like with sort/kind-checking, constraint-validity premises are shuffled to the constraint output of the conclusion, and variables bound in premises are quantified over. Deciding a subtyping relation which only includes congruences reduces to syntax-directed search and solving the emitted constraints. Hence, the subtyping relation, selected rules of which can be found in Figure 14, is certainly algorithmic.

The second judgment (denoted $<:$) is generated by a single rule, AT-Normalize. AT-Normalize encodes the first step of our two-step subtyping algorithm. To show that $\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : K \Rightarrow \Phi$, it suffices (and indeed it is necessary) to first normalize τ_1 and τ_2, and then judge that their normal forms are related by the normal form subtyping judgment, $\Psi; \Theta; \Delta \vdash \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : K \Rightarrow \Phi$.

Two distinct phases and normalization aside, the remaining way that biλ-Amor's subtyping differs from dλ-Amor's is in the removal of two rules. The rules S-Refl and S-Trans from dλ-Amor are not included in our algorithmic subtyping relation, as they are not syntax-directed. In Section 8, we show that reflexivity and transitivity are admissible for types in normal form, and that these results may be lifted to the full relation through evaluation.

7.1.2. *Bidirectional Typing Rules.* As expected, the typing judgments of biλ-Amor change the most. For one, we pass to a bidirectional type system. As discussed in Section 4.0.1, this process is fairly standardized, and so the reader who has seen bidirectional type systems in the past will find no surprises in biλ-Amor. The typing judgment is split in two, yielding a mutually recursive pair of checking and inference judgments. Secondly, typing judgment sports a constraint output in a manner identical to all of the other algorithmic judgments discussed so far. Finally, to handle the affine context Γ in an algorithmic way, we employ the I/O method from Section 4.0.4, adding an output context of unused variables Γ', which are threaded through rules in a state-passing manner.

While all of these algorithmization techniques were described in the abstract in Section 4, understanding how they work in the context of a type system as feature-rich as biλ-Amor is another matter entirely. To this end, we take some time to describe the selected rules presented in Figure 15.

Algorithmizing the declarative typing rules is mostly mechanical for the rules governing logical connectives, but requires some ingenuity for the non-logical ones. For each declarative typing rule, we bidirectionalize each premise along with the conclusion, convert to IO-style contexts, and output constraints. In most cases, this is sufficient to algorithmize a rule. However, for some of the cost-related rules like AT-bind or AT-Release, further "optimization" are required. In these cases, we build certain subtyping relations into the algorithmic typing rules to overcome the fact that subtyping can only be applied at the boundary between checking and inference, and not at will, as in the declarative typing rules of dλ-Amor.

AT-Var-1
$$\frac{x : \tau \in \Gamma}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash x \uparrow \tau \Rightarrow \top, \Gamma \setminus \{x : \tau\}}$$

AT-Lam
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma, x : \tau_1 \vdash e \downarrow \tau_2, \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \lambda x.e \downarrow \tau_1 \multimap \tau_2 \Rightarrow \Phi, \Gamma' \setminus \{x : \tau_1\}}$$

AT-App
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 \uparrow \tau_1 \multimap \tau_2 \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_2 \downarrow \tau_1 \Rightarrow \Phi_2, \Gamma_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1\, e_2 \uparrow \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2}$$

AT-TensorI
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 \downarrow \tau_1 \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_2 \downarrow \tau_2 \Rightarrow \Phi_2, \Gamma_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \langle\!\langle e_1, e_2 \rangle\!\rangle \downarrow \tau_1 \otimes \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2}$$

AT-TensorE
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \tau_1 \otimes \tau_2 \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_1, x : \tau_1, y : \tau_2 \vdash e' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{let}\ \ \langle\!\langle x, y \rangle\!\rangle = e\ \ \texttt{in}\ \ e' \downarrow \tau' \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2 \setminus \{x, y\}}$$

AT-Ret
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{ret}\ \ e \downarrow \mathbb{M}\, \phi(I, \vec{p})\, \tau \Rightarrow \Phi, \Gamma'}$$

AT-Bind
$$\frac{\begin{array}{c}\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 \uparrow \mathbb{M}\,(J, \vec{p})\, \tau_1 \Rightarrow \Phi_1, \Gamma_1 \\ \Psi; \Theta; \Delta; \Omega; \Gamma_1, x : \tau_1 \vdash e_2 \downarrow \mathbb{M}\,(I, \vec{q} - \vec{p})\, \tau_2 \Rightarrow \Phi_2, \Gamma_2 \qquad \Phi = (\vec{q} \geq \vec{p}) \wedge (I = J) \wedge \Phi_1 \wedge \Phi_2\end{array}}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{bind}\ \ x = e_1\ \ \texttt{in}\ \ e_2 \downarrow \mathbb{M}\,(I, \vec{q})\, \tau_2 \Rightarrow \Phi, \Gamma_2 \setminus \{x : \tau_1\}}$$

AT-Release
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 \uparrow [J|\vec{q}]\tau_1 \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_1, x : \tau \vdash e_2 \downarrow \mathbb{M}\,(I, \vec{p} + \vec{q})\, \tau_2 \Rightarrow \Phi_2, \Gamma_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{release}\ \ x = e_1\ \ \texttt{in}\ \ e_2 \downarrow \mathbb{M}\,(I, \vec{p})\, \tau_2 \Rightarrow (I = J \wedge \Phi_1 \wedge \Phi_2), \Gamma_2 \setminus \{x\}}$$

AT-Store
$$\frac{\begin{array}{c}\Theta; \Delta \vdash K : \mathbb{N} \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash \vec{w} : \vec{\mathbb{R}^{+}} \Rightarrow \Phi_2 \\ \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi_3, \Gamma' \qquad \Phi = \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge (\vec{p} \leq \vec{w} \leq \vec{q}) \wedge (I = J = K)\end{array}}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{store}[K|\vec{w}](e) \downarrow \mathbb{M}\, \phi(I, \vec{q})\, ([J|\vec{p}]\, \tau) \Rightarrow \Phi, \Gamma'}$$

AT-Sub
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \tau' \Rightarrow \Phi_1, \Gamma' \qquad \Psi; \Theta; \Delta \vdash \tau' <: \tau : \star \Rightarrow \Phi_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma'}$$

AT-Anno
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash (e : \tau) \uparrow \tau \Rightarrow \Phi, \Gamma'}$$

Figure 15. Selected Algorithmic Typing Rules

We begin with AT-Var-1, which allows us to use variables from the affine context. When $x : \tau \in \Gamma$, the term x infers the type τ. Using this rule in a derivation counts as a use of x, and so x must be removed from the output context, as it is no longer unused.

The pair of rules AT-Lam and AT-App exhibit a common pattern which is common to nearly all negative logical connectives in biλ-Amor. For introduction form, both the conclusion and premise are checking. In the elimination form, the conclusion as well as the principal judgment (the judgment typing the term being eliminated) are inferring, while all other premises check [17]. The AT-Lam rule also illustrates a small oddity of the I/O method when applied to affine types. Since variables *can* be left unsued, it's possible for the λ-bound variable x to end up in the output context Γ' of the premise checking the body of the lambda. For this reason, we must explicitly remove x from the context of unused variables as it falls out of scope, lest it be possible to typecheck terms like $\langle\!\langle \lambda x.(), x \rangle\!\rangle$, where a bound variable escapes its scope. The rule AT-App also illustrates how the "threading" aspect of the I/O method is easily combined with the two kinds of typing judgments. To check $e_1\, e_2$ in context Γ, the type of e_1 is inferred, returning unused variables Γ_1. Then, the type of e_2 is checked *in context* Γ_1. That judgment "returns" Γ_2, which is then used as the output judgment for the checking conclusion.

Dually, the rules AT-TensorI and AT-TensorE are a simple instance of the bidirectional rules for a positive logical connective. The introduction form has checking premises and conclusions, just like the negatives. On the other hand, the elimination form has an inferring principle judgment, but checking conclusion — this is because positive elims all take the form of a (many-armed) let-binding, and the type of the continuation cannot be inferred locally. Because of this let-binding style, most positive elims must remove bound variables from the output context to deal with the same scoping issue as AT-Lam.

The remaining rules for logical connectives following a similar pattern: their bidirectional behavior is predetermined by logical concerns discovered by prior work in the area. Algorithmizing the rules for nonlogical connectives, however, requires quite a bit more work and cleverness.

As a case study, consider the rule T-Bind from dλ-Amor:

[17] The connection between bidirectional type systems and polarization/focusing which makes this pattern so ubiquitous in the rules of biλ-Amor is deep, beautiful, and not fully understood. A wonderful overview of work on the subject, as well as exposition about how to bidirectionalizing your own declarative type systems can be found in a paper by Dunfield and Krishnaswami [21].

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_1 : \mathbb{M}\,(I,\vec{p})\,\tau_1 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_2, x:\tau_1 \vdash e_2 \downarrow \mathbb{M}\,(I,\vec{q})\,\tau_2}{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash \texttt{bind}\ \ x = e_1\ \ \texttt{in}\ \ e_2 \downarrow \mathbb{M}\,(I,\vec{p}+\vec{q})\,\tau_2}$$

This plainly follows the let-binding style of positive elimination forms (despite not being a logical connective), and so the same direction pattern seems like a good choice. This rule has no constraint solving premises, and so the output constraints can be conjoined together. Finally, this term has the form of a let-binding, and so we thread contexts through the premises, removing x in the conclusion. These three choices lead to the following first cut at an algorithmic bind rule:

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \uparrow \mathbb{M}\,(I,\vec{p})\,\tau_1 \Rightarrow \Phi_1,\Gamma_1 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_1, x:\tau_1 \vdash e_2 : \mathbb{M}\,(I,\vec{q})\,\tau_2 \Rightarrow \Phi_2,\Gamma_2}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{bind}\ \ x = e_1\ \ \texttt{in}\ \ e_2 : \mathbb{M}\,(I,\vec{p}+\vec{q})\,\tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2 \setminus \{x\}}$$

Unfortunately, this rule is insufficient for potentially subtle reasons. When we implement biλ-Amor, the checking judgment is implemented as a function (essentially) of type `ctx -> tm -> ty -> unit`, which proceeds by a (very large) case analysis on the term and type arguments. The algorithmic bind rule corresponds to the case where the term is the constructor for bind, and the type is the cost monad. However, we hope to not match further into the type to match the index term $\vec{p}+\vec{q}$, because the second component of the cost need not be syntactically a sum of potential vectors! To fix this, we take a slightly different approach. Instead of typing the conclusion at type $\mathbb{M}\,(I,\vec{p}+\vec{q})\,\tau_2$, we will instead have it check against the type $\mathbb{M}\,(I,\vec{q})\,\tau_2$, so long as the continuation checks against $\mathbb{M}\,(I,\vec{q}-\vec{p})\,\tau_2$ (when $\vec{q} \geq \vec{p}$). Intuitively, this new rule encodes the same logic: the total amortized cost of the composite is the sum of the costs of e_1 and e_2.

A similar situation plays out if we consider the first component of the cost pair. The rule above indicates that the first components in the three monadic types need to be *identical.* Just like requiring that the second component be *literally* a sum, this is far too strong a condition: we only need require that they are provably equal. This leads us to the completed AT-Bind rule, as shown in Figure 15. A nearly identical game is played with the rule for the potential elimination form, AT-Release: we generalize the potentials to have syntactically but provably equal bases.

The introduction rules for monads and potentials also require some tweaking. To illustrate, we consider the declarative store rule, T-Store.

$$\frac{\Theta; \Delta \vdash I : \mathbb{N} \qquad \Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \qquad \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{store}[I|\vec{p}](e) : \mathbb{M}\,(I, \vec{p})\,([I|\vec{p}]\,\tau)}$$

We bidirectionalize this in a straightforward manner, by making both the premise and the conclusion checking. The constraint and context outputs are similarly trivial: they are passed from the output of the premise directly to the conclusion. We are then faced with yet another matching problem: the Is and $\vec{p}$s in the term and type are required to be syntactically equal. It is clear how to generalize the bases: we allow all three to be different, but provably equal. The proper formulation for the coefficient components is less clear, however. Inspiration comes from considering the ranges of sound but imprecise typings for the positions. For $\texttt{store}[K|\vec{w}](e)$ to check against $\mathbb{M}\,(I, \vec{q})\ ([J|\vec{p}])$ when $I = J = K$, it ought to be allowable for $\vec{w}$ to be smaller than $\vec{q}$, and for $\vec{p}$ to be smaller than $\vec{w}$. When we ask for $\phi(I, \vec{q})$ potential, it is sound to overpay, and underdeliver. The final rule, AT-Store, allows just this.

The last two interesting rules to be discussed are AT-Sub and AT-Anno. These rules are not analogues of rules which were present in dλ-Amor. Instead, they are the two bidirectional-specific rules discussed in Section 4.0.1 which allow us to mediate between the checking and inference judgments. When a synactic form whose corresponding rule has a checking conclusion (such as a lambda) is placed in a position where its expected to infer (such as the principal position of application), an annotation must be introduced. However, in the opposite situation, a term whose rule has an inferring conclusion may always be used in a checking position, so long as the type which is inferred is more specific than the one the term is being checked against.

7.1.3. *Well-formedness and Presuppositions.* The judgments of biλ-Amor presented thusfar have all been *raw* judgments, in the same sense that we have presented no well-formedness restrictions. Just like in dλ-Amor, we restrict the positions of each relation by well-formedness presuppositions. Again, these are denoted with a subscript p on the turnstile. Unlike, dλ-Amor, these presuppositions are algorithmic in the sense that they use the corresponding judgments from biλ-Amor to impose restrictions. Recalling that the intended meaning of an algorithmic judgment with constraint output is that the declarative analogue holds when the constraint is valid, all of the conditions in algorithmic presuppositions require their constraints to be solved.

This is a natural restriction: once soundness and completeness have been established, these algorithmic presuppositions are equivalent to their declarative counterparts.

Definition 7.1. *We write* $\Theta; \Delta \vdash_p I : S \Rightarrow \Phi$ *to mean*

(1) $\Theta \vdash \Delta\ \mathit{wf} \Rightarrow \Phi_1$ *and* $\Theta; \cdot \vDash \Phi_1$

(2) $\Theta; \Delta \vdash I : S \Rightarrow \Phi$

Definition 7.2. *We write* $\Psi; \Theta; \Delta \vdash_p \tau : K \Rightarrow \Phi$ *to mean*

(1) $\Theta \vdash \Delta\ \mathit{wf} \Rightarrow \Phi_1$ *and* $\Theta; \cdot \vDash \Phi_1$

(2) $\Psi; \Theta; \Delta \vdash \tau : K \Rightarrow \Phi$

Definition 7.3. *We write* $\Psi; \Theta; \Delta \vdash_p \tau <: \tau' : K \Rightarrow \Phi$ *to mean that*

(1) $\Theta \vdash \Delta\ \mathit{wf} \Rightarrow \Phi_1$ *and* $\Theta; \cdot \vDash \Phi_1$

(2) $\Psi; \Theta; \Delta \vdash \tau : K \Rightarrow \Phi_2$ *and* $\Theta; \Delta \vDash \Phi_2$

(3) $\Psi; \Theta; \Delta \vdash \tau' : K \Rightarrow \Phi_3$ *and* $\Theta; \Delta \vDash \Phi_2$

(4) $\Psi; \Theta; \Delta \vdash \tau <: \tau' : K \Rightarrow \Phi$

Definition 7.4. *We say that* $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e \downarrow \tau \Rightarrow \Phi, \Gamma'$ *when*

(1) $\Theta \vdash \Delta\ \mathit{wf} \Rightarrow \Phi_1$ *and* $\Theta; \cdot \vDash \Phi_1$

(2) $\Psi; \Theta; \Delta \vdash \Omega\ \mathit{wf} \Rightarrow \Phi_2$ *and* $\Theta; \Delta \vDash \Phi_2$

(3) $\Psi; \Theta; \Delta \vdash \Gamma\ \mathit{wf} \Rightarrow \Phi_3$ *and* $\Theta; \Delta \vDash \Phi_3$

(4) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_4$ *and* $\Theta; \Delta \vDash \Phi_4$.

(5) $\Psi; \Theta; \Delta; \Omega \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'$

Definition 7.5. *We say that* $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e \uparrow \tau \Rightarrow \Phi, \Gamma'$ *when*

(1) $\Theta \vdash \Delta\ \mathit{wf} \Rightarrow \Phi_1$ *and* $\Theta; \cdot \vDash \Phi_1$

(2) $\Psi; \Theta; \Delta \vdash \Omega\ \mathit{wf} \Rightarrow \Phi_2$ *and* $\Theta; \Delta \vDash \Phi_2$

(3) $\Psi; \Theta; \Delta \vdash \Gamma\ \mathit{wf} \Rightarrow \Phi_3$ *and* $\Theta; \Delta \vDash \Phi_3$

(4) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_4$ *and* $\Theta; \Delta \vDash \Phi_4$.

(5) $\Psi; \Theta; \Delta; \Omega \vdash e \uparrow \tau \Rightarrow \Phi, \Gamma'$

8. Soundness and Completeness of biλ-Amor with respect to dλ-Amor

With the algorithmic system of biλ-Amor in place, the time has come to prove theorems about it. Ideally, we would like to prove that it behaves exactly the same as dλ-Amor. That way, when we build `LambdaAmor` in Section 9, we will know that (a) every program typechecked by our implementation declarative has the proper type in dλ-Amor, and that (b) every well-typed program in dλ-Amor *can be* checked by our implementation. In this context, these two properties are known as soundness and completeness[18], respectively.

As is usually the case with such things, the soundness proofs are very straightforward. This is because biλ-Amor is far more strict and structured than dλ-Amor, so it is always fairly easy to lift a biλ-Amor proof to a proof in dλ-Amor. This mismatch simultaneously makes completeness quite difficult to prove: compiling a proof of one of the judgments of dλ-Amor down to a structured one in biλ-Amor requires some work in general. For this reason, we will begin with proving the soundness theorems, and subsequently move to proving completeness.

The general shape of the soundness theorems are all the same: for every algorithmic judgment $\mathcal{J} \Rightarrow \Phi$ (and corresponding declarative judgment $\mathcal{J}$) we prove that if there is a derivation of $\mathcal{J} \Rightarrow \Phi$ and Φ is valid, then there is a derivation of $\mathcal{J}$. Individual theorems may vary — the inclusion of bidirectionality and the I/O method complicates the statement of soundness for typing — but this is the main flavor. This pattern justifies the intended use of the algorithmic system: we derive algorithmic judgments using the implementation, which outputs constraints. If the constraints are solvable by a solver, the corresponding declarative judgment holds.

Dually, the completeness theorems have the "opposite" shape: if $\mathcal{J}$ holds, then there is some *valid* Φ such that the corresponding algorithmic judgment $\mathcal{J} \Rightarrow \Phi$ is derivable. Modulo handling the bells and whistles of an individual judgment (such as IO contexts or bidirectionality), all of our completeness statements will have this form. To simplify the wording of our theorems somewhat, we adopt the "Twelf convention": theorem meta-variables which appear only in the

[18] This may seem backwards to the reader already familiar with the terms: we think of the declarative system as giving a "ground truth" semantics of which terms have which types, and the algorithmic system as a proof system in which one may manually derive proofs of well-typedness. From this perspective, soundness and completeness are as described above.

conclusion of a theorem are implicitly existentially quantified. For instance, we will write "If $\mathcal{J}$ then $\mathcal{J} \Rightarrow \Phi$ and $\vDash \Phi$" to mean "If $\mathcal{J}$ then there is some Φ such that $\mathcal{J} \Rightarrow \Phi$ and $\vDash \Phi$".

Intuitively, this theorem structure justifies our intended usage of λ-Amor, as an implementation of the algorithmic judgments must cover all possible uses of the declarative system. When combined with soundness (to translate algorithmic judgments back into declarative ones), our algorithm always succeeds to derive a proof of a declarative judgment, if one exists.

8.0.1. *Soundness of Index Terms, Constraints, Contexts, and Types.* The four most basic algorithmic judgments of biλ-Amor mirror their declarative counterparts rule-for-rule: the only "real" modification is the addition of constraint output. This uniformity means that the soundness proofs are fairly trivial single-pass inductions on derivations. Each of these proofs comes in two parts. First, we prove that the soundness holds as a a statement about "raw" judgments by omitting the presuppositions. These theorems are garbage-in, garbage-out: malformed judgments in biλ-Amor are sent to malformed judgments in dλ-Amor. Afterwards, we prove that the presuppositions are preserved, and so well-formed judgments in biλ-Amor are sent to well-formed judgments in dλ-Amor. This two-step process is only necessary because the presuppositions have a mutually inductive structure: to untangle the knot, we must first prove the raw statements, and then repackage them with the required presuppositions afterwards.

Below, we will only present the versions of the theorems with presuppositions included: the gory details can be found in Appendix A. All of the proofs proceed by elementary inductions on derivations, occasionally using easy properties about constraint validity.

THEOREM 8.1 (Soundness of Index Context Well-Formedness). *If* $\Theta \vdash \Delta \ \mathit{wf} \Rightarrow \Phi$ *and* $\Theta; \cdot \vDash \Phi$*, then* $\Theta \vdash \Delta \ \mathit{wf}$

THEOREM 8.2 (Soundness of Sort Checking). *If* $\Theta; \Delta \vdash_p I : S \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$*, then* $\Theta; \Delta \vdash_p I : S$

THEOREM 8.3 (Soundness of Constraint Well-Formedness). *If* $\Theta; \Delta \vdash_p \Phi \ \mathit{wf} \Rightarrow \Phi'$ *and* $\Theta; \Delta \vDash \Phi'$ *then* $\Theta; \Delta \vdash_p \Phi \ \ \mathit{wf}$

THEOREM 8.4 (Soundness of Kind Checking). *If* $\Psi; \Theta; \Delta \vdash_p \tau : K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$ *then* $\Psi; \Theta; \Delta \vdash_p \tau : K$*.*

8.0.2. *Soundness of Subtyping.* Subtyping provides a significantly more interesting soundness proof than the prior cases: we must justify that biλ-Amor's two-step normalize-then-compare strategy is in fact sound for the declarative type system. The proof proceeds in two parts corresponding to the two judgments. We first prove soundness for the normal form subtyping, and then lift it, using the results about normalization from Section 7.1, to the full algorithmic subtyping relation.

THEOREM 8.5 (Soundness of Subtyping for Normal Forms). *If* $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{nf} \tau_2 : K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$ *then* $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K$

THEOREM 8.6 (Soundness of Subtyping). *If* $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$*, then* $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K$

PROOF. There is only one case: $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K \Rightarrow \Phi$ by way of $\Psi; \Theta; \Delta \vdash \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : K \Rightarrow \Phi$ with $\Theta; \Delta \vDash \Phi$. By Theorem 8.5, $\Psi; \Theta; \Delta \vdash \texttt{eval}(\tau_1) <: \texttt{eval}(\tau_2) : K$. By Theorem 7.3 and two uses of S-Trans, $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K$, as required. □

8.0.3. *Soundness of Typechecking.* As is to be expected, the soundness of the bidirectional type-checking judgment is the most involved. Before we can prove it, a few small lemmata are required. First, we prove (as was noted before) that the output contexts of the I/O method in both typing judgments have a strong regularity condition: the output context is always a subset[19] of the input context.

THEOREM 8.7.

- *If* $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'$ *then* $\Gamma' \subseteq \Gamma$
- *If* $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \tau \Rightarrow \Phi, \Gamma'$ *then* $\Gamma' \subseteq \Gamma$

The next lemma is required for cases of Theorem 8.9 where variables are bound in premises and subsequently removed in the conclusion. In essence, it proves compatibility between the set difference operator which removes variables from the output, and context extension.

THEOREM 8.8. *If* $\Gamma' \subseteq \Gamma$*, then* $(\Gamma, x : \tau) \setminus \Gamma' \subseteq \Gamma \setminus (\Gamma' \setminus \{x : \tau\}), x : \tau$*. Moreover, if:*

[19] This containment is almost always strict: in fact, a corollary of Theorem 8.9 is that the containment is strict unless the term is closed.

- $\Theta \vdash \Delta$ *wf*
- $\Psi; \Theta; \Delta \vdash \Gamma$ *wf*
- $\Psi; \Theta; \Delta \vdash \tau : \star$

Then $\Psi; \Theta; \Delta \vdash_p \Gamma \setminus (\Gamma' \setminus \{x : \tau\}), x : \tau \sqsubseteq (\Gamma, x : \tau) \setminus \Gamma'$.

Proof. The first part follows by an elementary set-theoretic containment proof, and the second is immediate by applying the presuppositions. □

Because the two judgments (checking and inference) are mutually inductively defined, we must prove each judgment's corresponding soundness theorem simultaneously. The theorem must also handle two as-of-yet untreated differences of biλ-Amor and dλ-Amor. First, the syntax of algorithmic terms is different from the declarative ones. To translate an algorithmic term to a declarative one, we rely on the erasure transformation from Section 4.0.1 to remove all type annotations from a term. Second, the use of the I/O method means we must incorporate the "context-strengthening"-style completeness theorem from Section 4.0.4. All together, we arrive at the following theorem.

Theorem 8.9 (Soundness of Type Checking/Inference).

(1) If $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e \downarrow \tau \Rightarrow \Phi, \Gamma'$ *and* $\Theta; \Delta \vDash \Phi$ *then* $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash_p |e| : \tau$

(2) If $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e \uparrow \tau \Rightarrow \Phi, \Gamma'$ *and* $\Theta; \Delta \vDash \Phi$ *then* $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash_p |e| : \tau$

The proof of Theorem 8.9 is not particularly enlightening: we prove both claims simultaneously by induction on the judgment premises, liberally applying Theorem 8.8 when binders are used.

8.0.4. *Completeness of Sorts, Constraints, Contexts, and Kinds.* Perhaps expectedly, the four basic judgments admit very simple completeness proofs. Similarly to their soundness proofs, these are all proved by single-pass inductions on derivations. Again, these proofs are split into two parts to untie the knot: we first prove completeness of "raw" judgments, and then repackage the theorems with presuppositions after all of the raw theorems have been proven.

Theorem 8.10. *If* $\Theta; \Delta \vdash_p I : S$, *then* $\Theta; \Delta \vdash_p I : S \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$.

Theorem 8.11. *If* $\Theta \vdash \Delta$ *wf then* $\Theta \vdash \Delta$ *wf* $\Rightarrow \Phi$ *with* $\Theta; \cdot \vDash \Phi$

THEOREM 8.12. *If* $\Theta; \Delta \vdash_p \Phi$ ***wf***, *then* $\Theta; \Delta \vdash_p \Phi$ ***wf*** $\Rightarrow \Phi'$ *with* $\Theta; \Delta \vDash \Phi'$

THEOREM 8.13. *If* $\Psi; \Theta; \Delta \vdash_p \tau : K$, *then* $\Psi; \Theta; \Delta \vdash_p \tau : K \Rightarrow \Phi$ *with* $\Theta; \Delta \vDash \Phi$

8.0.5. *Completeness of Subtyping.* The proof that biλ-Amor's subtyping is complete is perhaps the most exciting proof we will see. As has been discussed numerous times, dλ-Amor's inclusion of index term-indexed types means that proving the algorithmic subtyping complete is tantamount to deciding β-equivalence of a limited lambda calculus. It may be tempting[20] to attempt to split the proof of completeness of subtyping into two statements: one could attempt to first prove that the algorithmic normal form subtyping relation is complete for types in normal form: i.e. that if $\tau_1 <: \tau_2$ in dλ-Amor with τ_1, τ_2 `nf`, then there is some solvable Φ such that $\tau_1 <:_{\mathtt{nf}} \tau_2 \Rightarrow \Phi$. Unfortunately, this is not easily provable: if the premise is a use of transitivity, the cut type may not be in normal form, and thus the inductive hypothesis cannot be applied.

The actual proof of completeness of algorithmic subtyping relies on two key admissibility theorems, namely of reflexivity and transitivity. Since dλ-Amor's subtyping includes the rules S-Refl and S-Trans but biλ-Amor's doesn't include analogues of these (for the purposes of syntax-directedness), the algorithmic subtyping must be able to emulate these rules whenever they occur in a declarative derivation. In both cases, the proof proceeds by proving the statement for normal forms, and then lifting the result to the full algorithmic relation through normalization.

THEOREM 8.14 (Reflexivity of Algorithmic Subtyping for Neutral Forms). *If* $\Psi; \Theta; \Delta \vdash_p \tau : K$ *and* τ ***ne***, *then* $\Psi; \Theta; \Delta \vdash_p \tau <:_{nf} \tau : K \Rightarrow \Phi$ *with* $\Theta; \Delta \vDash \Phi$

THEOREM 8.15 (Reflexivity of Algorithmic Subtyping for Normal Forms). *If* $\Psi; \Theta; \Delta \vdash_p \tau : K$ *and* τ ***nf***, *then* $\Psi; \Theta; \Delta \vdash_p \tau <:_{nf} \tau : K \Rightarrow \Phi$ *with* $\Theta; \Delta \vDash \Phi$

THEOREM 8.16 (Reflexivity of Algorithmic Subtyping). *If* $\Psi; \Theta; \Delta \vdash_p \tau : K$ *then* $\Psi; \Theta; \Delta \vdash_p \tau <: \tau : K \Rightarrow \Phi$ *with* $\Theta; \Delta \vDash \Phi$

PROOF. By AS-Normalize, it suffices to show that $\Psi; \Theta; \Delta \vdash \mathtt{eval}(\tau) <:_{\mathtt{nf}} \mathtt{eval}(\tau) : K \Rightarrow \Phi$. By Theorem 7.3, we have that $\mathtt{eval}(\tau)$ `nf`, and $\Psi; \Theta; \Delta \vdash_p \mathtt{eval}(\tau) : K$, By Theorem 8.15, we have $\Psi; \Theta; \Delta \vdash \mathtt{eval}(\tau) <:_{\mathtt{nf}} \mathtt{eval}(\tau) : K \Rightarrow \Phi$ and $\Theta; \Delta \vDash \Phi$, as required. □

[20] I certainly was tempted.

THEOREM 8.17 (Transitivity of Algorithmic Subtyping for Normal Forms). *If* $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{nf} \tau_2 : K \Rightarrow \Phi_1$ *and* $\Psi; \Theta; \Delta \vdash_p \tau_2 <:_{nf} \tau_3 : K \Rightarrow \Phi_2$ *with* $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$*, then* $\Psi; \Theta; \Delta \vdash \tau_1 <:_{nf} \tau_3 : K \Rightarrow \Phi$ *such that* $\Theta; \Delta \vDash \Phi$*.*

THEOREM 8.18 (Transitivity of Algorithmic Subtyping). *If* $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K \Rightarrow \Phi_1$ *and* $\Psi; \Theta; \Delta \vdash_p \tau_2 <: \tau_3 : K \Rightarrow \Phi_2$ *with* $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$*, then* $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_3 : K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$

PROOF. By inversion, $\Psi; \Theta; \Delta \vdash \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : K \Rightarrow \Phi_1$ and $\Psi; \Theta; \Delta \vdash \texttt{eval}(\tau_2) <:_{\texttt{nf}} \texttt{eval}(\tau_3) : K \Rightarrow \Phi_1$ By Theorem 7.3 and Theorem 8.17, $\Psi; \Theta; \Delta \vdash \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_3) : K \Rightarrow \Phi$ with $\Theta; \Delta \vDash \Phi$. Then, by AS-Normalize, $\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_3 : K \Rightarrow \Phi$, as required. □

The following theorem is essentially a subtyping version of Theorem 7.1: not only does index term substitution preserve the property that types are in normal form, it also preserves all subtyping relations. This theorem depends on a series of theorems that index-term substitution is admissible for sort assignment, kind assignment, and constraint well-formedness. These are all proved (in Appendix A) by appealing to the corresponding substitution theorem in dλ-Amor, and taking a round trip through soundness and completeness for the judgment in question.

THEOREM 8.19 (Admissibility of Normal Form Subtyping Substitution). *Suppose the following:*

- $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 <:_{nf} \tau_2 : K \Rightarrow \Phi$ *with* $\Theta; \Delta \vDash \Phi$ *and* $\Theta \vdash \Delta$ *wf.*
- $\Theta; \Delta \vdash_p I : S \Rightarrow \Phi_1$ *with* $\Theta; \Delta \vDash \Phi_1$
- $\Theta; \Delta \vdash_p J : S \Rightarrow \Phi_2$ *with* $\Theta; \Delta \vDash \Phi_2$
- $\Theta; \Delta \vDash I = J$

Then, $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] <:_{nf} \tau_2[J/i] : K \Rightarrow \Phi'$ *for some* Φ' *with* $\Theta; \Delta \vDash \Phi'$*.*

A corollary for the above admissibility theorem is that type evaluation essentially commutes with type family application. This theorem is pivotal for proving the AS-FamApp case of Theorem 8.21 below.

THEOREM 8.20 (Type Family Application Commutes with Evaluation). *If* $\Psi; \Theta; \Delta \vdash_p \mathit{eval}(\tau_1) <:_{nf} \mathit{eval}(\tau_2) : S \to K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi \wedge I = J$ *with* $\Theta; \Delta \vdash_p I : S$ *and* $\Theta; \Delta \vdash_p J : S$ *then* $\Psi; \Theta; \Delta \vdash_p \mathit{eval}(\tau_1\ I) <:_{nf} \mathit{eval}(\tau_2\ J) : K \Rightarrow \Phi'$ *for some* $\Theta; \Delta \vDash \Phi'$*.*

PROOF. By inversion on $\Psi;\Theta;\Delta \vdash \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : S \to K \Rightarrow \Phi$.

- For the first case, suppose the derivation was $\Psi;\Theta;\Delta \vdash \lambda i : S.\tau_1' <:_{\texttt{nf}} \tau_2' : S \to K \Rightarrow \Phi$ from $\Psi;\Theta,i : S;\Delta \vdash \tau_1' <:_{\texttt{nf}} \tau_2' : K \Rightarrow \Phi'$. By Theorem 8.19, $\Psi;\Theta;\Delta \vdash_p \tau_1'[I/i] <:_{\texttt{nf}} \tau_2'[J/i] : K \Rightarrow \Phi'$, for some $\Theta;\Delta \vDash \Phi'$. But $\texttt{eval}(\tau_1\ I) = \tau_1'[I/i]$ and $\texttt{eval}(\tau_2\ J) = \tau_2'[J/i]$.
- Now, suppose the derivation was $\Psi;\Theta;\Delta \vdash \tau_1'\ L_1 <:_{\texttt{nf}} \tau_2'\ L_2 : S \to K \Rightarrow \Phi \wedge (L_1 = L_2)$, where $\texttt{eval}(\tau_1) = \tau_1'\ L_1$ and $\texttt{eval}(\tau_2) = \tau_2\ L_2$. These must both be $\texttt{ne}$, since they are both applications, and therefore $\texttt{eval}(\tau_1)\ I = \texttt{eval}(\tau_1\ I)$ and $\texttt{eval}(\tau_2)\ J = \texttt{eval}(\tau_2\ J)$, as required.

□

Finally, we prove the full completeness of algorithmic subtyping. The proof proceeds by a single induction on the hypothesis. The reflexivity and transitivity cases are handled by Theorems 8.16 and 8.18.

THEOREM 8.21 (Completeness of Algorithmic Subtyping). *If $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_2 : K$ then $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_2 : K \Rightarrow \Phi$ and $\Theta;\Delta \vDash \Phi$.*

8.0.6. *Completeness of Typechecking.* Finally, we arrive at the completeness of biλ-Amor's typechecking algorithm. To begin, we must prove the admissibility of dλ-Amor's weakening rule (T-Weaken) in biλ-Amor. This requires a fairly sizable and involved simultaneous induction on the checking and inference judgments, which must account for all of the bells and whistles of the bidirectional typechecking with constraints and I/O contexts. The theorem is best understood as a biλ-Amor-specific version of the mock I/O weakening theorem from Section 4.0.4. When we weaken the affine context, the added variables flow through, and remain unused.

THEOREM 8.22 (Admissibility of Algorithmic Weakening).

(1) If $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e \downarrow \tau \Rightarrow \Phi, \Gamma''$ with $\Theta;\Delta \vDash \Phi$, then whenever $\Psi;\Theta;\Delta \vdash_p \Gamma' \sqsubseteq \Gamma$ and $\Psi;\Theta;\Delta \vdash_p \Omega' \sqsubseteq \Omega$, there are Φ_1, e_1, Γ_1 so that $|e_1| = |e|$, $\Theta;\Delta \vDash \Phi_1$, $\Psi;\Theta;\Delta \vdash_p \Gamma_1 \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash_p \Gamma_1 \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash_p e_1 \downarrow \tau \Rightarrow \Phi_1, \Gamma_1$.

(2) If $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e \uparrow \tau \Rightarrow \Phi, \Gamma''$ *with* $\Theta;\Delta \vDash \Phi$*, then whenever* $\Psi;\Theta;\Delta \vdash_p \Gamma' \sqsubseteq \Gamma$ *and* $\Psi;\Theta;\Delta \vdash_p \Omega' \sqsubseteq \Omega$*, there are* Φ_2*,* e_2*,* Γ_2 *so that* $|e_2| = |e|$*,* $\Theta;\Delta \vDash \Phi_2$*,* $\Psi;\Theta;\Delta \vdash_p \Gamma_2 \sqsubseteq \Gamma' \setminus \Gamma$*,* $\Psi;\Theta;\Delta \vdash_p \Gamma_2 \sqsubseteq \Gamma''$ *and* $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash_p e_2 \uparrow \tau \Rightarrow \Phi_2, \Gamma_2$*.*

The actual statement of completeness is easily understandable. For any declarative type assignment, we can always annotate the term with types so that it can either check or infer, while outputting a valid constraint. We note that it is not strictly necessary to prove the theorem in this form: the careful reader may have noticed that (2) is implied by (1) and a single use of AT-Anno. This method is in fact the traditional way of proving bidirectional completeness. However, the algorithm its proof encodes inserts many unnecessary annotations: any term in inference position will be annotated, even if the term is already a syntactic form whose rule has inferring conclusion. However, by providing an inductive hypothesis for an inference judgment at every stage, we give terms which may infer their own types the opportunity to do so. For this reason, the algorithm which the proof of completeness encodes inserts far fewer annotations than the traditional one.

Theorem 8.23 (Completeness of Type Checking/Inference). *If* $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e : \tau$*, then:*

(1) There are e'*,* Φ'*,* Γ' *such that* $|e'| = e$*,* $\Theta;\Delta \vDash \Phi'$*, and* $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e' \downarrow \tau \Rightarrow \Phi', \Gamma'$*.*

(2) There are e''*,* Φ''*,* Γ'' *such that* $|e''| = e$*,* $\Theta;\Delta \vDash \Phi''$*, and* $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e'' \uparrow \tau \Rightarrow \Phi'', \Gamma''$

9. Implementation of λ-Amor

To exhibit the practical use of dλ-Amor, we present an implementation of biλ-Amor, which we will simply refer to as `LambdaAmor`. The implementation consists of approximately 16,000 lines of OCaml, and is freely available at the URL below, under a BSD license.

https://github.com/jdublu10/lambda-amor

Functionally, the implementation sports a typechecker and interpreter for dλ-Amor, as well as a command-line interface for interactive use.

We begin the section by discussing the format of programs in `LambdaAmor`. To use `LambdaAmor` as a programming language, we require some language features other than the type-checking and evaluation of single expressions, as modeled by biλ-Amor. To this end, we introduce four top-level declaration forms which can be composed to form programs in `LambdaAmor`. Next, we

Let Declaration	`let x:t = e`
Do Declaration	`do x:t <- e`
Type Alias Declaration	`type a:k = t`
Index Alias Declaration	`idx i:s = it`

FIGURE 16. Declarations in `LambdaAmor`

discuss the artifact itself in some depth: giving an overview of the project's structure, elaborating on design decisions, and remarking on a few places where the implementation departs from the theory. Finally, we present an experimental evaluation of the tool, and compare it with existing resource-aware languages.

9.1. Declarations and Structure of `LambdaAmor` Programs. Programs in `LambdaAmor` are lists of *declarations*, which can be any of four forms: let-bound definitions, type and index term aliases, and `do`-declarations. The syntax for each can be seen in Figure 16. These four declaration types are allow programmers to ergonomically write interesting programs by composing them from smaller ones, and building up abstractions. All four declarations should be understood as simply exposing existing judgmental structure from dλ-Amor to the programmer, and as such they do not add or subtract from the expressive power of the language.

Index term and type aliases are somewhat self-explanatory: a programmer may give names to types and index terms they wish to use later. Since types in `LambdaAmor` can get quite complex, liberal use of type aliases is often very helpful. Importantly, both index term and type aliases may be of higher sort and kind, respectively, and so a user can give names to type families and index functions, too.

`do`-declarations are reminiscent of top-level interaction in Haskell's GHCI interpreter. In GHCI, computations in the IO monad entered at top level are not only evaluated, but forced for effect. The `do` of `LambdaAmor` serves a similar role by allowing monadic computations which have previously been built up to be run, and have their *actual* run-time costs computed. The reader may find it helpful to think of `do`-declarations as being a `bind` into an ambient cost-monadic context, again in a manner similar to GHCI. In order to preserve soundness of costs, the only

File	Description
`src/constr_elab.ml`	Constraint elaboration phase to eliminate $\vec{\mathbb{R}}^{+}$
`src/ctx.ml`	Typing context module
`src/env.ml`	Typing environment module
`src/fresh_var.ml`	Globably-unique variable generation
`src/freshen.ml`	AST freshening pass
`src/interp.ml`	Definitional interpreter
`src/lexer.mll`	OCamlLex lexer specification file
`src/main.ml`	Entry point and command line interface
`src/normalize.ml`	Type normalization routine
`src/parser.mly`	OCamlYacc parser specification file
`src/support.ml`	Debug info
`src/syntax.ml`	AST datatypes and helpers
`src/tycheck.ml`	Main bidirectional typechecking algorithms
`src/tyerror.ml`	Error handling monad for typechecking
`src/why_backend.ml`	Code to interface with Why3
`src/why_trans.ml`	Translate constraints into Why3 goals
`la.why`	Why3 supplementary theory file

FIGURE 17. File Structure of `LambdaAmor`

computations that may be run at the top level are those that return *observable* types — i.e. those that are built up from base types, lists, sums, products, and exponentials.

Let-bindings in `LambdaAmor` behave just like top-level bindings in any other functional language. The only twist is that all top-level let-bound variables are required to be exponential variables, and hence cannot use any affine variables bound by `do`-declarations.

9.2. Overview of Phases. A program to be run by `LambdaAmor` follows a straightforward path. It is first lexed and parsed from its textual form into an abstract syntax representation. This abstract syntax is then passed to the typechecker, which closely follows the algorithmic approach prescribed by biλ-Amor. This typechecking emits constraints, which are then passed to an SMT solver using the Why3 prover frontend [7]. If the constraints come back valid, the program can then be passed to the built-in definitional interpreter [66], which runs the program according to the cost semantics of Rajani et al. [64].

For the remainder of this section, we will take a tour through the implementation and design choices of each of the phases of the language's execution. Each of these roughly corresponds to a single module in the `LambdaAmor` source, so a table of files along with their descriptions can be found in Figure 17. Finally, some of the structure of the implementation is borrowed and inspired from previous developments in resource-aware and bidirectional type systems, and so we are careful to flag our predecessors for each pass.

9.2.1. *Lexing and Parsing.* `LambdaAmor` uses off-the-shelf OCaml lexer and parser generators, `ocamllex` and `ocamlyacc`. While not the most performant options, these do fine for our purposes. The syntax of `LambdaAmor` was carefully chosen to resemble the syntax of biλ-Amor as much as possible while retaining an unambiguous grammar, and while ensuring that users need not type unicode symbols.

While the language syntax is closely modeled on that of biλ-Amor, it must be extended to support the top-level features introduced for ease of use in `LambdaAmor`. The only change to the term syntax is the introduction of a syntax (wildcards/underscores) for typed holes in `LambdaAmor`, in the style of OLEG [48], Agda[56], or Haskell, which allow a programmer to typecheck partial programs, and be informed about what types the checker expects to fill the holes.

The lexer and parser are specified in `src/lexer.mll`, respectively `src/parser.mly`. The parser emits an abstract syntax representation of a program. The type of these syntax trees, as well as all of the abstract syntax of the language, is found in `src/syntax.ml`.

9.2.2. *AST Freshening Pass.* Since this development includes no mechanized metatheory, all variables in `LambdaAmor` are represented as strings for simplicity. This poses complications for the substitution operation, as one would need to α-convert variables at each substitution instance. To resolve this, the AST of a program is fed to a "freshener" immediately after parsing, which α-converts all terms so that every bound variable is globally unique[21]. A single freshening pass is sufficient to eliminate the possibility of variable capture in closed terms. When handling

[21]This pass is modeled off of a similar one from the Granule language [60].

open terms, such as those that include index and type aliases, we always re-freshen index terms and types before they are substituted in to prevent capture.

9.2.3. *Normalization.* Thanks to the theoretical simplicity of λ-Amor's type normalization algorithm, `LambdaAmor`'s implementation of it is similarly simple: the code (in `src/normalize.ml`) is only ≈ 150 lines. The procedure works in two passes, first by evaluating object-language types into a meta-language type of types *in normal form*, and then quoting back. In practice, most types in λ-Amor programs are in normal form. To avoid unnecessarily normalizing types, we tag types with a status bit which is set when the type is in normal form.

9.2.4. *Bidirectional Type Checking.* The core typechecking algorithm of `LambdaAmor` is very faithful to the core algorithmic calculus presented in Section 7. Search functions for each of the four user-facing judgments (sort-checking, kind-checking, subtyping, typechecking) are implemented in the file `src/tycheck.ml`. The sort-checking and kind-checking judgments both operate on fully annotated terms. For this reason, we implement full inference and checking for both: with sort/kind as output and input, respectively. Subtyping is implemented as expected: both types are normalized, and then passed to a helper function which decides the normal form subtyping relation of biλ-Amor. Finally, the main pair of type checking and inference judgments are implemented in the usual bidirectional style as a pair of mutually recursive functions. All of these functions, in addition to their usual return types (unit for checking functions, sort/kind/type for inference functions) also return the constraints output by their corresponding judgments, to be passed to the solver.

To simplify the lives of programmers, we do deviate slightly from the core calculus in a few places. First, the type checking and inference judgments include a few "parallel rules": instances where the bidirectional rule has a checking or inference conclusion, but we also include a case for the other mode. While not strictly required for completeness, these extra rules can make programming more ergonomic. Next, we always normalize the output of the type inference function: this is helpful in cases where the type of a term inferred in an elimination position has a β-redex as its head, and not the expected connective. This is also clearly still sound, as it can be emulated by adding an annotation in the requisite elimination position. Finally, we note the behavior of typed holes in `LambdaAmor`. This feature is a practical necessity in languages with type systems as complex as ours. Fortunately, the bidirectional framework makes them simple to implement: when the type checking or inference judgments hit the hole, checking is halted, and the expected type of the hole (in the case of a checking judgment) as well as the current context is printed for the user.

To simplify some of the boilerplate involved in implementing the functions corresponding to each algorithmic judgment, we introduce a monadic discipline inspired by the implementation of BiRelCost [10]. We use a combined state/error monad called `'a checker` to simultaneously handle the four fully structural contexts and the substructural one via the I/O method (hence state, not reader), as well as managing type errors. OCaml's `let*` syntax allows us to cleanly write the typechecker in a manner similar to `do`-notation in Haskell, while a preponderance of useful monadic combinators lifted from BiRelCost make for very readable code.

Of course, the typechecker must not only handle the core term calculus, but also the top-level declaration features. Because of the inclusion of type and index term aliases, the state part of the `checker` monad must also include a type and index term environment which binds aliases to their values, on top of the existing type contexts. The top-level declarations require more choices to be made.

Top level term declarations `let x :  t = e` are implicitly typed as exponential terms: the affine context is erased before checking them, and the variable `x` given type `t` in the exponential context Ω. This allows functions declared at the top level to be used many times, instead of just once, which is the intended pattern of use for a top-level definition.

The only terms which are bound in the affine context at top level are variables resulting from the `do` declarations, which are checked to have monadic type. Since the result of a monadic computation can store potential, the result of a `do` declaration must not be duplicated.

9.2.5. *Constraint Elaboration.* The language of dλ-Amor's constraints includes equations and inequalities over potential vectors (index terms of sort $\vec{\mathbb{R}^+}$). While many solvers allow us to define our own theory to handle this nonstandard type, it is instead preferable to instead appeal to built-in (and highly optimized) real arithmetic theories. For this purpose, all constraints output from the typechecker are elaborated to transform equalities over potential vectors to componentwise equalities over reals.

This works in three phases. First, the length of the potential vectors ($k \geq 2$) is determined. This can be passed as a command-line parameter of `LambdaAmor`, or computed as the largest potential vector appearing in the program. Then, all potential vector constants are padded with zeroes up to length k, while quantifiers over potential vector index variables $i : \vec{\mathbb{R}}^{+}$ are replaced by variables $i_n : \mathbb{R}$, for $0 \leq n < k$. Finally, index terms and (in)equalities over potential vectors are "flattened" to operate componentwise,

This elaboration only works because the language of index terms is sufficiently first-order. If one added truly higher-order index terms, the elaboration would be significantly more complex. The implementation of this transformation is quite simple, and can be found in `src/constr_-elab.ml`.

9.2.6. *Constraint Solving.* The actual constraint solving of `LambdaAmor` is handled by the Why3 platform [7]. Why3 is a unified frontend for a number of SMT solvers, which allows the user to switch between the proovers of their choosing. After the constraint elaboration phase, the constraints are translated into a format understandable by Why3 using its OCaml API. We then interface with the prover by building a Why3 proof goal for each constraint emitted by the typechecker. This set of proof goals is then checked in sequence by the prover, and the results are reported to the user.

9.2.7. *Interpreter.* Finally, a type-correct program can be interpreted. The interpreter included with `LambdaAmor` is a straightforward definitional implementation of the big-step cost-indexed operational semantics of dλ-Amor: nothing too fancy. When the interpreter is invoked, all of the `do` declarations are run. The cost semantics tallies up the total actual cost of running a single declaration, and presents it, along with the statically predictied cost to the user. By the soundness theorem of dλ-Amor, the predicted cost will always be an upper bound on the actual cost.

9.3. Examples and Experimental Evaluation. We now return to the examples presented in Section 3. All of these programs (and more) have been implemented in `LambdaAmor`: the source code for each can be found in the `.la` files in the `examples/` directory of the artifact repository. Below, we discuss each of the examples presented previously, as well as some new ones included with `LambdaAmor`. The examples were chosen to test a variety of `LambdaAmor`'s features, and show a breadth of the kinds of analyses that can be performed using `LambdaAmor`.

- `examples/addone.la`: As presented in Section 3. This program is quite simple, only making use of the RAML-style potentials and a single shift. We include this in the test suite purely as a coherence check.
- `examples/ins_sort.la`: As presented in Section 3. Insertion sort has quadratic complexity, so the inclusion of this example demonstrates how `LambdaAmor` can pass linear constraints to a solver when analyzing functions with degree-2 or higher cost behavior. The file also includes an example of a `do`-declaration, where the `ins_sort` function is run to sort a list.
- `examples/queue.la`: As presented in Section 3. The functional queue is the first example of amortized analysis, which makes nontrivial use of potentials combined with costs.
- `examples/binary.la`: The analysis of a binary counter is a classic example of an amortized analysis, and the running example of Chapter 3. This uses similar techniques to the functional queue (RAML-style potentials, refinements)
- `examples/map.la`: As presented in Section 3. The cost-polymorphic map is an example of a function which cannot be typed in RAML [31], the best-established resource-aware language, due to its complex use of higher order quantifiers over cost families. These quantifiers are reflected in the constraints which are emitted, which provide the first nontrivial goal for the solver backend.
- `examples/foldr.la`: The analysis and type of `foldr` can be expressed in a similar manner to that of `map`. We include this example primarily because it is discussed in depth in the original λ-Amor paper [64]
- `examples/church.la`: Our final example is the assignment of a *very* complex type to church numerals. In short, we generalize the type of the "iterate" function $!(\tau \multimap \tau) \multimap \mathbb{N} \multimap \tau \multimap \tau$ to operate over a type family $\alpha : \mathbb{N} \to \star$ and a cost family $C : \mathbb{N} \to \mathbb{R}^+$ to get the much more precise type:

$$\forall \alpha : \mathbb{N} \to \star . \forall C : \mathbb{N} \to \mathbb{R}^+ . \, (\forall i . \alpha\, i \multimap \mathbb{M} \langle C\, i \rangle \, (\alpha\, (i+1))) \multimap \mathtt{Nat}(n) \multimap \alpha\, 0 \multimap \mathbb{M} \langle \sum C\, i \rangle \, (\alpha n)$$

Further details can be found in Rajani et al. [64]. The file includes two operations defined on these church numerals, namely successor and addition.

File	Solving Time	Checking Time	Total Time
`examples/addone.la`	0.17s	0.0008s	0.29s
`examples/ins_sort.la`	0.29s	0.0039s	0.40s
`examples/queue.la`	0.25s	0.0044s	0.37s
`examples/binary.la`	0.34s	0.0040s	0.46s
`examples/map.la`	0.26s	0.0039s	0.38s
`examples/foldr.la`	0.24s	0.0041s	0.37s
`examples/church.la`	2.10s	0.0046s	2.23s

FIGURE 18. Benchmarks of `LambdaAmor`

The type-checking times for all of these examples can be found in Figure 18. Total time denotes the end-to-end running time, from parsing the input file all the way through running the program (when applicable). Solving time refers to the amount of time spent by the prover in solving the constraints, while checking time refers to the time taken by the type checker itself, as well as the constraint elaboration phase. Each statistic is an average over ten runs. In all cases, the execution time not spent constraint solving is negligible — the SMT solver is by far the largest bottleneck. However, the total time remains low, even for programs like `church` which emit very large high-order constraints. Finally, the total time not accounted for by checking and solving can be atributed to Why3 initialization and file IO.

The evaluation was performed on a 2018 MacBook Pro running OSX 10.13.6 with a 2.3GHz Intel Core i5 processor and 8GB of memory. The artifact itself is written in OCaml, and was compiled using OCaml v4.09.1, along with its included versions of OCamllex and OCamlyacc. Library dependencies can be found in the `dune` file. Constraints emitted from the typechecker were solved by Why3 v1.3.1 [7], using Z3 v4.8.5 [20] as a prover.

10. Related Work

While still greatly under-researched, the prospect formally verifying cost bounds of programs is sufficiently intriguing that researchers have been chipping away at the problem for decades. While `LambdaAmor` takes a specific intrinsic approach to verifying amortized cost, many other approaches exist.

10.0.1. *Program Logics and Static Analyses.* As program logics are commonly used to reason about and verify program correctness, it is natural to consider their utility in verifying costs of programs, as well! Some recent work in this line includes that of Carbonneaux et al. [9], who develop a Hoare logic with quantative reasoning capabilities. Moreover, modern logics like separation logic can used to emulate some kinds of amortized complexity analysis [12, 51]. Çiçek et al. [10] use a *relational* program logic to derive bounds on the cost difference between two programs of the same type. Finally, Li et al. [45] develop a dual pair of program logics [57] for reasoning about the cost of lazy functional programs. A related (usually more automated) technique for verifying cost properties of programs is static analysis. The COSTA project [1, 2, 3] has used this to great effect in their work on automatically verifying cost usage of Java bytecode programs. While all of these papers do a good job of achieving their goals, the goals are very different from ours. In particular, they all take an extrinsic point of view on cost verification: a program and the certificate of its cost are separate.

10.0.2. *Proof Assistant Libraries.* Some work has studied ways of extending existing proof assistants to reasoning about cost, or building libraries which emulate such features. Danielsson [14] presents an Agda library for semiformal cost analysis of purely functional data structures. This work is based on an indexed cost monad similar to that of `LambdaAmor`, but does not include types for handling potential. Interestingly, the framework handles cost analysis of laziness, which we do not. In the Coq side of the world, McCarthy et al. [50] have developed a similar library based on a similar monadic discipline, but this time for by-value evaluation. Both of these works are similar in spirit to `LambdaAmor`, as they provide intrinsic cost reasoning abilities in a richly-typed framework. However, both build this ability on top of an existing proof assistant, whose logic does not have a first-class notion of cost or potential.

10.0.3. *Languages and Proof Assistants with Resource-Aware Type Systems.* The works which bear the most resemblance to `LambdaAmor` are those which develop a full language with a resource-aware type systems from scratch. These projects all fall along a spectrum, from languages with fully automated cost bound inference which on the surface behave much like languages with built-in static analyses, to resource-aware richly-typed proof languages in the style of`LambdaAmor`.

A classic example of the first type is Resource Aware ML [31], an implementation of Automated Amortized Resource Analysis [36], from which λ-Amor borrows its univariate polynomial potentials. While RAML provides highly-automated cost-bound inference, it struggles to handle some moderately complex language features like variable-cost higher-order functions. `LambdaAmor` is at least as expressive as AARA, since AARA embeds into λ-Amor [64]. Moreover, `LambdaAmor` can verify cost-polymorphic functions such as `map` (Section 3). Further along the spectrum is a language like TiML [81], in which the programmer annotates regular ML functions with statically-checked cost bounds. The cost verification techniques involved are somewhat brittle. For instance, TiML includes a set of pattern-based heuristics for solving recurrences, and generates nonlinear constraints when handling polynomial potentials. TiML is also not sound for amortized analysis, as its type system is not affine. Both RAML and TiML can only type terminating functions, while `LambdaAmor` includes general (polymorphic) recursion. The closest-in-spirit language to `LambdaAmor` is Liquid Resource Types (LRT) [42], an ambitious project which builds on Liquid Haskell [79], and previous work on resource-guided synthesis [41] to provide a language for formally verifying amortized cost properties of functional programs. LRT has one notable feature that outpaces all other developments in the space (including `LambdaAmor`): it allows for value-dependent cost analyses. In principle, `LambdaAmor` could be extended to allow for such analyses by enhancing its refinement types, although we have yet to consider this. The largest practical benefit of `LambdaAmor` over LRT is the ability to associate any amount of poential to any value. LRT shares RAML's restriction that potentials may only be associated to the constructors of inductive types, and the amount of potential is dictated by the shape of the constructors. Because of this, to assign different amounts of potential to a datatype, one must carefully re-define the datatype so that its constructors hold the correct amount of potential. In contrast, `LambdaAmor` allows one to separate these concerns, and assign different amounts of potential to the same data type.

Recent work has also attempted to develop type theories with resource analysis capabilities. Quantitative Type Theory (QTT) [5] continues in with philosophy of **?**] in using a generalized form of n-linear types to track resources in a dependent type theory. Similar work by the Granule project [13] [60] seeks similar, albeit more practical, ends: the integration of resource reasoning capabilities into general dependently typed languages. However, none of these developments are specific to the resource of cost. To the author's knowledge, the only type theory with built-in notions of program cost is Cost-Aware Type Theory (CATT) [54].

CHAPTER 3

Amortized Analysis by Recurrence Extraction

1. Introduction

A common technique for analyzing the asymptotic resource complexity of functional programs is the *extract-and-solve* method, in which one extracts a recurrence expressing an upper bound on the cost of the program in terms of the size of its input, and then solves the recurrence to obtain a big-O bound. Typically, the connection between the original program and the extracted recurrence is left informal, relying on an intuitive understanding that the extracted recurrence correctly models the program. Previous work [18, 19, 37, 40, 15] has begun to explore more formal techniques for relating programs and extracted recurrences. The process of extracting a recurrence consists of two phases. The first is a monadic translation into the writer monad $\mathbb{C} \times \cdot$, translating a program to also "output" its cost along with its value. We call the result a *syntactic recurrence*, and at function type, the result is essentially a function that maps a value to a pair consisting of the cost of evaluating that function along with its result. At higher type, the syntactic recurrence maps a recurrence for the argument to a recurrence for the result. A *bounding logical relation* relates programs to syntactic recurrences, and the fundamental *bounding theorem* states that a program and its syntactic recurrence are related, which in particular implies that its actual runtime cost is bounded by the extracted prediction. Since inductive values are translated to (essentially) themselves, this phase does not abstract values to sizes; in effect, the syntactic recurrence describes the cost of the program in terms of its actual arguments. The second phase performs this size abstraction by interpreting (the language of) syntactic recurrences in a denotational model. The interpretation of each type is intended to be a domain of sizes for values of that type, and different models can implement different notions of size. For example, a list value (i.e., the list type and constructors) may be interpreted by its length in one model, or even more exotic notions of size, such as the number of pairwise inversions (as required for an analysis of insertion sort) for a list of numbers. Thus the interpretation of the syntactic recurrence extracted from a source program (what we might call the *semantic recurrence*) is a function that maps sizes (of source-program values) to a bound on the cost of that program on those values. It is these semantic recurrences that match the recurrences that arise from the typical "extract-and-solve" approach to analyzing program cost. Our previous work develops this methodology for functional programs with numbers and lists [18], inductive types with structural recursion [19], general recursion [40], and let-polymorphism [15].

As an example that demonstrates both the approach and a weakness of the underlying technique for cost analysis that it formalizes, let us consider the binary increment function, a standard motivating example for amortized analysis:

$$\begin{array}{lcl@{\qquad}lcl}
\texttt{inc} & : & \texttt{bit list} \to \texttt{bit list} & \texttt{set} & : & \texttt{nat} \to \texttt{bit list} \\
\texttt{inc}\,[\,] & = & [1] & \texttt{set}\,0 & = & [\,] \\
\texttt{inc}\,(0 :: bs) & = & 1 :: bs & \texttt{set}\,(S\,n) & = & \texttt{inc}(\texttt{set}\,n) \\
\texttt{inc}\,(1 :: bs) & = & 0 :: \texttt{inc}\,bs & & &
\end{array}$$

The value part of a monadic translation of a function into $\mathbb{C} \times \cdot$ is a function into a pair, but here we sugar that into a pair of functions, which may be mutually recursive. We denote the cost and value components by $(\cdot)_c$ and $(\cdot)_p$, respectively (this notation is explained in Section 3.1), and charge one unit of cost for each $::$ operation:

$$\begin{array}{lcl@{\qquad}lcl}
\texttt{inc}_c & : & \texttt{bit list} \to \mathbb{C} & \texttt{inc}_p & : & \texttt{bit list} \to \texttt{bit list} \\
\texttt{inc}_c\,[] & = & 1 & \texttt{inc}_p\,[] & = & [1] \\
\texttt{inc}_c\,(0 :: bs) & = & 1 & \texttt{inc}_p\,(0 :: bs) & = & 1 :: bs \\
\texttt{inc}_c\,(1 :: bs) & = & 1 + \texttt{inc}_c\,bs & \texttt{inc}_p\,(1 :: bs) & = & 0 :: \texttt{inc}_p\,bs
\end{array}$$

$$\begin{array}{lcl@{\qquad}lcl}
\texttt{set}_c & : & \texttt{nat} \to \mathbb{C} & \texttt{set}_p & : & \texttt{nat} \to \texttt{bit list} \\
\texttt{set}_c\,0 & = & 0 & \texttt{set}_p\,0 & = & [] \\
\texttt{set}_c\,(S\,n) & = & \texttt{set}_c(n) + \texttt{inc}_c(\texttt{set}_p\,n) & \texttt{set}_p\,(S\,n) & = & \texttt{inc}_p(\texttt{set}_p\,n)
\end{array}$$

We obtain the usual recurrences that we expect when we interpret these syntactic recurrences in an appropriate denotational semantics. We interpret `bit list` and `nat` by $\mathbb{N}$, the natural numbers, and interpret the constructors so that a `bit list` is interpreted by its length and a `nat` by its value. Doing so, we obtain semantic recurrences for the the cost and size of `inc`:

$$\begin{array}{c@{\qquad}c}
T_{\texttt{inc}}(0) = 1 & S_{\texttt{inc}}(0) = 1 \\
T_{\texttt{inc}}(n+1) = \max\{1, 1 + T_{\texttt{inc}}(n)\} & S_{\texttt{inc}}(n+1) = \max\{1+n, 1 + S_{\texttt{inc}}(n)\}
\end{array}$$

The usual techniques (in the semantics) then allow us to conclude that $T_{\texttt{inc}}(n) \le n+1$ and $S_{\texttt{inc}}(n) \le n+1$, which are correct and tight bounds on the cost and size of the `inc` function.

The semantic recurrences for `set` are

$$\begin{aligned} T_{\texttt{set}}(0) &= 0 & S_{\texttt{set}}(0) &= 0 \\ T_{\texttt{set}}(n+1) &= T_{\texttt{set}}(n) + T_{\texttt{inc}}(S_{\texttt{set}}(n)) & S_{\texttt{set}}(n+1) &= S_{\texttt{inc}}(S_{\texttt{set}}(n)) \\ &\le T_{\texttt{set}}(n) + S_{\texttt{set}}(n) + 1 & &\le S_{\texttt{set}}(n) + 1 \end{aligned}$$

and so we conclude that $S_{\texttt{set}}(n) \le n$ and hence $T_{\texttt{set}}(n) \in O(n^2)$, both of which are correct, but not tight, bounds.

On the one hand, through syntactic recurrence extraction, the bounding theorem, and soundness of the semantics, we have a formal connection between the original programs and the semantic recurrences that bound their cost and size. On the other, this example demonstrates a well-understood weakness in the informal technique: while the cost of a composition of functions is bounded by the composition of their costs, the bound is not necessarily tight. The tight bound is usually established with some form of amortized analysis, and *the goal of this paper is to provide a formalization of the banker's method for amortized analysis comparable to the formalization of [18, 19, 37] for non-amortized analysis.*

The *banker's method* for amortized analysis [75] permits one to "prepay" time cost to generate "credits" that are "spent" later to reduce time cost, rearranging the accounting of costs from one portion of a program to another (in particular, generating a credit costs 1 unit of time, while spending a credit reduces the cost by 1 unit of time). In this example, we maintain the invariant that one credit is attached to every 1 bit in the counter representation. The *amortized cost* of flipping a bit from 0 to 1 is then 2 units of time—one for the actual bit flip plus one to generate the credit. However, the amortized cost of flipping a bit from 1 to 0 is 0 units of time—the bit flip takes one unit of time, but that is paid for by the credit. Using these new amortized costs, we can see that $T_{\texttt{inc}}(n)$ is $O(1)$ amortized: in the case where the first bit is 0, we flip it to 1, which costs 2 units of time, and stop. In the case where the first bit is 1, we flip it *for free* to 0, and then make a recursive call, which inductively is bounded by 2. So $T_{\texttt{inc}}(n) = 2$, which means that $T_{\texttt{set}}(n) = 2n$, amortized. Since a single run of `set` starts with no credits, its actual cost will be bounded by the amortized cost $2n$: all of the credits spent during the call to `set`, which subtract from the cost, must have been created earlier, incurring a cost which balances out the gain garnered from spending it.

Formalizing recurrence extraction for the banker's method for amortized analysis requires us to move from a relatively standard source language based on the simply-typed λ-calculus with inductive datatypes to a more specialized one. We do not expect amortization policies (e.g. generate a credit when flipping a bit from 0 to 1, to be spent when flipping a bit from 1 to 0) to be automatically inferable in the general case—these policies are the part of an amortized analysis that requires the most cleverness. To notate these policies, we use an *intermediate language* λ^A (Section 2) [1], which has "effectful" operations for generating and spending credits (`create` and `spend`), as well as a modal type operator $!_\ell$ for associating credits with values (e.g. storing a credit with each 1 in a bit list). The type $!_\ell A$ classifies a value of type A that has ℓ credits associated with it. To correctly manage credits, this intermediate language is based on a form of linear logic, which prevents spending the same credit more than once; in particular, λ^A is an affine lambda calculus with all of the standard connectives $\otimes, \oplus, \&, \multimap, !$ plus multiplicities $!^k A$ (where k is a positive number) for tracking multiple-use values. The type structure of the intermediate language is inspired by the credits (written as $\diamond$) of [34, 35], n-linear types (e.g. [25, 65, 49, 5]), and the uses of credits and linear logic in in automatic amortized resource analysis (AARA) (e.g. [36, 32, 41]).

The target of the monadic translation is the *recurrence language* $\lambda^{\mathbb{C}}$, which, following [19, 37], is a standard simply-typed λ-calculus with a base type for costs (linearity is not needed at this stage). It is equipped with an inequality judgment $E \leq_T E'$ that can be used to express upper bounds. The translation we define here extracts a recurrence for the *amortized* cost of the program (where the costs have been "rearranged"), by translating the credit generation and spending operations in λ^A to modifications of the cost. We define a bounding relation (a cross-language logical relation) for the amortized case, and prove that a term is related to its extraction. As a corollary, we obtain that the amortized cost of running a program from λ^A is bounded by the cost component of its translation into $\lambda^{\mathbb{C}}$; for programs that use no external credits, this gives a bound on its actual cost as well. The recurrence language, recurrence extraction and bounding theorem are described in Section 3. Next, we use a denotational

[1] In an unfortunate coincidence, the recurrence extraction project including λ^A was developed concurrently with the λ-Amor project by disjoint sets of authors. This led to a name collision that we hope will not cause the reader too much grief.

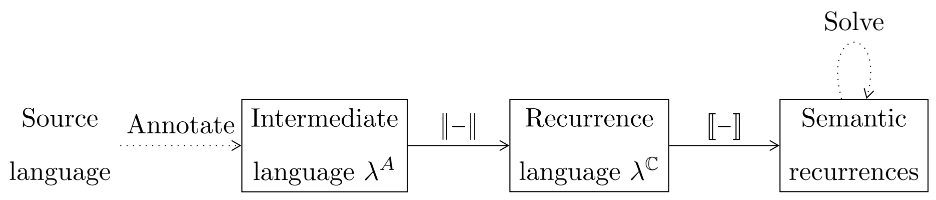
Source language
Annotate
Intermediate language λ^A
$\|-\|$
Recurrence language $\lambda^{\mathbb{C}}$
$[\![-]\!]$
Semantic recurrences
Solve

Figure 1. Recurrence Extraction Pipeline

semantics of the recurrence language in preorders, similar to [19], to justify the consistency of the recurrence language $\leq$ judgment, and to simplify and solve extracted recurrences (Section 4).

The version of λ^A and the recurrence extraction presented through Section 4 allows a statically fixed number of credits to be stored with each element of a data structure (e.g. 1 credit on element of a list, so n credits overall). For some analyses, it is necessary to choose the number of credits stored with an element dynamically. For example, when analyzing splay trees [72], the number of credits stored at each node in the tree is a function of the size of the subtree rooted at that node, which varies for different tree nodes. To support such analyses, we extend λ^A with existential quantifiers over credit variables in Section 5, and use them to code a portion of Okasaki [58]'s analysis of splay trees in our system.

The process of extracting and solving a recurrence in diagrammed in Figure 1. While automation of the annotation and solving steps is a worthwhile goal, our main motivation in this paper is to formally justify the extract-and-solve method for amortized analysis, a technique that we teach and that is typically used by practitioners. Connecting the extracted recurrence in terms of user-defined notions of size to the operational cost is the least justified step in this process, and so a formal account of it has important foundational value. It could likewise have important practical value: because students and practitioners are trained in the use of cost recurrences, reverse-engineering a recurrence that yields a worse-than-expected cost bound to the (mis)implementation may require a lower cognitive load than doing the same with more sophisticated techniques. Moreover, though this technique is less automated than others, it can handle at least some examples that existing techniques cannot—to our knowledge, splay trees cannot be analyzed by the existing automatic techniques. We give a detailed comparison with related work in Section 6.

2. Intermediate Language λ^A

In this section we discuss the static and operational semantics of λ^A, which is an *affine* lambda calculus—it permits weakening (unused variables) but not contraction (duplication of variables). It includes some standard connectives of linear logic, such as positive/eager/multiplicative products ($\otimes$ and 1), sums/coproducts ($\oplus$), and functions ($\multimap$), as well as negative/lazy/additive products ($\&$). The language has two basic datatypes, natural numbers ($\mathbb{N}$) and (eager) lists (`[`A`]`), both with structural recursion (though we expect these techniques to extend to all strictly positive inductive types [19, 15]).

In addition to these, λ^A contains some constructs specific to its role as an intermediate language for expressing amortized analyses. First, instead of fixing the operational costs of λ^A's programs themselves, we include a `tick` operation which costs 1 unit of time, and assume that the translation of a program into λ^A has annotated the program with sufficient ticks to model the desired operational cost [14] (for example, we can charge only for bit flips in the above binary counter program).

Second, we have operations `create` and `spend` for creating and spending credits, which respectively increase and decrease the *amortized* cost of the program *without changing* the true operational cost.

Third, we have a type constructor $!_\ell A$, where a value of this type is a value of type A with ℓ credits attached; its introduction and elimination rules allow for the movement of credits around a program. The combination of of `spend` and the $!_\ell$ modality motivates our affine type system: because spending credits decreases the amortized cost of a program, we must ensure that a credit is spent only once, so credits should not be duplicated; because credits can be stored in values, values cannot in general be duplicated as well. However, λ^A does allow credit weakening—choosing not to spend available credits—because this increases the amortized cost (relative to spending the credits), and we are interested in upper bounds on running time. While the basic affine type system allows a variable to be used only once, to simplify the expression of programs that use a variable a fixed number of times, we use n-linear types (see e.g. [25, 65, 49, 5]), where variables are annotated with a multiplicity k, and can be used at most k times.[2] This is

[2]While Girard's notation for multiplicities is $!_k A$ [25], we write superscripts following Atkey [5], and write subscripts for the credit-storing modality, which is used more frequently in our system.

$$\begin{array}{lll} \text{Types} & A, B, C & ::= \mathbb{N} \mid \texttt{[}A\texttt{]} \mid A \multimap B \mid A \otimes B \mid A \oplus B \mid A \& B \mid !^k_\ell A \\ \text{Terms} & M, N & ::= x \mid \texttt{tick}\,;\, M \mid \texttt{create}_\ell\, M \mid \texttt{spend}_\ell\, M \mid \texttt{save}^k_\ell\, M \mid \texttt{transfer}_{k'}\, !^k_\ell\, x = M \;\texttt{to}\; N \\ & & \mid \lambda x.M \mid M\; N \mid \texttt{inl}\, M \mid \texttt{inr}\, M \mid \texttt{case}_{k'}(M, x.N_1, y.N_2) \mid \langle M, N \rangle \mid \pi_1 M \mid \pi_2 M \\ & & \mid \texttt{split}(M, x.y.N) \mid 0 \mid S(M) \mid \texttt{nrec}(M, N_1, N_2) \mid \texttt{[]} \mid M :: N \mid \texttt{lrec}(M, N_1, N_2) \end{array}$$

FIGURE 2. λ^A Grammar

internalized by a modality $!^k A$, which represents an A that can be used at most k times. We additionally allow k to be ∞, in which case $!^\infty A$ is the usual exponential of linear logic, allowing unrestricted use. Using this modality, standard functional programs can be coded in λ^A, but our current recurrence extraction does not handle the $!^\infty$ fragment very well, as explained below—at present, we use $!^\infty$ mainly as a technical device for typing recursors. It is technically convenient to combine the two modalities into one type former $!^k_\ell A$, which represents an A that can be used k times, which also has ℓ credits attached (total, not ℓ credits with each use). Because k is a coefficient but ℓ is an additive constant, the individual modalities are recovered as $!^k A := !^k_0 A$ and $!_\ell A := !^1_\ell A$. In pure affine logic, one can think of $!^k_\ell A$ as $X \otimes \ldots \otimes X \otimes A \otimes \ldots \otimes A$ with ℓ Xs and k A's (in the case where k and ℓ are finite), for an atomic proposition X representing a single credit. However, our judgmental presentation is easier to work with for our bounding relation and theorem below, and the n-linear modality $!^k A$ ensures that additional invariant that it is the *same* value that can be used k times, i.e. it only allows the diagonal of $A \otimes \ldots \otimes A$.

2.1. Type System. In Fig. 3 we define a typing judgment of the form $\Gamma \vdash_f M : A$, where Γ is a standard context $x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$ and f is a *resource* term of the form $a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + \ell$, where $x_1, \ldots, x_n$ are the variables in Γ and a_i and ℓ are natural numbers or ∞. The resource term f can be thought of as annotating each variable x_i with the number of times a_i that it is allowed to occur, and additionally annotating the judgment with a nonnegative "bank" ℓ of available credits that can be used. For example, the judgment $x : A, y : B, z : C \vdash_{3x+2y+0z+2} M : D$, means that M is a term of type D, which may use x at most 3 times, y at most twice, z not at all, and has access to 2 credits. We consider these resource terms up to the usual arithmetic identities (associativity, unit, commutativity, distributivity, $0f = 0$, $\infty k = \infty$ otherwise, etc.). In the admissible substitution rule, we write $g[f/x]$ to denote the result of normalizing the textual substitution of f for x in g according to these identities;

e.g. $(3x + 2y + 2)[10a + 11b + 3/x] = 30a + 33b + 2y + 11$. Our judgmental presentation of n-linear types differs from some existing ones– the reader more familiar with Girard's BLL [25] may read $\Gamma \vdash_f M : A$ as analogous to $!_{\bar{f}}\Gamma \vdash M : A$ – but this type system was derived as an instance of a general framework for modal types [46], which, for our purposes, simplifies the presentation of standard metatheorems like substitution. Note that the resource terms f play a different role than the resource polynomials in Bounded Linear Logic and AARA [25, 32], which provide a mechanism for measuring the size and credit allocation in a data structure. The resource terms are also affine in the sense of a polynomial—the exponent of every variable is 1, except for the constant term ℓ—but we will avoid this meaning of affine to avoid confusion with "affine logic" (allowing weakening but not contraction).

2.1.1. *Structural Rules.* The rules make three structural principles admissible:

THEOREM 2.1 (Admissible structural rules).

- *Resource Weakening: Write $g \geq f$ for the coefficient-wise partial order on resource terms ($a_1x_1 + a_2x_2 + \ldots + \ell \geq b_1x_1 + b_2x_2 + \ldots + \ell'$ iff $a_i \geq b_i$ for all i and $\ell \geq \ell'$). Then if $\Gamma \vdash_f M : A$ and $g \geq f$ then $\Gamma \vdash_g M : A$.*
- *Variable Weakening: If $\Gamma \vdash_f M : A$ and y does not occur in Γ, then $\Gamma, y : B \vdash_{f+0y} M : A$.*
- *Substitution: If $\Gamma \vdash_f M : A$ and $\Gamma, x : A \vdash_g N : B$, then $\Gamma \vdash_{g[f/x]} N[M/x] : B$*

PROOF. By induction on derivations. □

First, we can weaken the resource subscript, allowing more uses of a variable or more credits in the bank (e.g. if $\cdot \vdash_3 M : A$, then $\cdot \vdash_5 M : A$). Second, we can weaken a context to include an unused variable (we write $f + 0y$ for emphasis, but by equating resource terms up to arithmetic identities, this is just f). Third, we can substitute one term into another, performing the corresponding substitution on resource terms. The idea is that, if N uses a variable x say 3 times, then it requires 3 times the resources needed to make M to duplicate M three times; this multiplication occurs when substituting f for the occurrence of x in g.

2.1.2. *Multiplicative/Additive Rules in n-linear Style.* In the n-linear types style of presentation, rules of linear logic that traditionally split the context (e.g. $\otimes$ introduction, $\multimap$ elimination) sum the resources used in each premise, but keep the same underlying variable context Γ in all

Admissible: $$\frac{\Gamma \vdash_f M : A \quad g \geq f}{\Gamma \vdash_g M : A} \qquad \frac{\Gamma \vdash_f M : A}{\Gamma, y : B \vdash_{f+0y} M : A} \qquad \frac{\Gamma \vdash_f M : A \quad \Gamma, x : A \vdash_g N : B}{\Gamma \vdash_{g[f/x]} N[M/x] : B}$$

$$\frac{}{\Gamma, x : A \vdash_{f+x} x : A} \qquad \frac{\Gamma \vdash_f M : A}{\Gamma \vdash_f \texttt{tick}\ ;\ M : A} \qquad \frac{\Gamma \vdash_{f+\ell} M : A}{\Gamma \vdash_f \texttt{create}_\ell\ M : A} \qquad \frac{\Gamma \vdash_f M : A}{\Gamma \vdash_{f+\ell} \texttt{spend}_\ell\ M : A}$$

$$\frac{\Gamma \vdash_f M : A \quad kf + \ell \leq g}{\Gamma \vdash_g \texttt{save}^k_\ell\ M : !^k_\ell A} \qquad \frac{\Gamma \vdash_f M : !^k_\ell A \quad \Gamma, x : A \vdash_{g+k'(kx+\ell)} N : B}{\Gamma \vdash_{k'f+g} \texttt{transfer}_{k'}\ !^k_\ell\, x = M\ \texttt{to}\ N : B}$$

$$\frac{\Gamma, x : A \vdash_{f+x} M : B}{\Gamma \vdash_f \lambda x.M : A \multimap B} \qquad \frac{\Gamma \vdash_f M : A \multimap B \quad \Gamma \vdash_g N : A}{\Gamma \vdash_{f+g} M\ N : B}$$

$$\frac{\Gamma \vdash_f M : A}{\Gamma \vdash_f \texttt{inl}\, M : A \oplus B} \qquad \frac{\Gamma \vdash_f M : B}{\Gamma \vdash_f \texttt{inr}\, M : A \oplus B} \qquad \frac{\begin{array}{c}\Gamma \vdash_f M : A \oplus B \\ \Gamma, x : A \vdash_{g+k'x} N_1 : C \\ \Gamma, y : B \vdash_{g+k'y} N_2 : C\end{array}}{\Gamma \vdash_{k'f+g} \texttt{case}_{k'}\,(M,\, x.N_1\,,\, y.N_2) : C}$$

$$\frac{\Gamma \vdash_f M : A \quad \Gamma \vdash_f N : B}{\Gamma \vdash_f \langle M, N \rangle : A \& B} \qquad \frac{\Gamma \vdash_f M : A_1 \& A_2}{\Gamma \vdash_f \pi_i M : A_i}$$

$$\frac{}{\Gamma \vdash_f () : 1} \qquad \frac{\Gamma \vdash_f M : A \quad \Gamma \vdash_g N : B}{\Gamma \vdash_{f+g} (M, N) : A \otimes B} \qquad \frac{\Gamma \vdash_f M : A \otimes B \quad \Gamma, x : A, y : B \vdash_{g+k'(x+y)} N : C}{\Gamma \vdash_{k'f+g} \texttt{split}_{k'}(M,\, x.y.N) : C}$$

$$\frac{}{\Gamma \vdash_f 0 : \mathbb{N}} \qquad \frac{\Gamma \vdash_f M : \mathbb{N}}{\Gamma \vdash_f S(M) : \mathbb{N}} \qquad \frac{\begin{array}{c}\Gamma \vdash_f M : \mathbb{N} \\ \Gamma \vdash_{g_1} N_1 : 1 \multimap C \\ \Gamma \vdash_{g_2} N_2 : !^\infty_0(\mathbb{N} \otimes (1 \multimap C) \multimap C)\end{array}}{\Gamma \vdash_{f+g_1+g_2} \texttt{nrec}\,(M, N_1, N_2) : C}$$

$$\frac{}{\Gamma \vdash_f \texttt{[]} : \texttt{[}A\texttt{]}} \qquad \frac{\Gamma \vdash_f M_1 : A \quad \Gamma \vdash_g M_2 : \texttt{[}A\texttt{]}}{\Gamma \vdash_{f+g} M_1\ ::\ M_2 : \texttt{[}A\texttt{]}} \qquad \frac{\begin{array}{c}\Gamma \vdash_f M : \texttt{[}A\texttt{]} \\ \Gamma \vdash_{g_1} N_1 : 1 \multimap C \\ \Gamma \vdash_{g_2} N_2 : !^\infty_0(A \otimes (\texttt{[}A\texttt{]} \& C) \multimap C)\end{array}}{\Gamma \vdash_{f+g_1+g_2} \texttt{lrec}\,(M, N_1, N_2) : C}$$

FIGURE 3. λ^A Typing Rules

premises. For example, in a positive pair $(M, N) : A \otimes B$, if M is allowed to use x 3 times and N is allowed to use x 4 times, then the whole pair must be allowed to use x 7 times. As a special case, if a variable is not allowed to occur in, e.g., N, it can be marked with a coefficient of 0. On the other hand, rules for additives (e.g. pairing for $A\&B$) use the same resource term in multiple premises. While the elimination rule for $\oplus$ is additive in sequent calculus style, in natural deduction there is some summing because it builds in a cut for the term being case-analyzed.

2.1.3. *Ticks, and Creating/Spending Credits.* The `tick` ; M construct is used to mark program points that are intended to incur one unit of time cost (e.g. bit flips in the binary counter example); it uses the same resources as M.

`create` is the means to create credits, where $\texttt{create}_\ell$ gives M access to ℓ extra credits to use, along with whatever resources are present in the ambient context; formally, this is represented by adding to the "bank" in the premise of the typing rule for M. In the operational semantics and recurrence extraction below, `create` adds ℓ steps to the amortized cost of M—it is used to "prepay" for later costs.

`spend` is the means to spend credits, where $\texttt{spend}_\ell$ spends ℓ credits; because credits can only be spent once, these ℓ credits in the conclusion of the typing rule are not also available in the premise for M. In the operational semantics/recurrence extraction, `spend` subtracts ℓ steps from the amortized cost of M—it is used to take advantage of prepaid steps. Note that `spend` satisfies the same typing judgments as an instance of resource weakening (because $f + \ell \geq f$); the "silent" weakening does not change the amortized cost, but instead is a case where our recurrence extraction might obtain a non-tight upper-bound.

2.1.4. $!^k_\ell$ *Modality.* Instead of having two separate modalities, one for n-use types and the other for types storing credits, we combine them into a single modality $!^k_\ell A$. A value of type $!^k_\ell A$ is a k-use A with ℓ credits attached (not $k \cdot \ell$ credits, which is what one would expect if each use had ℓ credits attached—though that could be modeled by the type $!^k_0(!^1_\ell A)$). While we write a and ℓ for nonnegative numbers or ∞, we restrict k to range over a *positive* number or ∞ – i.e. we do not allow a "zero-use" modality $!^0_\ell A$, which would complicate the erasure of λ^A to regular simply typed lambda calculus.

The introduction rule for $!^k_\ell$ says that if we can prove M has type A with f resources, then a version of M that can be used k times requires kf resources. If in addition, ℓ credits are to be attached, then $kf + \ell$ resources are required. Intuitively, one can think of $\texttt{save}^k_\ell\ M$ as the act of running M once to obtain its value, but repeating whatever requirement it imposes on the bank k times, which justifies making k uses of its value, and then attaching ℓ credits to this value. In order to make resource weakening admissible in general, it is necessary to build weakening into this rule (the second premise).

The elimination rule for the modality allows for the credit stored on a term to be released into the ambient context of another in order to be redistributed or spent. We first present a simplified version, and then explain the general version. Given $\Gamma \vdash_f M :!^k_\ell A$, we essentially have k copies of an A, along with ℓ extra credits. Given a term N which can use k copies of an A and ℓ credits, $\Gamma, y : A \vdash_{ky+\ell} N : C$, we can form the term $\Gamma \vdash_f \texttt{transfer}\,!^k_\ell y = M \texttt{ to } N : C$, which, intuitively, deconstructs M into its k-usable value and ℓ credits, and moves them to N, where they can be used. On top of this version, we make two modifications. Firstly, N should have access to resources other than just what's provided to it by M– so we add a resource term g available in N (and therefore required to type the `transfer`). Secondly, it may be necessary at the site of the transfer to further duplicate the $M :!^k_\ell A$ — this is required to prove a fusion law below, for example. To support this, we parameterize the `transfer` term by another number, k', arriving at the version of the rule presented in Figure 3, which should be thought of as eliminating k' copies of a $!^k_\ell A$ at once. The rules for other positive types $(\oplus, \otimes)$ similarly permit elimination of multiple copies at once.

The ! modality satisfies the following interactions with other logical connectives, where we write $A \dashv\vdash B$ to mean interprovability/functions in both directions:

Theorem 2.2 (Fusion Laws).

(1) $!^{k_1 k_2}_{\ell_1 + k_1 \cdot \ell_2} A \dashv\vdash !^{k_1}_{\ell_1} !^{k_2}_{\ell_2} A$

(2) $!^k_{\ell_1+\ell_2}(A \otimes B) \dashv\vdash !^k_{\ell_1} A \otimes !^k_{\ell_1} B$

(3) $!^k_\ell (A \oplus B) \dashv\vdash !^k_\ell A \oplus !^k_\ell B$

2.1.5. *Natural Number Recursor.* For natural numbers, while the rules for zero and successor are standard, the recursor takes a bit of explanation. We think of the recursor `nrec` as a function constant of type $\mathbb{N} \multimap (1 \multimap C) \multimap !^\infty_0 (\mathbb{N} \times (1 \multimap C) \multimap C) \multimap C$. The base case is "thunked" because we think of $\multimap$ as a call-by-value function type, but the base case should not be evaluated until the recurrence argument is 0. The ordinary type for the step function (inductive case) would be $(\mathbb{N} \times C \multimap C)$, but we also suspend the recursive call, to allow for a simple case analysis that chooses not to use the recursive call. The $!^\infty_0$ modality surrounding the step function is needed to ensure that the step function itself does not use any ambient credits, which is necessary because the step function is applied repeatedly by the recursor (n times if the value of M is n). Without this restriction, one could, for example, iterate a step function that spends k credits to subtract Mk credits from the amortized cost, while only having k credits in the bank to spend. For example, without the use of $!_0^\infty$, the term $\cdot \vdash_1 \texttt{nrec}\,(7, \lambda\_.0, \lambda\_.\texttt{spend}_1\, 0) : \mathbb{N}$ typechecks with only one credit in the ambient bank, but intuitively subtracts 7 from the amortized cost, rather than just the 1 credit that was allowed. We solve this problem using the type $!_0^\infty A$ (where A is the ordinary type of the step function $\mathbb{N} \otimes (1 \multimap C) \multimap C$), which represents an infinitely duplicable A that stores no additional credits. Being infinitely duplicable is an over-approximation, because the step function really only needs to be run M times, but being more precise would require reasoning about such values in the type system.

In the common case, the step function will use other infinite-use variables but no credits from the bank. A typical typing derivation for this case, where H is the type of a helper function and A is the type of the step function, would be

$$\frac{f : H \vdash_{\infty f} N_2' : A}{f : H \vdash_{\infty(\infty f) = \infty f} \texttt{save}_0^\infty\; N_2' :!_0^\infty A}$$

Using this as the third premise of the typing rule of $\texttt{nrec}$, we see that such an $\texttt{nrec}$ itself requires only the credits demanded by the number argument (M) and base case (N_1), assuming f is substituted by a helper function that uses no credits.

The way in which the $!^\infty$ modality "prevents" the use of credits from the bank is somewhat subtle: a step function *can* use credits from the bank, but this will require the bank to be infinite in the conclusion. This is because the introduction rule for $!_0^\infty$ inflates any finite resources to ∞ in the conclusion:

$$\frac{f : H \vdash_{2x+3} N_2' : A}{f : H \vdash_{\infty(2f+3) = \infty f + \infty} \texttt{save}_0^\infty\; N_2' :!_0^\infty A}$$

Thus, the step function is only permitted to use credits from the bank when the bank has ∞ credits in the conclusion, while we are generally interested in programs that use finitely many credits.

2.1.6. *List Recursor.* The list recursor $\texttt{lrec}\,(M, N_1, N_2)$ has the same "credit capture" problem as the recursor on naturals, which we solve using $!_0^\infty$. The list recursor has another challenge, though, because unlike a natural number, the values of the list can themselves store credits. Because of this, to prevent credits from being duplicated, in the cons case, the recursor may use *either* the tail of the list or the recursive result, but not both. We code this using an internal choice/negative product &. The negative product will itself be treated as a lazy type constructor, where an $A \& B$ pair is a value even when the A and B are not, so we do not need to further thunk the recursive result C here.

2.2. Operational Semantics for λ^A. We present a call-by-value big-step operational semantics for λ^A in Figure 4, whose primary judgment form is $M \downarrow^{(n,r)} v$, which means that M evaluates to the value v with cost (n, r). The first component of the cost, n (a non-negative number) indicates the *real cost* of evaluating M, in this case the number of `ticks` performed while evaluating M. The second component, r (which can be any integer), tracks `create`s and `spend`s — the (possibly negative) sum total of credits created and spent while evaluating M, where creating is positive and spending is negative. The *amortized cost* of evaluating M is $n+r$: the number of "actual" steps taken, plus the number of credits created, minus the number spent.

One reason we separate n and r in the judgment form is that there is a straightforward *erasure* of λ^A to ordinary simply typed λ-calculus (STLC with a `tick` operation), in which evaluating the STLC program has cost (number of ticks) n. Briefly, this translation translates $!^k_\ell A$ to A, translates all of the linear connectives to their unrestricted counterparts, drops all `create`, `spend`, `save` term constructors, and translates `transfer` to a `let`. The definition of n in each of our inference rules for $M \downarrow^{(n,r)} v$ is the same as the usual cost for STLC with a tick operation, so this erasure preserves cost. Because of this erasure, the n in $M \downarrow^{(n,r)} v$ is a meaningful cost to bound. Further, the distinction between n and r is why we have separate terms `create` and `tick`: `tick` increases the operational cost which should be preserved under erasure, while `create` increase the amortized cost only.

As discussed in Section 2.1.3, $\texttt{create}_\ell\ M$ creates ℓ credits for M to use for the price of ℓ units of time cost, whereas `spend` subtracts from the amortized cost of an expression — a speedup which is paid for by the ℓ credits which the body is no longer allowed to use. Both are reflected by corresponding changes to r.

The operational intuition for $\texttt{save}^k_\ell\ M : !^k_\ell A$ is that it runs M once, but repeats whatever effect this had on the credit bank k times, which justifies using the credits in the value of M

$$\frac{M \downarrow^{(n,r)} v}{\texttt{tick}\ ;\ M \downarrow^{(1+n,r)} v} \qquad \frac{M \downarrow^{(n,r)} v}{\texttt{create}_\ell\ M \downarrow^{(n,r+\ell)} v} \qquad \frac{M \downarrow^{(n,r)} v}{\texttt{spend}_\ell\ M \downarrow^{(n,r-\ell)} v}$$

$$\frac{M \downarrow^{(n,r)} v}{\texttt{save}^k_\ell\ M \downarrow^{(n,kr)} \texttt{save}^k_\ell\ v} \qquad \frac{M \downarrow^{(n_1,r_1)} \texttt{save}^k_\ell\ v_1 \quad N[v_1/x] \downarrow^{(n_2,r_2)} v}{\texttt{transfer}_{k'}\ !^k_\ell\, x = M\ \texttt{to}\ N \downarrow^{(n_1+n_2,k'r_1+r_2)} v}$$

$$\frac{}{\lambda x.M \downarrow^{(0,0)} \lambda x.M} \qquad \frac{M \downarrow^{(n_1,r_1)} \lambda x.M' \quad N \downarrow^{(n_2,r_2)} v_1 \quad M'[v_1/x] \downarrow^{(n_3,r_3)} v}{M\ N \downarrow^{(n_1+n_2+n_3,r_1+r_2+r_3)} v}$$

$$\frac{M \downarrow^{(n,r)} v}{\texttt{inr}\, M \downarrow^{(n,r)} \texttt{inr}\, v} \qquad \frac{M \downarrow^{(n_1,r_1)} \texttt{inr}\, v_1 \quad N_2[v_1/x] \downarrow^{(n_2,r_2)} v}{\texttt{case}_{k'}\,(M,\, x.N_1\,,\, y.N_2) \downarrow^{(n_1+n_2,k'r_1+r_2)} v}$$

$$\frac{M \downarrow^{(n,r)} v}{\texttt{inl}\, M \downarrow^{(n,r)} \texttt{inl}\, v} \qquad \frac{M \downarrow^{(n_1,r_1)} \texttt{inl}\, v_1 \quad N_1[v_1/x] \downarrow^{(n_2,r_2)} v}{\texttt{case}_{k'}\,(M,\, x.N_1\,,\, y.N_2) \downarrow^{(n_1+n_2,k'r_1+r_2)} v}$$

$$\frac{}{\langle M, N\rangle \downarrow^{(0,0)} \langle M, N\rangle} \qquad \frac{M \downarrow^{(n_1,r_1)} \langle N_1, N_2\rangle \quad N_i \downarrow^{(n_2,r_2)} v}{\pi_i M \downarrow^{(n_1+n_2,r_1+r_2)} v}$$

$$\frac{M \downarrow^{(n_1,r_1)} v_1 \quad N \downarrow^{(n_2,r_2)} v_2}{(M, N) \downarrow^{(n_1+n_1,r_1+r_2)} (v_1, v_2)} \qquad \frac{M \downarrow^{(n_1,r_1)} (v_1, v_2) \quad N[v_1/x, v_2/y] \downarrow^{(n_2,r_2)} v}{\texttt{split}_{k'}(M,\, x.y.N) \downarrow^{(n_1+n_2,k'r_1+r_2)} v}$$

$$\frac{}{0 \downarrow^{(0,0)} 0} \qquad \frac{M \downarrow^{(n,r)} v}{S(M) \downarrow^{(n,r)} S(v)} \qquad \frac{}{(\,) \downarrow^{(0,0)} (\,)}$$

$$\frac{M \downarrow^{(n_1,r_1)} 0 \quad N_1 \downarrow^{(n_2,r_2)} \lambda x.N_1' \quad N_2 \downarrow^{(n_3,r_3)} v' \quad N_1'[(\,)/x] \downarrow^{(n_4,r_4)} v}{\texttt{nrec}\,(M, N_1, N_2) \downarrow^{(n_1+n_2+n_3+n_4,r_1+r_2+r_3+r_4)} v}$$

$$\frac{\begin{array}{l} M \downarrow^{(n_1,r_1)} S(v_1) \\ N_2 \downarrow^{(n_2,r_2)} \texttt{save}^\infty_0\ (\lambda x.N_2') \\ N_1 \downarrow^{(n_3,r_3)} \lambda x.N_1' \\ N_2'[(v_1, \lambda z.(\texttt{nrec}\,(v_1, \lambda x.N_1', \texttt{save}^\infty_0\ (\lambda x.N_2'))))/x] \downarrow^{(n_4,r_4)} v \end{array}}{\texttt{nrec}\,(M, N_1, N_2) \downarrow^{(n_1+n_2+n_3+n_4,r_1+r_2+r_3+r_4)} v}$$

$$\frac{M \downarrow^{(n_1,r_1)} [\,] \quad N_1 \downarrow^{(n_2,r_2)} \lambda x.N_1' \quad N_2 \downarrow^{(n_3,r_3)} \texttt{save}^\infty_0\ (\lambda x.N_2') \quad N_1'[(\,)/x] \downarrow^{(n_4,r_4)} v}{\texttt{lrec}\,(M, N_1, N_2) \downarrow^{(n_1+n_2+n_3+n_4,r_1+r_2+r_3+r_4)} v}$$

$$\frac{\begin{array}{l} M \downarrow^{(n_1,r_1)} v_1 :: v_2 \\ N_2 \downarrow^{(n_2,r_2)} \texttt{save}^\infty_0\ (\lambda x.N_2') \\ N_1 \downarrow^{(n_3,r_3)} \lambda x.N_1' \\ N_2'[(v_1, \langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}^\infty_0\ (\lambda x.N_2'))\rangle)/x] \downarrow^{(n_4,r_4)} v \end{array}}{\texttt{lrec}\,(M, N_1, N_2) \downarrow^{(n_1+n_2+n_3+n_4,r_1+r_2+r_3+r_4)} v}$$

FIGURE 4. λ^A Operational Semantics

k times. (The erasure to STLC discussed above runs M only once, not k times—which would be challenging when k is ∞.) Formally, this means that the n in the conclusion is just the n in the premise, but the r is multiplied by k. Running $\texttt{save}^k_\ell$ does *not* add ℓ to the r component because `save` does not create credits (adding to the amortized cost), but only attaches some already existing credits to the value v. Recall that `transfer` detaches the credits from a $!^k_\ell$ value, and allows for them, along with the k copies of the value, to be used in another term. The evaluation rule says that, in order to evaluate $\texttt{transfer}_{k'}\, !^k_\ell\, x = M \texttt{ to } N$, we first evaluate M to a `save` value, and then evaluate the substitution instance $N[v_1/x]$. The k' in `transfer` means to repeat the evaluation of M k' times, allowing $k \cdot k'$ uses in the body of N, so this (similarly to `save`) repeats the credit effects r_1 of M k' times in the conclusion. The other positive elimination forms are similar.

2.3. Syntactic Properties. In the operational semantics judgment $M \downarrow^{(n,r)} v$, we think of $n + r$ (the actual cost n plus the credit difference r) as the amortized cost of the program. A key property of amortized analysis is that the amortized cost is an upper bound on the true cost, which means in this case that $n+r \geq n$, so we would like $r \geq 0$. While r is in general allowed to be a negative number, it is controlled by the credits a of the typing judgment $\cdot \vdash_a M : A$, intuitively because it is only `spend` operations that subtract from r, and `spend` operations are only allowed when the type system deems there to be sufficient credits available. Thus, we will be able to prove that $r \geq 0$ for well-typed terms. To do so, we strengthen the induction hypotheses to prove that $\cdot \vdash_a M : A$ and $M \downarrow^{(n,r)} v$ imply $a + r \geq 0$, which gives $r \geq 0$ for closed programs that use no external credits (so $a = 0$), which is what a "main" function is expected to be (e.g. `set` in the binary counter example). It is technically convenient to combine this with a preservation result, stating that the credits of v is in fact $a + r$ (the resource term in a typing judgment must be non-negative, so $a + r \geq 0$ is in fact a prerequisite for even asserting that $\cdot \vdash_{a+r} v : A$). The proofs of the following are relatively straightforward and may be found Appendix B.

THEOREM 2.3 (Preservation Bound). *If $\cdot \vdash_a M : A$ and $M \downarrow^{(n,r)} v$, then $a + r \geq 0$ and $\cdot \vdash_{a+r} v : A$.*

We also have that values evaluate in 0 steps:

THEOREM 2.4. *If v is a value, and $v \downarrow^{(n,r)} v$, then $n = r = 0$.*

$$
\begin{aligned}
\cdot \vdash_0 \mathtt{inc} := \lambda b.\mathtt{lrec}(b, \lambda\_.\mathtt{tick}\ ;\ \mathtt{create}_1\ (\mathtt{inl}\,(\mathtt{save}_1^1\,()))\ ::\ \mathtt{[]},\\
\mathtt{save}_0^{\infty}(\lambda(a,tr).\mathtt{case}_1(a, \_.\mathtt{tick}\ ;\ \mathtt{create}_1\ (\mathtt{inl}\,(\mathtt{save}_1^1\,()))\ ::\ \pi_1 tr,\\
y.\mathtt{transfer}_1\ !_1^1\_ = y\ \mathtt{to}\\
\mathtt{spend}_1\ (\mathtt{tick}\ ;\ \mathtt{inl}\,()\ ::\ \pi_2 tr)))) : \mathtt{[bit]} \multimap \mathtt{[bit]}
\end{aligned}
$$

$$
\vdash_0 \mathtt{set} := \lambda n.\mathtt{nrec}(n, \lambda\_.\mathtt{[]}, \mathtt{save}_0^{\infty}(\lambda p.\mathtt{split}_1(p,\ \_.x.\mathtt{inc}\ (x\ ()))))) : \mathbb{N} \multimap \mathtt{[bit]}
$$

FIGURE 5. Binary Counter Terms in λ^A

and that values of type $\mathbb{N}$ contain no credits:

THEOREM 2.5 (Resource strengthening for $\mathbb{N}$). *If* $\cdot \vdash_a v : \mathbb{N}$*, then* $\cdot \vdash_0 v : \mathbb{N}$

2.4. Binary Counter Annotation. As an example, we translate the binary counter program from Section 1 to λ^A, decorating the program with `create`, `spend`, `save`, and `transfer` in order to emulate the analysis described in Section 1. Since the analysis stores credits on 1 bits, the type of bits is $\mathtt{bit} = 1 \oplus !_1^1 1$; a value $\mathtt{inl}\,()$ represents a 0 bit, and a value $\mathtt{inr}\,(\mathtt{save}_1^1\,())$ represents a 1 bit, with a credit attached. A binary number is represented as a list of bits, `[bit]`. The cost of interest is the number of bit flips, so we insert `tick`s everywhere a bit is flipped from 0 to 1 or vice versa. Next, to handle the credits, we `create` and subsequently `save` a credit when we flip a bit from 0 to 1, and `transfer` then `spend` when flipping bits from 0 to 1. This annotation is shown in Figure 5 – for simplicity, we use `inc` as a meta-level name for the term implementing the function, so its occurrence in `set` really means a copy of that entire term (to do this at the object level, we could alternatively think of a top-level definition of `inc` as binding an infinite-use variable).

3. Recurrence Language $\lambda^{\mathbb{C}}$, Amortized Recurrence Extraction, and Bounding Theorem

Next, we define a translation from λ^A into a *recurrence language* $\lambda^{\mathbb{C}}$. Unlike λ^A, $\lambda^{\mathbb{C}}$ has a fully structural (weakening and contraction) type system, and no special constructs for amortized analysis (it is mostly unchanged from [19, 37]). Further, because we view $\lambda^{\mathbb{C}}$ as a syntatx for mathematical expressions, it is designed as a call-by-name language– this is in contrast to λ^A, which is by-value. The recurrence translation takes a function in λ^A to a function that outputs the original function's cost in $\lambda^{\mathbb{C}}$, using a cost type $\mathbb{C}$ (which we will often take to be integers). Formally, $\mathbb{C}$ can be any commutative ring with an ∞ element, the typical example being the ("tropical") max-plus ring on the integers, i.e. integers with addition and binary maxes. Some of the typing rules for $\lambda^{\mathbb{C}}$ are presented in Figure 6.

Relative to our previous work, the main conceptual change for supporting amortized analysis is that, instead of extracting recurrences for the true cost of a program (n in $M \downarrow^{(n,r)} v$), we extract recurrences that given an upper bound on the program's amortized cost $n + r$, which is itself a bound on the true cost for programs which begin with an empty bank of credits.

$$\frac{}{\Gamma, x : T \vdash x : T} \qquad \frac{k \in \mathbb{Z}}{\Gamma \vdash k : \mathbb{C}} \qquad \frac{\Gamma \vdash E_1 : \mathbb{C} \quad \Gamma \vdash E_2 : \mathbb{C}}{\Gamma \vdash E_1 + E_2 : \mathbb{C}} \qquad \frac{}{\Gamma \vdash () : 1}$$

$$\frac{\Gamma \vdash E_1 : T_1 \to T_2 \quad \Gamma \vdash E_2 : T_1}{\Gamma \vdash E_1\ E_2 : T_2} \qquad \frac{\Gamma, x : T_1 \vdash E : T_2}{\Gamma \vdash \lambda x.E : T_1 \to T_2} \qquad \frac{\Gamma \vdash E_1 : T_1 \quad \Gamma \vdash E_2 : T_2}{\Gamma \vdash (E_1, E_2) : T_1 \times T_2} \qquad \frac{\Gamma \vdash E : T_1 \times T_2}{\Gamma \vdash \pi_i E : T_i}$$

$$\frac{\Gamma \vdash E : T_1}{\Gamma \vdash \mathtt{inl}\, E : T_1 + T_2} \qquad \frac{\Gamma \vdash E : T_2}{\Gamma \vdash \mathtt{inr}\, E : T_1 + T_2} \qquad \frac{\Gamma \vdash E : T_1 + T_2 \quad \Gamma, x : T_1 \vdash E_1 : T \quad \Gamma, y : T_2 \vdash E_2 : T}{\Gamma \vdash \mathtt{case}\,(E, x.E_1, y.E_2) : T}$$

$$\frac{}{\Gamma \vdash 0 : \mathbb{N}} \qquad \frac{\Gamma \vdash E : \mathbb{N}}{\Gamma \vdash S(E) : \mathbb{N}} \qquad \frac{\Gamma \vdash E : \mathbb{N} \quad \Gamma \vdash E_1 : 1 \to T \quad \Gamma \vdash E_2 : \mathbb{N} \times T \to T}{\Gamma \vdash \mathtt{nrec}\,(E, E_1, E_2) : T}$$

$$\frac{}{\Gamma \vdash \mathtt{[]} : \mathtt{[}T\mathtt{]}} \qquad \frac{\Gamma \vdash E_1 : T \quad \Gamma \vdash E_2 : \mathtt{[}T\mathtt{]}}{\Gamma \vdash E_1 :: E_2 : \mathtt{[}T\mathtt{]}} \qquad \frac{\begin{array}{l}\Gamma \vdash E : \mathtt{[}T_1\mathtt{]} \\ \Gamma \vdash E_1 : 1 \to T \\ \Gamma \vdash E_2 : T_1 \times (\mathtt{[}T_1\mathtt{]} \times T) \to T\end{array}}{\Gamma \vdash \mathtt{lrec}\,(E, E_1, E_2) : T}$$

FIGURE 6. Recurrence Language $\lambda^{\mathbb{C}}$ Definition

3.1. Monadic Translation from λ^A to $\lambda^{\mathbb{C}}$. Following [18, 19], a function $A \multimap B$ in λ^A will be translated to a function $\langle\!\langle A \rangle\!\rangle \to \mathbb{C} \times \langle\!\langle B \rangle\!\rangle$, where for a λ^A type A, a value of $\lambda^{\mathbb{C}}$ type $\langle\!\langle A \rangle\!\rangle$ represents the size of a value in λ^A. Intuitively, this means that a function in λ^A is translated to a $\lambda^{\mathbb{C}}$ function that, in terms of the size of the input, gives the cost of running the function on that argument and the size of the output. Generalized to higher-type, "size" is properly viewed as "use-cost;" it is a property that tells us how the value affects the cost of a computation that uses it. In an unfortunate terminological clash, prior work [17] refers to this concept as *potential* (as in "potential cost" or "future cost"), with no intentional connotation of potential functions from the physicist's method of amortized analysis. In order to keep this work consistent with the sequence of papers it follows, and since λ^A is based on the banker's method, we will only use "potential" to refer to the use-cost of a value, and so call $\langle\!\langle A \rangle\!\rangle$ the *potential type* for A and a value of type $\langle\!\langle A \rangle\!\rangle$ a *potential*. The size of the output is needed for the translation to be compositional: the recurrence extracted for a term should be composed of the recurrences extracted for its subterms, but the cost of e.g. a function application depends on the size of the argument itself, not just its cost. A recurrence extraction of this form can be packaged as a monadic translation into the writer monad $\mathbb{C} \times A$.

As discussed in Section 1, the proper notion of size for a specific datatype may vary from analysis to analysis. To this end, we follow [19] in deferring the abstraction of values as sizes to denotational semantics of $\lambda^{\mathbb{C}}$ defined in Section 4, which allows the same recurrence extraction and bounding theorem to be reused for multiple models with different notions of size.

We call the pair of a cost and a potential a *complexity*. The translation consists of three separate functions, the definitions of which are shown in Figure 7. Firstly, $\langle\!\langle \cdot \rangle\!\rangle$ takes a type A in λ^A and maps it to the type $\langle\!\langle A \rangle\!\rangle$ whose elements are the potentials of type A. We extend this to contexts pointwise: $\langle\!\langle \Gamma, x : A \rangle\!\rangle = \langle\!\langle \Gamma \rangle\!\rangle, x : \langle\!\langle A \rangle\!\rangle$. The second is $\|A\| := \mathbb{C} \times \langle\!\langle A \rangle\!\rangle$, which takes a type A to the corresponding type of complexities. Finally, we overload $\|\cdot\|$ to denote the recurrence extraction function from terms of λ^A to terms in $\lambda^{\mathbb{C}}$. For convenience, when $E : \mathbb{C} \times T$, we often write $\pi_1 E$ as E_c (cost) and $\pi_2 E$ as E_p (potential). [3] We also use special notation for adding a cost to a complexity, writing $E +_c E'$ for $(E + E'_c, E'_p)$ when $E : \mathbb{C}$ and $E' : \mathbb{C} \times T$.

Overall, the idea is that a term is translated to a function from potentials of its context to complexities of its type:

THEOREM 3.1 (Extraction Preserves Types). *If* $\Gamma \vdash_a M : A$ *then* $\langle\!\langle \Gamma \rangle\!\rangle \vdash \|M\| : \|A\|$

We comment on some of the less obvious aspects of this translation:

- $!^k_\ell A$: The type translation erases the $!^k_\ell$ modality.

[3]We regard the subscript notation as binding tighter than ordinary projection: i.e. $\pi_1 E_p = \pi_1(E_p)$.

$$\|A\| = \mathbb{C} \times \langle\!\langle A \rangle\!\rangle$$

$$\langle\!\langle 1 \rangle\!\rangle = 1 \qquad \langle\!\langle A \otimes B \rangle\!\rangle = \langle\!\langle A \rangle\!\rangle \times \langle\!\langle B \rangle\!\rangle \qquad \langle\!\langle A \oplus B \rangle\!\rangle = \langle\!\langle A \rangle\!\rangle + \langle\!\langle B \rangle\!\rangle \qquad \langle\!\langle A \multimap B \rangle\!\rangle = \langle\!\langle A \rangle\!\rangle \to \|B\|$$

$$\left\langle\!\!\left\langle !^k_\ell A \right\rangle\!\!\right\rangle = \langle\!\langle A \rangle\!\rangle \qquad \langle\!\langle A \& B \rangle\!\rangle = \|A\| \times \|B\|$$

$$\langle\!\langle \mathbb{N} \rangle\!\rangle = \mathbb{N} \qquad \langle\!\langle \texttt{[}A\texttt{]} \rangle\!\rangle = \texttt{[}\,\langle\!\langle A \rangle\!\rangle\,\texttt{]}$$

$$\|x\| = (0, x)$$

$$\|()\| = (0, ()) \qquad \|(M, N)\| = (\|M\|_c + \|N\|_c, (\|M\|_p, \|N\|_p)) \qquad \|\pi_i M\| = \|M\|_c +_c \pi_i\left(\|M\|_p\right)$$

$$\|\texttt{inl}\, M\| = (\|M\|_c, \texttt{inl}\ \|M\|_p) \qquad \|\texttt{inr}\, M\| = (\|M\|_c, \texttt{inr}\ \|M\|_p)$$

$$\|\texttt{case}_{k'}\,(M,\, x.N_1\,,\, y.N_2)\| = k'\,\|M\|_c + \texttt{case}\,(\|M\|_p\,,\, x.\,\|N_1\|\,,\, y.\,\|N_2\|)$$

$$\|\lambda x.M\| = (0, \lambda x.\,\|M\|) \qquad \|M\ N\| = (\|M\|_c + \|N\|_c) +_c \|M\|_p\ \|N\|_p$$

$$\|\langle M, N\rangle\| = (0, (\|M\|\,, \|N\|))$$

$$\|\texttt{split}_{k'}(M,\, x.y.N)\| = k'\,\|M\|_c +_c \|N\|\,[\pi_1\,\|M\|_p\,/x, \pi_2\,\|M\|_p\,/y]$$

$$\|0\| = (0, 0) \qquad \|S(M)\| = (\|M\|_c\,, S(\|M\|_p))$$

$$\|\texttt{[]}\| = (0, \texttt{[]}) \qquad \|M :: N\| = (\|M\|_c + \|N\|_c\,, \|M\|_p :: \|N\|_p)$$

$$\|\texttt{tick}\ ;\ M\| = 1 +_c \|M\|$$

$$\left\|\texttt{transfer}_{k'}\ !^k_\ell\ x = M\ \texttt{to}\ N\right\| = k'\,\|M\|_c +_c \|N\|\,[\|M\|_p\,/x] \qquad \left\|\texttt{save}^k_\ell\ M\right\| = (k\,\|M\|_c\,, \|M\|_p)$$

$$\|\texttt{create}_\ell\ M\| = \ell +_c \|M\| \qquad \|\texttt{spend}_\ell\ M\| = (-\ell) +_c \|M\|$$

$$\|\texttt{nrec}\,(M, N_1, N_2)\| = (\|M\|_c + \|N_1\|_c + \|N_2\|_c) +_c \texttt{nrec}\left(\|M\|_p\,, \|N_1\|_p\,, \lambda x.\,\|N_2\|_p\,(\pi_1 x, \lambda z.\pi_2 x)\right)$$

$$\begin{aligned}\|\texttt{lrec}\,(M, N_1, N_2)\| = {} & (\|M\|_c + \|N_1\|_c + \|N_2\|_c) +_c \\ & \texttt{lrec}\left(\|M\|_p\,, \|N_1\|_p\,, \lambda x.\,\|N_2\|_p\,(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x))\right)\end{aligned}$$

FIGURE 7. Recurrence Extraction

- $A\&B$: Since the negative product in λ^A is lazy, a value of type $A\&B$ is a pair of un-evaluated terms. Thus, the potential of a term of type $A\&B$ must include the cost of evaluating each term, since that will factor into the cost of using such a value.

- `tick`: Since `tick ;` M evaluates with (true cost and) amortized cost 1 higher than M's, the cost component of $\|\texttt{tick} ; M\|$ is $1 + \|M\|_c$.
- $\texttt{save}^k_\ell$: The extracted amortized cost of $\texttt{save}^k_\ell\ M$ is k times the extracted cost of M, with the potential remaining the same. This is in principle a non-exact bound, because we are conceptually multiplying the operational amortized cost of $M \downarrow^{(n,r)} v$, which is $n + r$, by k, whereas the operational semantics gives the more precise $n + kr$. We view this as a consequence of the fact that amortized analyses extract recurrences for the amortized cost $n + r$, rather than n and r separately. However, this inflation is not a problem for our uses of $!^\infty$ in typing recursors because the branches of the recursor are usually values, which have 0 cost, and $\infty \times 0 = 0$. In future work, we might consider a recurrence translation into the $\mathbb{C} \times \mathbb{C} \times A$ monad, with separate extractions of n and r, if more precision is needed. This would allow for λ^A to be used in the place of the (linear fragment) of the source language in previous work [19]. Embedding that language into the $!^\infty$ fragment of λ^A and then extracting recurrences into $\mathbb{C} \times \mathbb{C} \times A$ would yield the same results as applying the non-amortized recurrence extraction. We emphasize that the loss of precision from not making this change has no bearing on *amortized* algorithm analyses, it would only allow for *non-amortized* analyses to also be performed with λ^A– but such analyses are already handled by prior work [19, 40]
- `transfer`: A similar imprecision arises with respect to the multiplicity k' here, but otherwise `transfer` is translated like a `let`.
- `nrec`: As in the operational semantics, because we think of the recursor as a call-by-value function constant, some cost is in principle incurred for evaluating the branches to function values, though the branches are usually values in practice.
- `lrec`: The type of the step function in a list recursor is $!^\infty_0(A \otimes ([A] \& C) \multimap C)$, and the potential translation of this type is $\langle\!\langle A\rangle\!\rangle \times ((\mathbb{C} \times [\langle\!\langle A\rangle\!\rangle]) \times (\mathbb{C} \times \langle\!\langle C\rangle\!\rangle)) \to \mathbb{C} \times \langle\!\langle C\rangle\!\rangle$. However, this does not match the required type of the step function of the list recursor in $\lambda^{\mathbb{C}}$, which must be $T_1 \times ([T_1] \times T_2) \to T_2$. Taking $T_1 = \langle\!\langle A\rangle\!\rangle$ and $T_2 = \mathbb{C} \times \langle\!\langle C\rangle\!\rangle$, the translation of the step function additionally requires a $\mathbb{C}$ input representing the cost of the tail of the list. However, lists are eager, so the step function is always applied to a value, so we can supply 0 cost here.

$$\mathcal{C} ::= [\,] \mid \pi_0\mathcal{C} \mid \pi_1\mathcal{C} \mid \mathcal{C}\ E \mid \texttt{case}\,(\mathcal{C},\, x.E\,,\, y.E') \mid \texttt{nrec}\,(\mathcal{C}, E_1, E_2) \mid \texttt{lrec}\,(\mathcal{C}, E_1, E_2)$$

$$\frac{\Gamma, x : T' \vdash \mathcal{C}[x] : T \quad \Gamma \vdash E_0 \leq_{T'} E_1}{\Gamma \vdash \mathcal{C}[E_0] \leq_T \mathcal{C}[E_1]} \qquad \frac{}{\Gamma \vdash E \leq_T E} \qquad \frac{\Gamma \vdash E_1 \leq_T E_2 \quad \Gamma \vdash E_2 \leq_T E_3}{\Gamma \vdash E_1 \leq_T E_3}$$

$$\frac{}{\Gamma \vdash E_1[E/x] \leq_T \texttt{case}\,(\texttt{inl}\,E,\, x.E_1\,,\, y.E_2)} \qquad \frac{}{\Gamma \vdash E_2[E/x] \leq_T \texttt{case}\,(\texttt{inr}\,E,\, x.E_1\,,\, y.E_2)}$$

$$\frac{}{\Gamma \vdash E[E'/x] \leq_T (\lambda x.E)\ E'} \qquad \frac{}{\Gamma \vdash E_i \leq_{T_i} \pi_i(E_1, E_2)}$$

$$\frac{}{\Gamma \vdash E_1\ () \leq_T \texttt{nrec}\,(0, E_1, E_2)} \qquad \frac{}{\Gamma \vdash E_2\ (E, \texttt{nrec}\,(E, E_1, E_2)) \leq_T \texttt{nrec}\,(S(E), E_1, E_2)}$$

$$\frac{}{\Gamma \vdash E_1\ () \leq_T \texttt{lrec}\,(\texttt{[]}, E_1, E_2)} \qquad \frac{}{\Gamma \vdash E_2\ (E, (E', \texttt{lrec}\,(E', E_1, E_2))) \leq_T \texttt{lrec}\,(E :: E', E_1, E_2)}$$

FIGURE 8. Syntactic Ordering on $\lambda^{\mathbb{C}}$

3.2. Recurrence Language Inequality Judgment. $\lambda^{\mathbb{C}}$ has a syntactic inequality judgment $\Gamma \vdash E_1 \leq_T E_2$ (Figure 8), which intuitively means that the recurrence E_1 is bounded above by E_2. For now, we include only those inequalities that are necessary to prove the bounding theorem; this allows for the most models of the recurrence language, and additional axioms valid in particular models can be added in order to simplify recurrences syntactically. The necessary axioms are congruence in the principal positions of elimination forms, as well as the fact that β-reducts are bounded above by their redexes. We often omit the context and type subscript from $\Gamma \vdash E_1 \leq_T E_2$, writing $E_1 \leq_T E_2$ or $E_1 \leq E_2$, though formally it is a relation on well-typed terms in context. This relation is primarily a technical device to provide closure properties for the bounding relation. Because of this, we omit a more lengthy discussion of the relation here, and refer the reader to the prior work [19] which introduces this type of relation.

3.3. Bounding Relation and Its Closure Properties. The correctness of the recurrence extraction is stated in terms of a logical relation between terms in λ^A and terms in $\lambda^{\mathbb{C}}$. The intended meaning is that the $\lambda^{\mathbb{C}}$ recurrence term is an upper bound on the λ^A term's cost and potential.

DEFINITION 3.1 (Bounding Relation). *When $\cdot \vdash_a M : A$ and $\cdot \vdash E : \|A\|$, then $M \sqsubseteq^{A,a} E$ if and only if, when $M \downarrow^{(n,r)} v$,*

- $n \leq E_c - r$
- $v \sqsubseteq^{A,a+r}_{val} E_p$

When $\cdot \vdash_a v : A$ and $\cdot \vdash E : \langle\!\langle A \rangle\!\rangle$, we define $v \sqsubseteq_{val}^{A,a} E$ by induction on A.

- $\mathbf{save}_\ell^k\ v \sqsubseteq_{val}^{!_\ell^k A,c} E$ *if there exists $d \geq 0$ so that $kd + \ell \leq c$, and $v \sqsubseteq_{val}^{A,d} E$*
- $\lambda x.M \sqsubseteq_{val}^{A \multimap B,c} E$ *if whenever $v \sqsubseteq_{val}^{A,d} E'$, we have that $M[v/x] \sqsubseteq^{B,c+d} E\ E'$*
- $(v_1, v_2) \sqsubseteq_{val}^{A_1 \otimes A_2,a} E$ *if there are a_1, a_2 such that $a_1 + a_2 = a$ and $v_i \sqsubseteq_{val}^{A_i,a_i} \pi_i E$ for $i \in \{1,2\}$*
- $[] \sqsubseteq_{val}^{[A],a} E$ *iff $[] \leq_{[\langle\!\langle A \rangle\!\rangle]} E$*
- $v_1 :: v_2 \sqsubseteq_{val}^{[A],a} E$ *iff there are E_1, E_2 with $E_1 :: E_2 \leq_{[\langle\!\langle A \rangle\!\rangle]} E$, and there are a_1, a_2 such that $a_1 + a_2 = a$ such that $v_1 \sqsubseteq_{val}^{A,a_1} E_1$ and $v_2 \sqsubseteq_{val}^{[A],a_2} E_2$.*
- $0 \sqsubseteq_{val}^{\mathbb{N},a} E$ *iff $0 \leq E$*
- $S(v) \sqsubseteq_{val}^{\mathbb{N},a} E$ *iff there is some E' such that $S(E') \leq_{\mathbb{N}} E$, and $v \sqsubseteq_{val}^{\mathbb{N},a} E'$*
- $\mathbf{inl}\, v \sqsubseteq_{val}^{A \oplus B,a} E$ *if there exists E' such that $\mathbf{inl}\, E' \leq_{\langle\!\langle A \rangle\!\rangle} E$ and $v \sqsubseteq_{val}^{A,a} E'$.*
- $\mathbf{inr}\, v \sqsubseteq_{val}^{A \oplus B,a} E$ *if there exists E' such that $\mathbf{inr}\, E' \leq_{\langle\!\langle B \rangle\!\rangle} E$ and $v \sqsubseteq_{val}^{B,a} E'$.*
- $() \sqsubseteq_{val}^{1,a} E$ *if $() \leq_1 E$.*
- $\langle M, N \rangle \sqsubseteq_{val}^{A \& B,a} E$ *if $M \sqsubseteq^{A,a} \pi_1 E$, and $N \sqsubseteq^{B,a} \pi_2 E$.*

We extend the value bounding relation to substitutions pointwise: $\theta \sqsubseteq_{sub}^{\Gamma,\sigma} \Theta$ if for all $x : A \in \Gamma$, $\theta(x) \sqsubseteq_{val}^{A,\sigma(x)} \Theta(x)$. Finally, we define the bounding relation for open terms: when $\Gamma \vdash_f M : A$, we say that $M \sqsubseteq E$ if for all $\theta \sqsubseteq_{sub}^{\Gamma,\sigma} \Theta$, we have $M[\theta] \sqsubseteq^{A,f[\sigma]} E[\Theta]$.

The *term/expression bounding relation* $M \sqsubseteq^{A,a} E$ says first that the cost component of E is an upper bound on the amortized cost of M, which is $n + r \leq E_c$ (since we will eventually be interested in bounding the actual cost of evaluating M, we write this as $n \leq E_c - r$). Additionally, expression bounding says that the potential component of E is an "upper bound" on the value that M evaluates to; this is expressed via a mutually-defined type-varying *value bounding relation* $M \sqsubseteq_{\mathtt{val}}^{A,a} E$. The value bounding relation is defined first by induction on the type A, and the cases for natural numbers and lists have a local induction on the number/list value as well.[4] We write the credit bank a as a parameter of the bounding relations, but it is a presupposition that this number is the same one that was used to type check $\cdot \vdash_a \{M, v\} : A$ (because the bounding relation is on closed terms, the resource subscript is just a single number a).

[4]In general, it is necessary to define the relations for inductive types inductively [19], but the values of $\mathbb{N}$ and $[A]$ are simple enough that induction on values suffices here.

We extend the bounding relation to open terms by considering all closing substitutions: a term $\Gamma \vdash_f M : A$ is bounded by E if for every substitution θ which is bounded pointwise by Θ with some credit function σ, then the closed term $M[\theta]$ is bounded by $E[\Theta]$ with $f[\sigma]$ credits. In this definition, σ gives a number of credits a_i for each variable x_i, because θ is a substitution of closed terms for variables $(\cdot \vdash_{a_1} v_1 : A_1)/x_1, (\cdot \vdash_{a_2} v_2 : A_2)/x_2, \ldots$.

3.4. Bounding Theorem. As usual for a logical relation, we first require some lemmas about the bounding relation, before a main loop proving the fundamental theorem that terms are related to their extractions. The proofs of the following theorems can be found in Appendix B.

First, we have an analogue of Theorem 2.5:

THEOREM 3.2 ($\mathbb{N}$-strengthening). *For all $\cdot \vdash_a v : \mathbb{N}$, if $v \sqsubseteq_{\mathtt{val}}^{\mathbb{N},a} E$, then $v \sqsubseteq_{\mathtt{val}}^{\mathbb{N},0} E$.*

Second, we can weaken a bound by recurrence language inequality:

THEOREM 3.3 (Weakening).

(1) If $M \sqsubseteq^{A,a} E$, and $E \leq_{\|A\|} E'$, then $M \sqsubseteq^{A,a} E'$

(2) If $v \sqsubseteq_{\mathtt{val}}^{A,a} E$, and $E \leq_{\langle\!\langle A \rangle\!\rangle} E'$, then $v \sqsubseteq_{\mathtt{val}}^{A,a} E'$

Next, we have an analogue of resource weakening in Theorem 2.1:

THEOREM 3.4 (Credit Weakening). *If $a_1 \leq a_2$, then:*

(1) If $M \sqsubseteq^{A,a_1} E$, then $M \sqsubseteq^{A,a_2} E$

(2) If $v \sqsubseteq_{\mathtt{val}}^{A,a_1} E$, then $v \sqsubseteq_{\mathtt{val}}^{A,a_2} E$

Next, we have inductive lemmas that will be used in the recursor cases of the fundamental theorem:

THEOREM 3.5 ($\mathbb{N}$-Recursor). *If $\lambda x.N_1' \sqsubseteq_{\mathtt{val}}^{1\multimap C,c_3} E_1$, $\lambda x.N_2' \sqsubseteq_{\mathtt{val}}^{\mathbb{N}\otimes(1\multimap C)\multimap C,d} E_2$ with $d \geq 0$, then $\forall n \geq 0$, if $\overline{n} \sqsubseteq_{\mathtt{val}}^{\mathbb{N},0} E$, then $\mathtt{nrec}(\overline{n}, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')) \sqsubseteq^{C,c_3+\infty\cdot d} \mathtt{nrec}(E, E_1, \lambda p.E_2\ (\pi_1 p, \lambda z.\pi_2 p))$*

THEOREM 3.6 ($\mathtt{[}A\mathtt{]}$-Recursor). *If $\lambda x.N_1' \sqsubseteq_{\mathtt{val}}^{1\multimap C,c_1} E_1$ and $\lambda x.N_2' \sqsubseteq_{\mathtt{val}}^{A\otimes(\mathtt{[}A\mathtt{]}\&C)\multimap C,c_2} E_2$, then for all values $\cdot \vdash_d v : \mathtt{[}A\mathtt{]}$ such that $v \sqsubseteq_{\mathtt{val}}^{\mathtt{[}A\mathtt{]},d} E$, we have that*

$\mathtt{lrec}(v, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')) \sqsubseteq^{C,c_1+d+\infty\cdot c_2} \mathtt{lrec}(E, E_1, \lambda x.E_2(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x)))$

$$\begin{aligned}\cdot \vdash \|\texttt{inc}\| := (0, \lambda bs.\texttt{lrec}(bs, \lambda\_.(2, (\texttt{inr}\,()) :: \texttt{[]}),\\ \lambda p.(\lambda x.\texttt{case}(\pi_1 x,\\ \_.(2 + (\pi_1\pi_2 x)_c, (\texttt{inr}\,()) :: (\pi_1\pi_2 x)_p)\\ \_.((\pi_2\pi_2 x)_c, (\texttt{inl}\,()) :: (\pi_2\pi_2 x)_p))\\ )(\pi_1 p, ((0, \pi_1\pi_2 p), \pi_2\pi_2 p)))) : \mathbb{C} \times (\,\texttt{[}1+1\texttt{]} \to \mathbb{C} \times \texttt{[}1+1\texttt{]}\end{aligned}$$

$$\begin{aligned}\cdot \vdash \|\texttt{set}\| := (0, \lambda n.\texttt{nrec}(n, \lambda\_.(0, \texttt{[]}), \lambda u.(0, \lambda p.(\pi_2 p\,())_c +_c \|\texttt{inc}\|_p\,(\pi_2 p\,())_p)_p\\ (\pi_1 u, \lambda\_.\pi_2 u))) : \mathbb{C} \times (\mathbb{N} \to \mathbb{C} \times \texttt{[}1+1\texttt{]}\,)\end{aligned}$$

FIGURE 9. Binary Counter Recurrences in $\lambda^{\mathbb{C}}$

Using these, we prove the main result:

THEOREM 3.7 (Bounding Theorem). *If* $\Gamma \vdash_f M : A$, *then* $M \sqsubseteq^A \|M\|$

Finally, for terms that use no external credits, the true cost is bounded by the extracted recurrence:

COROLLARY 3.1 (True cost bounding). *If* $\cdot \vdash_0 M : A$ *and* $M \downarrow^{(n,r)} v$ *then* $n \leq \|M\|_c$.

PROOF. By Theorem 3.7, we have $n \leq \|M\|_c - r$, but by preservation (Theorem 2.3), we have that $0 + r \geq 0$, so $n \leq \|M\|_c$. □

3.5. Binary Counter Recurrences. As an example, the binary counter program in λ^A (Figure 5) is translated by the recurrence extraction translation to the terms in Figure 9. Next, we will use a denotational semantics of the recurrence language to simplify these recurrences to the desired closed form.

4. Recurrence Language Semantics

The final step of our technique is to simplify recurrences to closed forms. This can be done semantically, in a denotational model of the recurrence languages, or syntactically, by adding axioms to the inequality judgment $\Gamma \vdash E \leq_T E'$ corresponding to properties true in a particular model. Here, we will work in a denotational model of $\lambda^{\mathbb{C}}$ in preorders, which mostly follows previous work [18, 19, 37].

4.1. Semantic Interpretation. We describe the semantic interpretation of $\lambda^{\mathbb{C}}$ in preorders here, and highlight the differences from [37], which gives a similar presentation with mechanized proofs.

The semantics of types and terms is given in Figure 10, omitting function and product types, which are interpreted using the standard cartesian product and exponential objects of preorders. For each type A of $\lambda^{\mathbb{C}}$, we associate a partially ordered set $[\![A]\!]$ equipped with a top element (∞) and binary maximums ($\vee$) for which the top element is an annihilator. We write 1 for the one-element poset, and $\mathbb{N} \cup \infty$ for the natural numbers with an infinite element added, with the usual $0 \leq 1 \leq 2 \leq \ldots \leq \infty$ total order, and $\mathbb{Z} \cup \infty$ for the integers with an infinite element added, with the usual total order. We write $P \times Q$ for the cartesian product of posets with the pointwise order, and Q^P for the poset of monotone functions from P to Q, ordered pointwise; these have binary maxes and top elements given pointwise. We write $P{+}Q/{\sim}$ for the "coalesced" sum, which first takes the disjoint union of P and Q, with only $\texttt{inl}(x) \leq \texttt{inl}(y)$ if $x \leq_P y$ and similarly for $\texttt{inr}$, and then equates $\texttt{inl}(\infty_P)$ and $\texttt{inr}(\infty_Q)$ to create a top element $\infty_{P+Q/\sim}$; binary maxes are defined using maxes in P and Q for two elements whose injections match, and to be ∞ otherwise. The translation on types is extended to contexts: $[\![\cdot]\!] = 1$, $[\![\Gamma, x : A]\!] = [\![\Gamma]\!] \times [\![A]\!]$. Finally, we interpret terms of $\lambda^{\mathbb{C}}$ as *monotone* (but not necessarily infinity- or max-preserving) maps[5] from the interpretation of their contexts into the interpretation of their types. These maps are morphisms in the category **Poset** of partially ordered sets and monotone maps, and so we write them as elements of $\mathrm{Hom}_{\mathbf{Poset}}(A, B)$, the set of monotone maps between posets A and B.

In Figure 10, we show some representative cases of the interpretation of terms for sums, natural numbers and lists. For costs, the interpretation of cost constants and addition uses the elements and addition of $\mathbb{Z} \cup \infty$. In this model, we interpret both natural numbers and lists as $\mathbb{N} \cup \infty$; for lists, this interprets a list as its length. $\mathbb{N} \cup \infty$ has a 0 element and a monotone successor function S, where $S(\infty) = \infty$; these are used to interpret 0/the empty list and successor/cons. The elimination forms for positives are more complex, and use some auxiliary monotone functions (which are the morphisms in the category of posets):

THEOREM 4.1. *For any posets A, B, C, G with ∞ and $\vee$,*

[5] We write the composition of maps $f : A \to B$ and $g : B \to C$ in diagrammatic order, $f ; g : A \to C$.

$[\![\mathbb{C}]\!] = \mathbb{Z} \cup \{\infty\}$

$[\![\mathbb{N}]\!] = \mathbb{N} \cup \{\infty\}$

$[\![\texttt{[}T\texttt{]}]\!] = \mathbb{N} \cup \{\infty\}$

$[\![T_1 + T_2]\!] = ([\![T_1]\!] + [\![T_2]\!]) \,/\, {\sim}$ where $\texttt{inl}\,\infty \sim \texttt{inr}\,\infty$

$[\![\Gamma, x : T, \Gamma' \vdash x : T]\!] = \pi_1^k ; \pi_2$ where $|\Gamma'| = k$

$[\![\Gamma \vdash k : \mathbb{C}]\!] = \mathrm{const}\,(k)$

$[\![\Gamma \vdash E_1 + E_2 : \mathbb{C}]\!] = ([\![\Gamma \vdash E_1 : \mathbb{C}]\!], [\![\Gamma \vdash E_2 : \mathbb{C}]\!]); +$

$[\![\Gamma \vdash () : 1]\!] = \mathrm{const}\,(())$

$[\![\Gamma \vdash \texttt{inl}\, E : T_1 + T_2]\!] = [\![\Gamma \vdash E : T_1]\!]; \texttt{inl}$

$[\![\Gamma \vdash \texttt{inr}\, E : E_1 + E_2]\!] = [\![\Gamma \vdash E : E_2]\!]; \texttt{inr}$

$[\![\Gamma \vdash \texttt{case}\,(E,\, x.E_1\,,\, y.E_2) : T]\!] = \left(1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E : T_1 + T_2]\!]\right); \texttt{scase}([\![\Gamma, x : T_1 \vdash E_1 : T]\!], [\![\Gamma, y : T_2 \vdash E_2 : T]\!])$

$\texttt{scase} \in \mathrm{Hom}_{Poset}\left(C^{G\times A} \times C^{G\times B}, C^{G\times(A+B)}\right)$

$\texttt{scase}(f,g)(\gamma, \texttt{inl}\, a) = f(\gamma, a) \vee g(\gamma, \infty)$

$\texttt{scase}(f,g)(\gamma, \texttt{inr}\, b) = f(\gamma, \infty) \vee g(\gamma, b)$

$[\![\Gamma \vdash 0 : \mathbb{N}]\!] = \mathrm{const}\,(0)$

$[\![\Gamma \vdash S(M) : \mathbb{N}]\!] = [\![\Gamma \vdash M : \mathbb{N}]\!]; S$

$[\![\Gamma \vdash \texttt{nrec}\,(E, E_1, E_2) : T]\!] = \left(1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E : \mathbb{N}]\!]\right); \texttt{snrec}([\![\Gamma \vdash E_1 : 1 \to T]\!], [\![\Gamma \vdash E_2 : \mathbb{N} \times T \to T]\!])$

$\texttt{snrec} \in \mathrm{Hom}_{Poset}\left(\left(C^1\right)^G \times \left(C^{\mathbb{N}\times C}\right)^G, C^{G\times\mathbb{N}}\right)$

$\texttt{snrec}(f,g)(\gamma, 0) = f(\gamma)()$

$\texttt{snrec}(f,g)(\gamma, n+1) = g(\gamma)(n, \texttt{snrec}(f,g)(\gamma, n)) \vee f(\gamma)()$

$[\![\Gamma \vdash \texttt{[]} : \texttt{[}A\texttt{]}]\!] = \mathrm{const}\,(0)$

$[\![\Gamma \vdash E_1 :: E_2 : \texttt{[}A\texttt{]}]\!] = [\![\Gamma \vdash E_2 : \texttt{[}A\texttt{]}]\!]; S$

$[\![\Gamma \vdash \texttt{lrec}\,(E, E_1, E_2) : T]\!] = \left(1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E : \texttt{[}T\texttt{]}']\!]\right);$

$\qquad\qquad\qquad\texttt{slrec}([\![\Gamma \vdash E_1 : 1 \to T]\!], [\![\Gamma \vdash E_2 : T' \times (\texttt{[}T\texttt{]}' \times T) \to T]\!])$

$\texttt{slrec} \in \mathrm{Hom}_{Poset}\left(\left(C^1\right)^G \times \left(C^{A\times(\mathbb{N}\times C)}\right)^G, C^{G\times\mathbb{N}}\right)$

$\texttt{slrec}(f,g)(\gamma, 0) = f(\gamma)()$

$\texttt{slrec}(f,g)(\gamma, n+1) = g(\gamma)(\infty, (n, \texttt{slrec}(f,g)(\gamma, n))) \vee f(\gamma)()$

FIGURE 10. Semantic Interpretation Definition

(1) $\texttt{snrec} \in \mathrm{Hom}_{Poset}\left(\left(C^1\right)^G \times \left(C^{\mathbb{N}\times C}\right)^G, C^{G\times\mathbb{N}}\right)$
(2) $\texttt{slrec} \in \mathrm{Hom}_{Poset}\left(\left(C^1\right)^G \times \left(C^{A\times(\mathbb{N}\times C)}\right)^G, C^{G\times\mathbb{N}}\right)$
(3) $\texttt{scase} \in \mathrm{Hom}_{Poset}\left(C^{G\times A} \times C^{G\times B}, C^{G\times(A+B)}\right)$

The definition of `scase` is required to respect the quotienting $\texttt{inl}(\infty) = \texttt{inr}(\infty)$; by maxing each branch the image of ∞ from the other branch, we obtain $f(\gamma,\infty) \vee g(\gamma,\infty)$ as the image of both of those. The definition of `snrec` is required to be monotone in the $0 \le 1 \le \ldots \le \infty$ ordering; taking the maximum of the base case and the inductive step achieves this, because it forces the image of 1 to dominate the image of 0. The definition of `slrec` is similar; the new question that arises is that, because we have abstracted lists as their lengths, forgetting the elements, we do not have a value for the head of the list to supply to g (which, when we use this operation, will be the translation of the cons branch given to the $\lambda^{\mathbb{C}}$ recursor). Here, we always supply ∞ as the head list element, which is sufficient when the analysis really does not require any information about the elements of the list (otherwise, one can make a model where lists are interpreted more precisely than as their lengths [19, 15]).

The interpretation satisfies standard soundness theorems, the proofs of which are in Appendix B.

THEOREM 4.2 (Compositionality). *If* $\Gamma, x : T_1 \vdash E : T_2$, *and* $\Gamma \vdash E' : T_1$, *then* $[\![\Gamma \vdash E[E'/x] : T_2]\!] = \left(1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E' : T_1]\!]\right); [\![\Gamma, x : T_1 \vdash E : T_2]\!]$

THEOREM 4.3 (Soundness (Terms)). *If* $\Gamma \vdash E : T$, *then* $[\![\Gamma \vdash E : T]\!] \in \mathrm{Hom}\left([\![\Gamma]\!], [\![T]\!]\right)$

THEOREM 4.4 (Soundness (Inequality)). *If* $\Gamma \vdash E \le E'$, *then for all* $\gamma \in [\![\Gamma]\!]$, $[\![\Gamma \vdash E : T]\!](\gamma) \le [\![\Gamma \vdash E' : T]\!](\gamma)$

4.2. Binary Counter Conclusion. We interpret the binary counter recurrences from Figure 9 in preorders by unfolding the definitions in Figure 10; the result is shown in Figure 11. For the function `inc`, this yields a monotone map $[\![\|\texttt{inc}\|_p]\!] \in \mathrm{Hom}(1, \mathbb{N} \to \mathbb{Z} \times \mathbb{N})$, which is (essentially) a function from an input list size to the cost of evaluation and the length of the output. For the function `set`, this yields a monotone map $[\![\|\texttt{set}\|]\!] \in \mathrm{Hom}(1, \mathbb{Z} \times (\mathbb{N} \to \mathbb{Z} \times \mathbb{N}))$,

$$
\begin{aligned}
[\![\|\texttt{inc}\|_p]\!] = \lambda\gamma.\lambda bs.\texttt{slrec}(&\lambda\gamma.\lambda z.(\boxed{2}, 1),\\
&\lambda\gamma.\lambda p.\texttt{case}(\lambda x.(\boxed{2}, 1 + \pi_1\pi_2 p),\\
&\qquad\qquad \lambda x.(\boxed{\pi_1\pi_2\pi_2 p}, 1 + \pi_2\pi_2\pi_2 p)\\
&\qquad\qquad)(\gamma', \pi_1\pi_1 p)\\
&\qquad \text{where } \gamma' = ((\gamma, p), (\pi_1 p, ((0, \pi_1\pi_2 p), \pi_2\pi_2 p)))\\
&)(\gamma, bs)\\
[\![\|\texttt{set}\|_p]\!] = \lambda\gamma.(0, \lambda n.\texttt{snrec}(&\lambda\gamma'.\lambda x.(\boxed{0}, 0)\\
&\lambda\gamma.\lambda p.(\boxed{\pi_1\pi_2 p + \pi_2([\![\|inc\|_p]\!]()(\pi_2\pi_2 p))}, \pi_2([\![\|inc\|_p]\!]()(\pi_2\pi_2 p)))))
\end{aligned}
$$

FIGURE 11. Binary Counter Recurrences Interpreted

which is a pair of a cost (the cost of evaluating the function definition — 0 since set is a value) and a function from input size to the cost of evaluation and the length of the output.

We have boxed the parts of the term that are related to computing the cost. The boxed portions of inc express that its amortized cost is 2 on the empty list (to create a 1 bit with a credit), is 2 when the bit is 0, and is exactly the same number of steps as the recursive call when the bit is 1. The boxed portions of set express that for zero it costs 0, and for successor it costs the recursive call plus the cost of inc on the potential of the output of the recursive call. However, because we will show that inc turns out to be constant amortized time, we do not need to bound the potential of the output of set. Intuitively, to see that inc has constant amortized time, observe that the slrec will always supply the ∞ bit as the head of the list, which by definition of the coalesced sum is both true and false, so the case is effectively the maximum of 2 and $\pi_1\pi_2\pi_1 p$. Thus, we effectively have recurrence where $T_{\texttt{inc}}(0) = 2$ and $T_{\texttt{inc}}(n) = 2 \vee T_{\texttt{inc}}(n-1)$, which solves to $T(n) = 2$ by induction. Substituting this into the recurrence for set, we have essentially $T_{\texttt{set}}(0) = 0$ and $T_{\texttt{set}}(n) = T_{\texttt{set}}(n-1) + 2$, which is of course $O(n)$. More formally, we can show by induction that for all $n \geq 0$, $([\![\|\texttt{inc}\|_p]\!]()(n))_c \leq 2$, and that for all n, $([\![\|\texttt{set}\|_p]\!]()(n))_c \leq 2n$, establishing bounds on these recurrences in this denotational semantics in preorders.

By the bounding theorem (Corollary 3.1), we have that, for the true operational cost m of evaluating $\texttt{set}(n) \downarrow^{(m,r)} v$, we have $m \leq_{\mathbb{C}} \|\texttt{set}\|_p(n)_c$ in terms of the syntactic preorder judgment in $\lambda^{\mathbb{C}}$. By the soundness of the interpretation in preorders (Theorem 4.4), we have that $m \leq_{\mathbb{Z}\sqcup\infty} [\![\|\mathtt{set}\|_p]\!]()(n)_c$ in the preorder model. Therefore, by transitivity, we have $m \leq 2n$ in the preorder model, so our technique proves that the true operational cost m of setting the binary counter to n is in fact $O(n)$, as desired.

5. Variable-Credit Extension

The version of λ^A described thus far supports amortized analyses where the amount of credit stored on each element of a data structure is fixed (e.g. $\mathtt{[!_2}A\mathtt{]}$ is a list with 2 credits on each element). However, in some important amortized analyses, different amounts of credit must be stored in different parts of a data structure—e.g. for balanced binary search trees implemented via splay trees [72], the number of credits stored on each node is a function of the size of the subtree rooted at that node. In this section, we show that adding existential quantification over credit amounts to λ^A suffices to analyze such examples, using a portion of splay trees as an example. Using existentials, a value of type $\exists\alpha.!_\alpha A$ is a value of type A which carries α credits, for some α; for example, a tree whose elements are of type $\exists\alpha.!_\alpha\mathbb{N}$ stores a variable number of credits with the number on each node. In keeping with our methodology of doing as much of an analysis as possible in the recurrence language and its semantics, the fact that a particular piece of code uses existentials to implement a desired credit policy will not be tracked by the type system, but proved after recurrence extraction. An alternative approach would be to enrich λ^A with some form of indexed or dependent types to track the sizes of data structures in the type system, but such an extension is not necessary for our approach. The proofs of the results in this section are in Appendix B.

5.1. Existential Types in λ^A. To support existential quantifiers over credits, we extend the main typing judgment to be one of the form $\Delta|\Gamma \vdash_f M : A$, where $\Delta = \alpha_1, \ldots, \alpha_n$ is a list of "credit variables". Any of the α_i can occur free in the types in Γ, the resource term f, the term M, or the type A. Credit variables α range over *credit terms* c, which are (finite) sums of credit variables like α, β and credit constants ℓ — i.e. $\alpha_1 + \alpha_2 + \ldots + \alpha_n + l$. We write $\Delta \vdash c$ $\mathtt{credit}$ to mean that a credit term is well-formed from the variables in Δ. We consider credit terms up to the usual equations for addition on natural numbers. These credit terms can then be used as the "bank" in resource terms: the resource term $3x + 2y + (\alpha + 2)$ describes a context where one can use x 3 times, y twice, and has access to the credit term $\alpha + 2$ credits.

$$\frac{\Delta|\Gamma \vdash_f M : A[c/\alpha] \quad \Delta,\alpha \vdash A \quad \Delta \vdash c \ \texttt{credit}}{\Delta|\Gamma \vdash_f \texttt{pack}_{\alpha=c} M : \exists\alpha.A} \qquad \frac{M \downarrow^{(n,r)} v}{\texttt{pack}_{\alpha=\ell} M \downarrow^{(n,r)} \texttt{pack}_{\alpha=\ell} v}$$

$$\frac{\Delta|\Gamma \vdash_f M : \exists\alpha.A \quad \Delta,\alpha|\Gamma, x : A \vdash_{g+x} N : C \quad \Delta \vdash C}{\Delta|\Gamma \vdash_{f+g} \texttt{unpack}\ (\alpha, x) = M\ \texttt{in}\ N : C} \qquad \frac{M \downarrow^{(n_1,r_1)} \texttt{pack}_{\alpha=\ell} v_1 \quad N[\ell/\alpha, v_1/x] \downarrow^{(n_2,r_2)} v}{\texttt{unpack}\ (\alpha, x) = M\ \texttt{in}\ N \downarrow^{(n_1+n_2, r_1+r_2)} v}$$

FIGURE 12. Extension of λ^A with existential types

Most importantly, credit terms are now allowed to appear in the subscript of the ! modality (generalizing the natural number constants ℓ allowed above): a term $\alpha \mid \Gamma \vdash_f M :!_\alpha A$ with is an A with α credits attached. We add a new type $\exists\alpha.A$ for existentially quantifying over credit variables. A value of type $\exists\alpha.A$ is a value of type $A[c/\alpha]$, for some credit term c. Such a value does not store the ability to *use* the credits c — it stores a number of credits itself. However, combining the existential with the ! modality, a value of type $\exists\alpha.!_\alpha A$ is an A with c credits attached, for some credit term c. The operational semantics is defined for terms with no free credit variables, so its structure remains unchanged.

The typing rules and operational semantics for existential types are presented in Figure 12. The terms for existentials are standard `pack`/`unpack` terms. The operational semantics of `pack` and `unpack` are also standard; because we only evaluate closed terms, the credit term being packed/unpacked with the value will always be a (closed) natural number ℓ.

The rest of the rules for λ^A are mostly unchanged, so we do not repeat them: they are obtained from the rules in Figure 3 by carrying the credit variable context Δ through all of the rules, and, in the $!^k_c$ modality and the `save`, `transfer`, `create`, and `spend` terms, the natural number constants ℓ are generalized to credit terms c constructed from these variables. Finally, since the resource terms may contain free credit variables, the ordering judgment on resource terms must be augmented with a credit variable context, and the ordering itself extended to contain the coefficient-wise ordering on credit variables. The operational semantics for these constructs in unchanged, because closed credit terms are precisely the credit values ℓ used above.

For this extension, substitution and type preservation are stated as follows:

THEOREM 5.1 (Substitution).

- *If* $\Delta \vdash c$ *credit and* $\Delta, \alpha \vdash c'$ *credit, then* $\Delta \vdash c'[c/\alpha]$ *credit*

$$\begin{array}{rcl}
\langle\!\langle \exists\alpha.A \rangle\!\rangle & = & \$ \times \langle\!\langle A \rangle\!\rangle \\
\|\texttt{pack}_{\alpha=c} M\| & = & (\|M\|_c, (c, \|M\|_p)) \\
\|\texttt{unpack}\ (\alpha, x) = M\ \texttt{in}\ N\| & = & \|M\|_c +_c \|N\| [\pi_1 \|M\|_p / \alpha, \pi_2 \|M\|_p / x] \\
\|\texttt{create}_c\ M\| & = & (\text{to}\mathbb{C}(c) + \|M\|_c, \|M\|_) \\
\|\texttt{spend}_c\ M\| & = & (-\text{to}\mathbb{C}(c) + \|M\|_c, \|M\|_p)
\end{array}$$

FIGURE 13. Recurrence extraction for credit existentials

- *If* $\Delta \vdash c$ ***credit*** *and* $\Delta, \alpha|\Gamma \vdash_f M : A$, *then* $\Delta|\Gamma[c/\alpha] \vdash_{f[c/\alpha]} M[c/\alpha] : A[c/\alpha]$

THEOREM 5.2 (Preservation). *If* $\cdot|\cdot \vdash_a M : A$ *and* $M \downarrow^{(n,r)} v$, *then* $a + r \geq 0$ *and* $\cdot|\cdot \vdash_{a+r} v : A$.

5.2. Extracting Recurrences for Existentials. Recall that the recurrence extraction in Figure 7 erases the $!^k_\ell A$ modalities and translates $\texttt{create}_\ell\ M$ and $\texttt{spend}_\ell\ M$ by adding/subtracting ℓ to/from the amortized cost. Since we now allow credit variables α, such as those coming from unpacking an existential type, in the credit position of `create`/`spend`, the recurrence extraction will need to refer to the values chosen for α in order to know how much to add/subtract to/from the amortized cost. Thus, we add a type \$ to the recurrence language, the values of which are numbers of credits, represented by natural numbers. The credit context Δ is translated to recurrence language variables of type \$ (i.e. $\langle\!\langle \Delta, \alpha \rangle\!\rangle = \langle\!\langle \Delta \rangle\!\rangle, \alpha : \$$), while existential types $\exists\alpha.A$ are translated to pairs $\$ \times \langle\!\langle A \rangle\!\rangle$. A simple pair suffices because the ! modality is erased by $\langle\!\langle \cdot \rangle\!\rangle$, and this is the only place where credit terms can occur in the syntax of types, so all occurrences of α under the binder are removed, and $\langle\!\langle A \rangle\!\rangle$ is a closed type.

We show the new and changed cases of recurrence extraction in Figure 13. The introduction and elimination rules for $\exists\alpha.A$ translate to the corresponding introduction and elimination forms for $\$ \times \langle\!\langle A \rangle\!\rangle$. For `create` and `spend`, in principle, we would like the cost component of $\texttt{create}_c\ M$ to be $c + \|M\|_c$, but this will not type check, given that $c : \$$ but $\|M\|_c : \mathbb{C}$. Recalling that costs $\mathbb{C}$, though axiomatized as a monoid with some operations, are morally integers, we add a coercion $\texttt{to}\mathbb{C} : \$ \to \mathbb{C}$, which is morally the inclusion of natural numbers into integers.

THEOREM 5.3 (Extraction Preserves Types). *If* $\Delta|\Gamma \vdash_f M : A$, *then* $\langle\!\langle \Delta \rangle\!\rangle, \langle\!\langle \Gamma \rangle\!\rangle \vdash \|M\| : \|A\|$

5.3. Bounding Relation and Bounding Theorem. The definition of the bounding relation for values (Definition 3.1) is extended with

- $\texttt{pack}_{\alpha=\ell} v \sqsubseteq_{\texttt{val}}^{\exists\alpha.A,a} E$ iff $\ell \leq_{\$} \pi_1 E$ and $v \sqsubseteq_{\texttt{val}}^{A[\ell/\alpha],a} \pi_2 E$

Recalling that $E : \langle\!\langle \exists\alpha.A \rangle\!\rangle = \$ \times \langle\!\langle A \rangle\!\rangle$, this simply states that the amount of credit packed by α is bounded by the amount described by $\pi_1 E$, and that the value packed with the credit amount is in fact bounded by $\pi_2 E$. We remark that this definition may give the careful reader pause–inducting on a substitution instance of an existential type where the existential variable ranges over *types* leads to well-definedness issues. But, our existential variables range over *credits*, so we may simply regard a closed substitution instance of a type $\alpha \vdash A\,\texttt{type}$ as a smaller type than A.

The definition of the bounding relation for open terms must also be modified to quantify over closing substitutions for the credit context, as well as the term context. First, if ω is a substitution of credit amounts ℓ for credit variables, and Ω is a substitution of closed terms of type \$ for recurrence language variables, then $\omega \sqsubseteq^{\Delta} \Omega$ means that for all $\alpha \in \Delta$, $\omega(\alpha) \leq_{\$} \Omega(\alpha)$. Then for $\Delta|\Gamma \vdash_f M : A$ we write $M \sqsubseteq^{A} E$ if for all $\omega \sqsubseteq^{\Delta} \Omega$ and for all $\theta \sqsubseteq^{\Gamma[\omega],\sigma} \Theta$, we have that $M[\omega,\theta] \sqsubseteq^{A[\omega],f[\omega,\sigma]} E[\Omega,\Theta]$. Using this notation, the bounding theorem is

THEOREM 5.4 (Bounding Theorem). *If $\Delta|\Gamma \vdash_f M : A$, then $M \sqsubseteq^{A} \|M\|$*

and the cases which differ from the original Theorem 3.7 are proved in the supplementary materials.

5.4. Splay Tree Analysis. We now describe somewhat informally how to use the above machinery to analyze splay trees; the complete formalism is given in Appendix B (Figure B.1).Following Okasaki's presentation [58], the key operation is a $\texttt{split} : (A \times \texttt{tree}\,(A)) \to \texttt{tree}\,(A) \times \texttt{tree}\,(A)$ function that splits a given tree into elements larger and smaller than a given pivot. Insertion, deletion, union, intersection, difference etc. can be all implemented from `split` and a `join` operation that combines two sorted trees where all the elements of the first are less than the elements of the second. Showing that `split` is amortized $O(\log n)$ time, where n is the size of the tree, is the most difficult part of the amortized analysis, and implies the desired time bounds for the other operations. The key idea of splay trees is that each access rearranges the tree so

$$\begin{aligned}
N_1 &= \lambda\_.\mathtt{pack}_{\alpha=0}(\mathtt{save}_0^1\,())\\
N_2 &= \lambda(\_,(\alpha,\mathtt{save}_1^{\alpha}\,())).\mathtt{create}_1\,(\mathtt{pack}_{\beta=\alpha+1}\mathtt{save}_1^{\alpha+1}\,())\\
\mathtt{spawn}(n) &= \mathtt{nrec}\,(n, N_1, \mathtt{save}_0^{\infty}\,N_2) : \exists\alpha.!_{\alpha}^{1}1
\end{aligned}$$

FIGURE 14. λ^A term for the `spawn` function

that accessing the same element twice in a row is quicker the second time. In Okasaki's presentation, this rearrangement takes place in `split`, which performs a series of tree rotations. These rotations ensure that the amortized cost of `split` (amortized over any sequence of binary search tree operations) is $O(\log n)$, even though the tree is not always balanced. The most challenging cases of the code unpack the tree to depth two, and rotate the output if they traverses the same direction twice while searching for the pivot:

$$\begin{aligned}
&\mathit{split}\;p\;(N(x,N(y,a_{11},a_{12}),N(z,a_{21},a_{22})))|\;x \geq p\;\&\&\;y \geq p =\\
&\quad(\mathit{small}, N(y,\mathit{big},N(x,a_{12},N(z,a_{21},a_{22}))))\;\;\mathtt{where}\;\;(\mathit{small},\mathit{big}) = \mathtt{split}\;p\;a_{11}
\end{aligned}$$

Okasaki's analysis of split maintains the invariant that there are $\varphi(t) = \lceil \lg(|t|+1)\rceil$ credits associated with the root of every subtree t in a splay tree, and uses the potential/physicists method to analyze the amortized cost.

The addition of existentials to λ^A allows us to encode this analysis, by giving `split` the type $A \otimes \mathtt{tree}\left(\exists\alpha.!_{\alpha}^{1}A\right) \multimap \mathtt{tree}\left(\exists\alpha.!_{\alpha}^{1}A\right) \otimes \mathtt{tree}\left(\exists\alpha.!_{\alpha}^{1}A\right)$, and using code to maintain the invariant that each of these α's are precisely $\varphi(t)$.

5.4.1. *Creating Variable Amounts of Credit.* To maintain this invariant, we will sometimes need to `create` amounts of credit determined by a run-time natural number, like $\varphi(t)$ for some tree t—but the primitive $\mathtt{create}_c\ M$ term allows for waiting only for a credit term c, which cannot depend on run-time values. However, we can write a recursive loop that spawns a number of credits dependent on a run-time value, and package this as a function $\mathtt{spawn} : \mathbb{N} \multimap \exists\alpha.!_{\alpha}^{1}1$ such that the α packed in the result of $\mathtt{spawn}(n)$ is (the credit term representing) n. The implementation of `spawn` is shown in Figure 14—at a high level, the term loops $\mathtt{create}_1$ in a $\mathbb{N}$-recursor, using a credit existential as a counter variable. In this example, and throughout this section, we use pattern-matching notation as syntactic sugar for the elimination rules for positive types like $\exists, !, \otimes$, with the convention that matching on the result of a thunked recursive call implicitly forces it.

In Section 2.2, we argued that the n component in the operational cost semantics $M \downarrow^{n,r} v$ captures the actual operational cost of an erasure to simply-typed λ-calculus, as long as `tick`s in λ^A are inserted for each STLC β-redex. Because we do not include any `tick` terms in `spawn`, its abstract operational cost n is zero. Thus, to realize this cost semantics, `spawn` must be erased before actually running the program. Fortunately, a simple program optimization suffices to do this: translate λ^A to simply-typed λ-calclus by dropping both the $\exists$ and $!$ types and the associated terms, at which point `spawn` has type $\mathbb{N} \to 1$; then replace all terms of type 1 with the trivial value. That is, we think of `spawn` as a *ghost loop* — code that is meant for the extracted recurrence, but not intended to actually be run.

5.4.2. *Definition of Trees in λ^A.* Extending λ^A with the requisite `tree` type constructor and its rules follows both previous work [19] and the pattern illustrated with lists above. The type of trees is essentially $\texttt{tree}\,(A) = \texttt{Emp} \mid \texttt{N of } A \otimes \mathbb{N} \otimes \texttt{tree}\,(A) \otimes \texttt{tree}\,(A)$. The $\mathbb{N}$ argument caches the size of the tree, making the function $\texttt{size} : \texttt{tree}\,(A) \multimap \mathbb{N} \otimes \texttt{tree}\,(A)$ — which projects out that field and then rebuilds the tree[6] — constant time. To support coding the `split` function described above, we directly add a recursor that performs a two-level pattern match, with cases for the empty tree, for a node with one child or the other empty and the other is another node, and for a node with two nodes as children; in the latter case, the recursor provides recursive calls on all four subtrees.

5.4.3. *Splay Tree Implementation.* We define a *splay tree* to be a binary search tree t : $\texttt{tree}\,(\exists \alpha. !^{\infty}_{\alpha} A)$ satisfying the property that if $\texttt{size}(t) = n$, then if $t = N(\_, m, t_0, t_1)$, then t_0 and t_1 are splay trees, and for $[\![\|t\|_p]\!] = N((\alpha, \_), \_, \_)$, we have $\alpha = \phi(n)$. In other words, the credit invariant holds at each node in the tree. We note that each element of the tree not only carries α credits, but is also infinitely usable since we are required to compare nodes in the tree more than constantly many times. This causes no issues for the extracted recurrences, because keys in the tree are always values. We then prove a lemma which states that `split` preserves the splay tree property — i.e. that the existentially quantified credits stored in the tree satisfy the desired invariant.

[6]The tree can be rebuilt because values of type $\mathbb{N}$ are duplicable— there is a diagonal map $\mathbb{N} \multimap \mathbb{N} \otimes \mathbb{N}$. Also, we will often use `size` as a function $\texttt{tree}\,(A) \multimap \mathbb{N}$, and silently contract the second projection for re-use of the argument.

LEMMA 5.1. *If $t : \mathtt{tree}(\exists\alpha.!^{\infty}_{\alpha} A)$ is a splay tree and $\mathtt{split}(t) \downarrow (t_0, t_1)$, then t_0 and t_1 are also splay trees.*

To illustrate the λ^A term for `split`, we show one key case of the recursor, which corresponds to the snippet given at the beginning of this section and to [58, Theorem 5.2]. For this case, we are in the situation where the root, labeled by x, has two subtrees, y with subtrees a_{11}, a_{12}, and z with subtrees $, a_{21}, a_{22}$. If the pivot is less than both x and y, we recur on the leftmost subtree a_{11}, which produces the elements of a_{11} that are smaller and bigger than the pivot. Then *smaller* contains all the elements of the original tree smaller than the pivot. The elements bigger than the pivot are *bigger* and everything else from the original tree; we combine these together into a new tree, performing a rotation to put y at the root.

The λ^A version of this term, presented in Figure 15, annotates the above code with some additional information about the sizes of trees, and with some code for manipulating credits. The variables x, y, z are the values of type A at the root and its immediate children; these come with existentially-quantified numbers of credits α, β, γ (α credits are stored with x, β with y, and γ with z), and also with natural numbers caching the sizes of the subtrees that they are the roots of (n_1, n_2, n_3 respectively). The variables a_{ij} stand for the four subtrees with their (suspended) recursive call outputs; we write $\mathtt{split}(p, a_{11})$ for projecting and forcing the recursive call, and write a_{ij} for projecting the other subtrees. The credit manipulation involves spending the credits α and β stored with x and y in the input tree (we do not spend z, because the z node is left unchanged in the output), calculating the sizes of the new nodes t' and s' that will be part of the output, and `spawn`ing credits corresponding to φ of these sizes. The term presented in Figure 15 is one branch of one of the step functions passed to the `treerec` which forms the outermost structure of `split`.

To analyze splay trees, we pass this λ^A term through recurrence extraction and the preorder semantics and then prove the following:

THEOREM 5.5. *If $t : \mathtt{tree}(\exists\alpha.!^{\infty}_{\alpha} A)$ is a splay tree with $\mathtt{size}(t) = n$, then for any $v : A$, $[\![\|\mathtt{split}(t, v)\|_c]\!] \leq 1 + 2\varphi([\![\|\mathtt{size}\|_p\,(t)]\!]) \in O(\lg n)$.*

```
λ((α, save^∞_α x), n_1, (β, save^∞_β y), n_2, (γ, save^∞_γ z), n_3, a_11, a_12, a_21, a_22).
if  x ≥ p && y ≥ p then tick; let
    (small, big)       = spend_{α+β} (split(p, a_11))             d        = N(pack_{α=γ}(save^∞_γ z), n_3, a_21, a_22)
    n_12               = size(a_12)                                n_big    = size(big)
    t'_size            = 1 + n_12 + n_3                            s'_size  = 2 + n_big + n_12 + n_3
    ((α', _), (β', _)) = (spawn(φ(t'_size)), spawn(φ(s'_size)))    t'       = N(pack_{α=α'}(save^∞_α' x), t'_size, a_12, d)
    s'                 = N(pack_{α=β'}(save^∞_β' y), s'_size, big, t')
  in (small, s') end
else ...
```

FIGURE 15. Part of the λ^A term for `split`

PROOF. As an example, we show the case for the code in Figure 15. The cost component of the extracted recurrence is

$$1 - \alpha - \beta + [\![\|\texttt{split}(p, a_{11})\|]\!] + \varphi(1 + n_{12} + n_3) + \varphi(2 + n_{big} + n_{12} + n_3)$$

The 1 comes from the `tick`; α and β are subtracted because they are `spent`; and the φ of the sizes of t' and s' are added because they are `created`. By definition, $1 + n_{12} + n_3 = [\![\|\texttt{size}\|_p\,(t')]\!]$ and $2 + n_{big} + n_{12} + n_3 = [\![\|\texttt{size}\|_p\,(s')]\!]$. By the credit invariant, $\alpha = \varphi([\![\|\texttt{size}\|_p\,(t)]\!])$, and $\beta = \varphi([\![\|\texttt{size}\|_p\,(s)]\!])$, where s is the subtree of t rooted at y. Rewriting by these and commuting terms, the extracted recurrence is precisely

$$1 + [\![\|\texttt{split}(p, a_{11})\|]\!] + \varphi([\![\|\texttt{size}\|_p\,(s')]\!]) + \varphi([\![\|\texttt{size}\|_p\,(t')]\!]) - \varphi([\![\|\texttt{size}\|_p\,(s)]\!]) - \varphi([\![\|\texttt{size}\|_p\,(t)]\!])$$

which Okasaki [58, Theorem 5.2] proves is bounded by $1 + 2\varphi(\texttt{size}(t))$, as required. □

6. Related Work

Techniques for extracting (asymptotic) cost information from high-level program source code is a project that is almost as old as studying programming languages. For non-amortized analysis of functional languages, we have examples from the 1970s and 1980s by Wegbreit [82], Le Métayer [44], and Rosendahl [69]. The idea of simultaneously extracting information about cost and size, and defining the size of a function to be a function itself (leading to higher-order recurrences) has its roots in Danner and Royer [17], which in turn draws from ideas in Shultis [71], Sands [70], and Van Stone [73]. Using bounded modal operators to describe resource usage goes back at least to Girard et al. [25], and Orchard et al. have recently incorporated these ideas into the Granule language [59]. Perhaps the work that is closest in spirit to ours is Benzinger's ACA system for analyzing call-by-name NUPRL programs [6]. From a cost-annotated operational semantics, he extracts a "symbolic semantics" that is similar in flavor to our recurrence language and extracted recurrences, although without amortization. The symbolic semantics yields higher-order recurrences, which he reduces to first-order recurrences that can be analyzed with a computer algebra system.

There is also extensive work on recurrence extraction from first-order imperative languages. The COSTA project [1, 2, 3] takes Java bytecode as its source language, extracts cost relations (essentially, non-deterministic cost recurrences), and solves them for upper bounds. In this line of work, Alonso-Blas and Genaim [4] and Flores-Montoya [23] investigate the failure to derive tight upper bounds in settings where amortized analysis is typically deployed. They trace the issue to the fact that typically cost relations do not depend on the results of the analyzed functions. Making this possible allows more precise constraints which, when solved, yield tighter bounds. The dependency on output corresponds roughly to total accumulated savings, and they infer an appropriate potential function (in the terminology of the physicist's method), modulo a choice of templates. To analogize with our work, they delay the determination of the credit policy until solving for upper bounds of extracted recurrences, whereas we specify the credit policy as part of the source program, which directly yields a recurrence for cost that takes the policy into account.

Two recent approaches that handle amortized analysis for functional programs are Timed ML (TiML, [81]) and automatic amortized resource analysis (AARA, [32, 30, 33, 55]). In TiML, ML type and function definitions are annotated with indices that convey size information. The notion of size is left unspecified and the indices are very flexible, and can include constraints such as those required to define red-black trees. Type inference generates verification conditions. Depending on the details of the annotations, solving the verification conditions provides exact or asymptotic bounds on the cost of the original program. The focus is on worst-case analysis, but the annotation language is sufficiently rich to encode the physicist's method of amortized analysis. Although it is not part of their focus, the formalism does not appear to enable analysis of higher-order functions whose cost depends on the complexity behavior of the function arguments.

AARA provides a type inference system for resource bound analysis of higher-order functional programs that incorporates amortization. Credit allocation is built into the type system itself. Soundness says that the net credit change during evaluation is bounded by the net credit change described by the typing. AARA focuses primarily on strict languages, but Jost et al. [38] use similar ideas to analyze programs under lazy evaluation. In AARA, the credit allocation and usage is described in the type judgment. Type inference generates constraints, and the solution of these constraints is essentially a credit allocation strategy. Our approach describes usage in the type judgment, but requires the strategy to be explicit in the program (via `save`, `create`, `spend`, etc.), which places a greater burden on the programmer. However, reasoning about that strategy (e.g., establishing a credit invariant) in the semantics may provide more flexibility, though that requires more investigation.

We note that the technical differences between TiML and AARA and our approach arise from a difference in what we might consider the philosophical underpinnings. TiML and AARA introduce novel type systems with a goal of inferring cost bounds to the greatest extent possible. Those bounds are extracted as part of the type inference procedure. This is not how most programmers conceptualize a cost analysis, and our interest is in staying as close to typical informal analyses as we can. While λ^A is a novel type system, the novelty exists solely in order to make the programmer be explicit about how credits are allocated and used. This task is part of a banker's-method analysis, though it is usually stated informally ("put one credit on each 1 in the bit list"). After that, it is extraction of ordinary (semantic) recurrences which one hopes to be able to bound using whatever methods are at the programmer's disposal.

Bibliography

[1] Elvira Albert, Puri Arenas, Samir Genaim, and Germán Puebla. Closed-form upper bounds in static cost analysis. *Journal of Automated Reasoning*, 46:161–203, 2011. doi: 10.1007/s10817-010-9174-1. 10.0.1, 6

[2] Elvira Albert, Puri Arenas, Samir Genaim, German Puebla, and Damiano Zanardini. Cost analysis of object-oriented bytecode programs. *Theoretical Computer Science*, 413 (1):142–159, 2012. doi: 10.1016/j.tcs.2011.07.009. 10.0.1, 6

[3] Elvira Albert, Samir Genaim, and Abu Naser Masud. On the Inference of Resource Usage Upper and Lower Bounds. *ACM Transactions on Computational Logic*, 14(3):22:1–22:35, 2013. doi: 10.1145/2499937.2499943. 10.0.1, 6

[4] Diego Esteban Alonso-Blas and Samir Genaim. On the limits of the classical approach to cost analysis. In Antoine Miné and David Schmidt, editors, *Static Analysis*, volume 7460 of *Lecture Notes in Computer Science*, pages 405–421. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-33125-1_27. 6

[5] Robert Atkey. Syntax and semantics of quantitative type theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018*, pages 56–65. ACM Press, 2018. doi: 10.1145/3209108.3209189. 10.0.3, 1, 2, 2

[6] Ralph Benzinger. Automated higher-order complexity analysis. *Theoretical Computer Science*, 318(1-2):79–103, 2004. doi: 10.1016/j.tcs.2003.10.022. 6

[7] François Bobot, Jean-Christophe Filliâtre, Claude Marché, and Andrei Paskevich. Why3: Shepherd your herd of provers. In *Boogie 2011: First International Workshop on Intermediate Verification Languages*, pages 53–64, Wrocław, Poland, August 2011. `https://hal.inria.fr/hal-00790310`. 9.2, 9.2.6, 9.3

[8] EDWIN BRADY. Idris, a general-purpose dependently typed programming language: Design and implementation. *Journal of Functional Programming*, 23(5):552–593, 2013. 1

[9] Quentin Carbonneaux, Jan Hoffmann, and Zhong Shao. Compositional certified resource bounds. *SIGPLAN Not.*, 50(6):467–478, June 2015. ISSN 0362-1340. doi: 10.1145/2813885.2737955. URL https://doi.org/10.1145/2813885.2737955. 10.0.1

[10] Ezgi Çiçek, Weihao Qu, Gilles Barthe, Marco Gaboardi, and Deepak Garg. Bidirectional type checking for relational properties. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, page 533–547, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367127. doi: 10.1145/3314221.3314603. URL https://doi.org/10.1145/3314221.3314603. 9.2.4, 10.0.1

[11] Iliano Cervesato, Joshua S Hodas, and Frank Pfenning. Efficient resource management for linear logic proof search. *Theoretical Computer Science*, 232(1-2):133–163, 2000. 5, 4.0.4

[12] Arthur Charguéraud and François Pottier. Verifying the correctness and amortized complexity of a union-find implementation in separation logic with time credits. *Journal of Automated Reasoning*, 62(3):331–365, 2019. doi: 10.1007/s10817-017-9431-7. URL https://doi.org/10.1007/s10817-017-9431-7. 10.0.1

[13] Pritam Choudhury, Harley Eades III, Richard A. Eisenberg, and Stephanie Weirich. A graded dependent type system with a usage-aware semantics. *Proc. ACM Program. Lang.*, 5 (POPL), January 2021. doi: 10.1145/3434331. URL https://doi.org/10.1145/3434331. 10.0.3

[14] Nils Anders Danielsson. Lightweight semiformal time complexity analysis for purely functional data structures. In George Necula and Philip Wadler, editors, *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008*, page 133–144. ACM Press, 2008. doi: 10.1145/1328438.1328457. 1, 2.0.1, 10.0.2, 2

[15] Norman Danner and Daniel R. Licata. Denotational semantics as a foundation for cost recurrence extraction for functional languages, 2020. 1, 2, 4.1

[16] Norman Danner and Daniel R. Licata. Denotational semantics as a foundation for cost recurrence extraction for functional languages, 2021. 1

[17] Norman Danner and James S. Royer. Adventures in time and space. *Logical Methods in Computer Science*, 3(9):1–53, 2007. doi: 10.2168/LMCS-3(1:9)2007. 3.1, 6

[18] Norman Danner, Jennifer Paykin, and James S. Royer. A static cost analysis for a higher-order language. In Matthew Might and David Van Horn, editors, *Proceedings of the 7th workshop on Programming languages meets program verification, PLPV 2013*, pages 25–34. ACM Press, 2013. doi: 10.1145/2428116.2428123. 1, 1, 3.1, 4

[19] Norman Danner, Daniel R. Licata, and Ramyaa Ramyaa. Denotational cost semantics for functional languages with inductive types. In Kathleen Fisher and John Reppy, editors, *Proceedings of the 20th ACM SIGPLAN International Conference on Functional Programming, ICFP 2015*, pages 140–151. ACM Press, 2015. doi: 10.1145/2784731.2784749. (document), 1, 1, 2, 3, 3.1, 3.1, 3.2, 4, 4, 4.1, 5.4.2

[20] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008. 9.3

[21] Jana Dunfield and Neel Krishnaswami. Bidirectional typing. `arXiv:1908.05839[cs.PL]`, 2019. 17

[22] Jana Dunfield and Frank Pfenning. Tridirectional typechecking. In X.Leroy, editor, *Conference Record of the 31st Annual Symposium on Principles of Programming Languages (POPL'04)*, pages 281–292, Venice, Italy, January 2004. ACM Press. Extended version available as Technical Report CMU-CS-04-117, March 2004. 4.0.1

[23] Antonio Flores-Montoya. Upper and lower amortized cost bounds of programs expressed as cost relations. In John Fitzgerald, Constance Heitmeyer, Stefania Gnesi, and Anna Philippou, editors, *FM 2016: Formal Methods*, volume 9995 of *Lecture Notes in Computer Science*, pages 254–273. Springer International Publishing, 2016. doi: 10.1007/978-3-319-48989-6_16. 6

[24] Marco Gaboardi, Shin-ya Katsumata, Dominic Orchard, Flavien Breuvart, and Tarmo Uustalu. Combining effects and coeffects via grading. *SIGPLAN Not.*, 51(9):476–489, September 2016. ISSN 0362-1340. doi: 10.1145/3022670.2951939. URL `https://doi.org/10.1145/3022670.2951939`. 2.0.1

[25] Jean-Yves Girard, Andre Scedrov, and Philip J. Scott. Bounded linear logic: a modular approach to polynomial-time computability. *Theoretical Computer Science*, 97(1):1–66, 1992. doi: 10.1016/0304-3975(92)90386-T. 4, 1, 2, 2, 2.1, 6

[26] J. A. Goguen and Jose Meseguer. Security policies and security models. *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 11–20, 1982. ISSN 1063-7109. 1

[27] Martin A. T. Handley, Niki Vazou, and Graham Hutton. Liquidate your assets: Reasoning about resource usage in liquid haskell. *Proc. ACM Program. Lang.*, 4(POPL), December 2019. doi: 10.1145/3371092. URL `https://doi.org/10.1145/3371092`. 1

[28] Jan Hoffmann. *Types with Potential: Polynomial Resource Bounds via Automatic Amortized Analysis*. PhD thesis, LMU Munich, 2011. 2.1

[29] Jan Hoffmann and Martin Hofmann. Amortized resource analysis with polynomial potential: A static inference of polynomial bounds for functional programs. In *Proceedings of the 19th European Conference on Programming Languages and Systems*, ESOP'10, page 287–306, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3642119565. doi: 10.1007/978-3-642-11957-6_16. URL `https://doi.org/10.1007/978-3-642-11957-6_16`. 1, 2.1

[30] Jan Hoffmann and Zhong Shao. Automatic static cost analysis for parallel programs. In Jan Vitek, editor, *Programming Languages and Systems: 24th European Symposium on Programming, ESOP 2015*, volume 9032 of *Lecture Notes in Computer Science*, pages 132–157. Springer-Verlag, 2015. doi: 10.1007/978-3-662-46669-8_6. 6

[31] Jan Hoffmann, Klaus Aehlig, and Martin Hofmann. Resource aware ml. In *Proceedings of the 24th International Conference on Computer Aided Verification*, CAV'12, page 781–786, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 9783642314230. doi: 10.1007/978-3-642-31424-7_64. URL `https://doi.org/10.1007/978-3-642-31424-7_64`. 1, 9.3, 10.0.3

[32] Jan Hoffmann, Klaus Aehlig, and Martin Hofmann. Multivariate Amortized Resource Analysis. *ACM Transactions on Programming Languages and Systems*, 34(3):14:1–14:62, 2012. doi: 10.1145/2362389.2362393. 1, 2.1, 6

[33] Jan Hoffmann, Ankush Das, and Shu-Chun Weng. Towards automatic resource bound analysis for OCaml. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017*, pages 359–373. ACM Press, 2017. ISBN 978-1-4503-4660-3. doi: 10.1145/3009837.3009842. 6

[34] Martin Hofmann. The strength of non-size increasing computation. In *Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, page 260–269, 2002. doi: 10.1145/503272.503297. 1

[35] Martin Hofmann. Linear types and non-size-increasing polynomial time computation. *Information and Computation*, 183(1):57–85, 2003. doi: 10.1016/S0890-5401(03)00009-9. 1

[36] Martin Hofmann and Steffen Jost. Static prediction of heap space usage for first-order functional programs. In Alex Aiken and Greg Morrisett, editors, *Proceedings of the 30th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, page 185–197. ACM Press, 2003. doi: 10.1145/604131.604148. 10.0.3, 1

[37] Hudson, Bowornmet. Computer-Checked Recurrence Extraction for Functional Programs. Master's thesis, Wesleyan University, 2016. 1, 3, 4, 4.1

[38] Steffen Jost, Pedro Vasconcelos, Mário Florido, and Kevin Hammond. Type-based cost analysis for lazy functional languages. *Journal of Automated Reasoning*, 59(1):87–120, 2017. doi: 10.1007/s10817-016-9398-9. 6

[39] G. A. Kavvos. Dual-Context Calculi for Modal Logic. Volume 16, Issue 3, August 2020. 16

[40] G. A. Kavvos, Edward Morehouse, Daniel R. Licata, and Norman Danner. Recurrence extraction for functional programs through call-by-push-value. *Proceedings of the ACM on Programming Languages*, 4(POPL), 2019. doi: 10.1145/3371083. 1, 1, 3.1

[41] Tristan Knoth, Di Wang, Nadia Polikarpova, and Jan Hoffmann. Resource-guided program synthesis. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019. 10.0.3, 1

[42] Tristan Knoth, Di Wang, Adam Reynolds, Jan Hoffmann, and Nadia Polikarpova. Liquid resource types. *Proc. ACM Program. Lang.*, 4(ICFP), August 2020. doi: 10.1145/3408988. URL https://doi.org/10.1145/3408988. 1, 10.0.3

[43] Joomy Korkut, Maksim Trifunovski, and Daniel R. Licata. Intrinsic verification of a regular expression matcher. 1, 1

[44] Daniel Le Métayer. ACE: an automatic complexity evaluator. *ACM Transactions on Programming Languages and Systems*, 10(2):248–266, 1988. doi: 10.1145/42190.42347. 6

[45] Yao Li, Li yao Xia, and Stephanie Weirich. Reasoning about the garden of forking paths, 2021. 10.0.1

[46] Daniel R. Licata, Michael Shulman, and Mitchell Riley. A fibrational framework for substructural and modal logics. In Dale Miller, editor, *2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017*, volume 84 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:22. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. doi: 10.4230/LIPIcs.FSCD.2017.25. 2.1

[47] John M Lucassen and David K Gifford. Polymorphic effect systems. In *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 47–57, 1988. 2

[48] Connor McBride. *Dependently Typed Functional Programs and their Proofs.* PhD thesis, University of Edinburgh, 2000. 9.2.1

[49] Conor McBride. I got plenty o' nuttin'. In Sam Lindley, Conor McBride, Phil Trinder, and Don Sannella, editors, *A List of Successes That Can Change the World: Essays Dedicated to Philip Wadler on the Occasion of His 60th Birthday*, volume 9600 of *Lecture Notes in Computer Science*. Springer-Verlag, 2016. doi: 10.1007/978-3-319-30936-1_12. 1, 2

[50] Jay McCarthy, Burke Fetscher, Max New, Daniel Feltey, and Robert Bruce Findler. A coq library for internal verification of running-times. In Oleg Kiselyov and Andy King, editors, *Functional and Logic Programming*, pages 144–162, Cham, 2016. Springer International Publishing. ISBN 978-3-319-29604-3. 1, 10.0.2

[51] Glen Mével, Jacques-Henri Jourdan, and François Pottier. Time credits and time receipts in iris. In Luís Caires, editor, *Programming Languages and Systems*, pages 3–29, Cham, 2019. Springer International Publishing. ISBN 978-3-030-17184-1. 10.0.1

[52] Microsoft. TypeScript, 2012 - 2021. URL `http://typescriptlang.org/`. 1

[53] Eugenio Moggi. Notions of computation and monads. *Information and computation*, 93(1): 55–92, 1991. 2

[54] Yue Niu and Robert Harper. Cost-aware type theory, 2020. 2, 10.0.3

[55] Yue Niu and Jan Hoffmann. Automatic space bound analysis for functional programs with garbage collection. In Gilles Barthe, Geoff Sutcliffe, and Margus Veanes, editors, *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 57 of *EPiC Series in Computing*, pages 543–563. EasyChair, 2018. doi: 10.29007/xkwx. 6

[56] Ulf Norell. Dependently typed programming in agda. In *Proceedings of the 6th International Conference on Advanced Functional Programming*, AFP'08, page 230–266, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3642046517. 1, 1, 9.2.1

[57] Peter W. O'Hearn. Incorrectness logic. *Proc. ACM Program. Lang.*, 4(POPL), December 2019. doi: 10.1145/3371078. URL `https://doi.org/10.1145/3371078`. 10.0.1

[58] Chris Okasaki. *Purely Functional Data Structures*. Cambridge University Press, 1998. 3.0.3, 1, 5.4, 5.4.3, 5.4.3

[59] Dominic Orchard, Vilem-Benjamin Liepelt, and Harley Eades III. Quantitative program reasoning with graded modal types. *Proceedings of the ACM on Programming Languages*, 3(ICFP):110:1–110:30, 2019. doi: 10.1145/3341714. 6

[60] Dominic Orchard, Vilem-Benjamin Liepelt, and Harley Eades III. Quantitative program reasoning with graded modal types. *Proc. ACM Program. Lang.*, 3(ICFP), July 2019. doi: 10.1145/3341714. URL `https://doi.org/10.1145/3341714`. 21, 10.0.3

[61] Benjamin Pierce. Types and programming languages: The next generation. Presented at Eighteenth Annual IEEE Symposium on Logic In Computer Science (2003(. URL `https://www.cis.upenn.edu/~bcpierce/papers/tng-lics2003-slides.pdf`. 13

[62] Benjamin C. Pierce and David N. Turner. Local type inference. *ACM Trans. Program. Lang. Syst.*, 22(1):1–44, January 2000. ISSN 0164-0925. doi: 10.1145/345099.345100. URL `https://doi.org/10.1145/345099.345100`. 1, 2, 4.0.1

[63] Gordon Plotkin and John Power. Computational effects and operations: An overview. 2002. 2

[64] Vineet Rajani, Marco Gaboardi, Deepak Garg, and Jan Hoffmann. A unifying type-theory for higher-order (amortized) cost analysis. *Proc. ACM Program. Lang.*, 5(POPL), January 2021. doi: 10.1145/3434308. URL `https://doi.org/10.1145/3434308`. (document), 1, 1, 1, 5, 2.1, 3, 5, 5.2.6, 6.2.1, 9.2, 9.3, 10.0.3

[65] Jason Reed. Names are (mostly) useless. Presented at 3rd Informal ACM SIGPLAN Workshop on Mechanizing Metatheory 2008. URL `https://www.cis.upenn.edu/~sweirich/wmm/wmm08-programme.html`. 1, 2

[66] John C. Reynolds. Definitional interpreters for higher-order programming languages. In *Proceedings of the ACM Annual Conference - Volume 2*, ACM '72, page 717–740, New

York, NY, USA, 1972. Association for Computing Machinery. ISBN 9781450374927. doi: 10.1145/800194.805852. URL https://doi.org/10.1145/800194.805852. 4, 9.2

[67] Yann-Joachim Ringard. Mustard Watches: an Integrated Approach to Time and Food. 1990. (document)

[68] Talia Ringer, Karl Palmskog, Ilya Sergey, Milos Gligoric, and Zachary Tatlock. Qed at large: A survey of engineering of formally verified software. *Foundations and Trends® in Programming Languages*, 5(2-3):102–281, 2019. ISSN 2325-1131. doi: 10.1561/2500000045. URL http://dx.doi.org/10.1561/2500000045. 1, 1

[69] Mads Rosendahl. Automatic complexity analysis. In Joseph E. Stoy, editor, *Proceedings of the Fourth International Conference on Functional Programming Languages and Computer Architecture, FPCA 1989*, pages 144–156. ACM Press, 1989. doi: 10.1145/99370.99381. 6

[70] David Sands. *Calculi for Time Analysis of Functional Programs*. PhD thesis, University of London, 1990. 6

[71] Jon Shultis. On the complexity of higher-order programs. Technical Report CU-CS-288-85, University of Colorado at Boulder, 1985. 6

[72] Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32(3), 1985. 1, 5

[73] Kathryn Van Stone. *A Denotational Approach to Measuring Complexity in Functional Programs*. PhD thesis, Carnegie Mellon University, 2003. 6

[74] Nikhil Swamy, Juan Chen, and Ben Livshits. Verifying higher-order programs with the dijkstra monad. In *ACM Programming Language Design and Implementation (PLDI) 2013*. ACM, June 2013. URL https://www.microsoft.com/en-us/research/publication/verifying-higher-order-programs-with-the-dijkstra-monad/. 1, 1

[75] Robert Endre Tarjan. Amortized computational complexity. *SIAM Journal on Algebraic and Discrete Methods*, 6(2):306–318, 1985. doi: 10.1137/0606031. (document), 1, 1, 1

[76] The Coq Development Team. The Coq Proof Assistant, 1989 - 2021. URL http://coq.inria.fr/. 1

[77] The D Language Foundation. D, 2007 - 2021. URL http://dlang.org/. 1

[78] The Rust Core Team. Rust, 2010 - 2021. URL http://rust-lang.org/. 1

[79] Niki Vazou, Eric L. Seidel, Ranjit Jhala, Dimitrios Vytiniotis, and Simon Peyton-Jones. Refinement types for haskell. In *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming*, ICFP '14, page 269–282, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450328739. doi: 10.1145/2628136.2628161. URL https://doi.org/10.1145/2628136.2628161. 10.0.3

[80] Niki Vazou, Eric L. Seidel, Ranjit Jhala, Dimitrios Vytiniotis, and Simon Peyton-Jones. Refinement types for haskell. *SIGPLAN Not.*, 49(9):269–282, August 2014. ISSN 0362-1340. doi: 10.1145/2692915.2628161. URL https://doi.org/10.1145/2692915.2628161. 1

[81] Peng Wang, Di Wang, and Adam Chlipala. Timl: A functional language for practical complexity analysis with invariants. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA), 2017. doi: 10.1145/3133903. 1, 10.0.3, 6

[82] Ben Wegbreit. Mechanical program analysis. *Communications of the Association for Computing Machinery*, 18(9):528–539, 1975. doi: 10.1145/361002.361016. 6

[83] J.B. Wells. Typability and type checking in system f are equivalent and undecidable. *Annals of Pure and Applied Logic*, 98(1):111–156, 1999. ISSN 0168-0072. doi: https://doi.org/10.1016/S0168-0072(98)00047-5. URL https://www.sciencedirect.com/science/article/pii/S0168007298000475. 2

[84] Hongwei Xi. Dependent ml, an approach to practical programming with dependent types. *Journal of Functional Programming*, 17(2):215–286, 2007. doi: 10.1017/S0956796806006216. 1, 2.0.3, 4.0.2, 5

[85] Christoph Zenger. Indexed types. *Theor. Comput. Sci.*, 187(1–2):147–165, November 1997. ISSN 0304-3975. doi: 10.1016/S0304-3975(97)00062-5. URL https://doi.org/10.1016/S0304-3975(97)00062-5. 4.0.2

APPENDIX A

1. Example Terms

1.1. Insertion Sort.

```
$\texttt{insert} : \forall n : \mathbb{N}. !\tau \multimap L^n\,(!\tau) \multimap \mathbb{M}\,(n, \langle 0,1\rangle)\,\left(L^{n+1}\,(!\tau)\right)$

$\texttt{insert} = \texttt{fix}(\texttt{insert}.$
   $\Lambda n.\lambda b_x.\texttt{let}\,!x = b_x\,\texttt{in}$
   $\lambda xs.$
   $\texttt{case}(xs, \texttt{ret}(!(x :: \texttt{nil})), by.ys.$
      $\texttt{shift}($
         $\texttt{bind}\,b = \texttt{leq}\,(x,y)\,[n-1]\,\texttt{in}$
         $\texttt{if}\,b\,\texttt{thenret}\,((!x) :: (!y) :: ys)$
         $\texttt{else bind}\,zs = \texttt{insert}\,[n-1]\,(!x)\,ys\,\texttt{in}$
         $\texttt{ret}\,((!y) :: zs)$
      $)$
   $)$
$)$
```

```
$\texttt{ins\_sort} : \forall n : \mathbb{N}.\,L^n\,(!\tau) \multimap \mathbb{M}\,(n, \langle 0,0,1\rangle)\,(L^n\,(!\tau))$

$\texttt{ins\_sort} = \texttt{fix}(\texttt{ins\_sort}.$
   $\Lambda n.\lambda xs.$
   $\texttt{case}(xs, \texttt{ret}\,\texttt{nil}, y, ys.$
      $\texttt{shift}($
         $\texttt{bind}\,ys' = \texttt{ins\_sort}\,[n-1]\,ys\,\texttt{in}$
         $\texttt{insert}\,[n-1]\,y\,ys$
      $)$
   $)$
$)$
```

1.2. Functional Queue.

```
$\texttt{enq} : \forall n, m : \mathbb{N}.\,[3]\,1 \multimap \tau \multimap L^n([2]\,\tau) \otimes L^m\,\tau \multimap \mathbb{M}\,\langle 0\rangle\,\left(L^{n+1}([2]\,\tau) \otimes L^m\,\tau\right)$

$\texttt{enq} = \Lambda n.\Lambda m.\lambda p.\lambda x.\lambda ins.\lambda outs.\Lambda j.$
   $\texttt{release}\,\_ = p\,\texttt{in}$
   $\texttt{bind}\,\_ = \texttt{tick}[j|[1.0]]\,\texttt{in}$
   $\texttt{bind}\,x' = \texttt{store}[2.0](x)\,\texttt{in}$
   $\texttt{ret}(x' :: ins, outs)$
```

```
move : $\forall n:\mathbb{N}.\forall m:\mathbb{N}.L^n([2]\,\tau) \multimap L^m\tau \multimap \mathbb{M}\langle 0\rangle L^{m+n}\tau$

move = fix(move.
  $\Lambda n.\Lambda m.\lambda xs.\lambda ys.\Lambda j.$
  case($xs$, ret ys, $z.zs$.
    release $z'$ = $z$ in
    bind _ = tick[$j$|[2.0]] in
    bind $zs'$ = move $[n-1]\,[m]\,zs\,ys\,[j]$ in
    ret($z' :: zs'$)
  )
)
```

```
deq : $\forall m,n:\mathbb{N}.(m+n>0) \implies L^n([2]\,\tau)\otimes L^m\,\tau \multimap$
        $\mathbb{M}\langle 0\rangle \left(\exists n',m':\mathbb{N}.(n'+m'+1=n+m)\&\left(L^{n'}([2]\,\tau)\otimes L^{m'}\,\tau\right)\right)]$

deq = fix(deq.
  $\Lambda n.\Lambda m.\Lambda.\lambda l_1.\lambda l_2.\lambda j.$
  case($l2$,
    bind $l_r$ = move $[n]\,[0]\,l_1$ nil $[j]$ in
    case($l_r$, fix($x.x$), $h_r.l_r'$
      ret (pack[0] (pack[$n-1$] < (($h_r$, nil), $l_r'$) >))
    ), $h_2.l_2'$.ret (pack[$n$] (pack[$m-1$] < (($h2, l_1$), $l_2'$) >))
  )
)
```

1.3. Map.

```
map : $\forall\alpha,\beta:\star.\forall C:\mathbb{N}\to\mathbb{R}^{+}.\forall n:\mathbb{N}.$
        $!\,(\forall i:\mathbb{N}.[C\,i]\,1 \multimap$ Nat$(i) \multimap \alpha \multimap \mathbb{M}\,\langle 0\rangle\,\beta) \multimap$
        $!$Nat$(n) \multimap$
        $L^{n}\,\alpha \multimap$
        $\mathbb{M}\,\langle$const$\,(\sum_{0\le i<n} C(i))\rangle\;(L^{n}\,\beta)$

map = $\Lambda\alpha.\Lambda\beta.\Lambda C.$fix(map.
  $\lambda p.\lambda f_b.\lambda xs.\lambda N_b.\Lambda j.$
    let $!f = f_b$ in
    let $!N = N_b$ in
    release _ = $p$ in
    case($xs$, ret nil, $y.ys.$
      bind $p'$ = store$[C\,(n-1)]()$ in
      bind $z = f\,[n-1]\,p'$ (pred$\,[n]\,[]\,N$)        )
)
```

2. Rules of dλ-Amor

2.1. Wellformedness Judgments.

WF-CCtxE
$$\frac{}{\Theta \vdash \cdot \ \mathtt{wf}}$$

WF-CCtxNe
$$\frac{\Theta \vdash \Delta \ \mathtt{wf} \qquad \Theta; \Delta \vdash \Phi \ \mathtt{wf}}{\Theta \vdash \Delta, \Phi \ \mathtt{wf}}$$

WF-TCtxE
$$\frac{}{\Psi; \Theta; \Delta \vdash \cdot \ \mathtt{wf}}$$

WF-TCtxNE
$$\frac{\Psi; \Theta; \Delta \vdash \Gamma \ \mathtt{wf} \qquad \Psi; \Theta; \Delta \vdash \tau : \star}{\Psi; \Theta; \Delta \vdash \Gamma, x : \tau \ \mathtt{wf}}$$

2.2. Sort Assignment.

I-Var
$$\frac{i : S \in \Theta}{\Theta; \Delta \vdash i : S}$$

I-Plus
$$\frac{\Theta; \Delta \vdash I : bS \qquad \Theta; \Delta \vdash J : bS}{\Theta; \Delta \vdash I + J : bS}$$

I-Minus
$$\frac{\Theta; \Delta \vdash I : bS \qquad \Theta; \Delta \vdash J : bS \qquad \Theta; \Delta \vDash I \geq J}{\Theta; \Delta \vdash I - J : bS}$$

I-Times-$\mathbb{R}$
$$\frac{c \in \mathbb{R}^+ \qquad \Theta; \Delta \vdash I : \mathbb{R}^+}{\Theta; \Delta \vdash c \cdot I : \mathbb{R}^+}$$

I-Times-$\vec{\mathbb{R}}$
$$\frac{c \in \mathbb{R}^+ \qquad \Theta; \Delta \vdash I : \vec{\mathbb{R}}^+}{\Theta; \Delta \vdash c \cdot I : \vec{\mathbb{R}}^+}$$

I-Times-$\mathbb{N}$
$$\frac{c \in \mathbb{N} \qquad \Theta; \Delta \vdash I : \mathbb{N}}{\Theta; \Delta \vdash c \cdot I : \mathbb{N}}$$

I-Shift
$$\frac{\Theta; \Delta \vdash I : \vec{\mathbb{R}}^+}{\Theta; \Delta \vdash \lhd I : \vec{\mathbb{R}}^+}$$

I-Lam
$$\frac{\Theta, i : bS; \Delta \vdash I : S}{\Theta; \Delta \vdash \lambda i : bS.I : bS \to S}$$

I-App
$$\frac{\Theta; \Delta \vdash I : bS \to S \qquad \Theta; \Delta \vdash J : bS}{\Theta; \Delta \vdash I\ J : S}$$

I-Sum
$$\frac{\Theta; \Delta \vdash I_0 : \mathbb{N} \qquad \Theta; \Delta \vdash I_1 : \mathbb{N} \qquad \Theta, i : \mathbb{N}; \Delta, I_0 \leq i < I_1 \vdash J : bS}{\Theta; \Delta \vdash \sum_{i=I_0}^{I_1} J : bS}$$

I-ConstVec
$$\frac{\Theta; \Delta \vdash I : \mathbb{R}^+}{\Theta; \Delta \vdash \mathtt{const}(I) : \vec{\mathbb{R}}^+}$$

I-Vec-Lit
$$\frac{\cdot; \cdot \vDash \bigwedge_i c_i \geq 0}{\Theta; \Delta \vdash (c_0, \ldots, c_k) : \mathbb{R}^+}$$

I-Nat-Lit
$$\frac{\cdot; \cdot \vDash n \geq 0}{\Theta; \Delta \vdash n : \mathbb{N}}$$

I-PosReal-Lit
$$\frac{\cdot; \cdot \vDash r \geq 0.0}{\Theta; \Delta \vdash r : \mathbb{R}^+}$$

2.3. Constraint Wellformedness.

$$\textsc{C-Top}\ \frac{}{\Theta;\Delta \vdash \top\ \texttt{wf}} \qquad \textsc{C-Bot}\ \frac{}{\Theta;\Delta \vdash \bot\ \texttt{wf}} \qquad \textsc{C-Conj}\ \frac{\Theta;\Delta \vdash \Phi_1\ \texttt{wf} \qquad \Theta;\Delta \vdash \Phi_2\ \texttt{wf}}{\Theta;\Delta \vdash \Phi_1 \wedge \Phi_2\ \texttt{wf}}$$

$$\textsc{C-Disj}\ \frac{\Theta;\Delta \vdash \Phi_1\ \texttt{wf} \qquad \Theta;\Delta \vdash \Phi_2\ \texttt{wf}}{\Theta;\Delta \vdash \Phi_1 \vee \Phi_2\ \texttt{wf}} \qquad \textsc{C-Impl}\ \frac{\Theta;\Delta \vdash \Phi_1\ \texttt{wf} \qquad \Theta;\Delta,\Phi_1 \vdash \Phi_2\ \texttt{wf}}{\Theta;\Delta \vdash \Phi_1 \rightarrow \Phi_2\ \texttt{wf}}$$

$$\textsc{C-Forall}\ \frac{\Theta, i : S;\Delta \vdash \Phi\ \texttt{wf}}{\Theta;\Delta \vdash \forall i : S.\Phi\ \texttt{wf}} \qquad \textsc{C-Exists}\ \frac{\Theta, i : S;\Delta,\Phi \vdash \Phi\ \texttt{wf}}{\Theta;\Delta \vdash \exists i : S.\Phi\ \texttt{wf}} \qquad \textsc{C-Leq}\ \frac{\Theta;\Delta \vdash I : bS \qquad \Theta;\Delta \vdash J : bS}{\Theta;\Delta \vdash I \leq J\ \texttt{wf}}$$

$$\textsc{C-Lt}\ \frac{\Theta;\Delta \vdash I : bS \qquad \Theta;\Delta \vdash J : bS}{\Theta;\Delta \vdash I < J\ \texttt{wf}} \qquad \textsc{C-Eq}\ \frac{\Theta;\Delta \vdash I : bS \qquad \Theta;\Delta \vdash J : bS}{\Theta;\Delta \vdash I = J\ \texttt{wf}}$$

2.4. Kind Assignment.

$$\textsc{K-Var}\ \frac{\alpha : K \in \Psi}{\Psi;\Theta;\Delta \vdash \alpha : K} \qquad \textsc{K-Unit}\ \frac{}{\Psi;\Theta;\Delta \vdash 1 : \star} \qquad \textsc{K-Arr}\ \frac{\Psi;\Theta;\Delta \vdash \tau_1 : \star \qquad \Psi;\Theta;\Delta \vdash \tau_2 : \star}{\Psi;\Theta;\Delta \vdash \tau_1 \multimap \tau_2 : \star}$$

$$\textsc{K-Tensor}\ \frac{\Psi;\Theta;\Delta \vdash \tau_1 : \star \qquad \Psi;\Theta;\Delta \vdash \tau_2 : \star}{\Psi;\Theta;\Delta \vdash \tau_1 \otimes \tau_2 : \star} \qquad \textsc{K-With}\ \frac{\Psi;\Theta;\Delta \vdash \tau_1 : \star \qquad \Psi;\Theta;\Delta \vdash \tau_2 : \star}{\Psi;\Theta;\Delta \vdash \tau_1 \& \tau_2 : \star}$$

$$\textsc{K-Sum}\ \frac{\Psi;\Theta;\Delta \vdash \tau_1 : \star \qquad \Psi;\Theta;\Delta \vdash \tau_2 : \star}{\Psi;\Theta;\Delta \vdash \tau_1 \oplus \tau_2 : \star} \qquad \textsc{K-Bang}\ \frac{\Psi;\Theta;\Delta \vdash \tau : \star}{\Psi;\Theta;\Delta \vdash !\tau : \star} \qquad \textsc{K-IForall}\ \frac{\Psi;\Theta, i : S;\Delta \vdash \tau : \star}{\Psi;\Theta;\Delta \vdash \forall i : S.\tau : \star}$$

$$\textsc{K-IExists}\ \frac{\Psi;\Theta, i : S;\Delta \vdash \tau : \star}{\Psi;\Theta;\Delta \vdash \exists i : S.\tau : \star} \qquad \textsc{K-TForall}\ \frac{\Psi, \alpha : K;\Theta;\Delta \vdash \tau : \star}{\Psi;\Theta;\Delta \vdash \forall \alpha : K.\tau : \star}$$

K-List
$$\frac{\Theta; \Delta \vdash I : \mathbb{N} \qquad \Psi; \Theta; \Delta \vdash \tau : \star}{\Psi; \Theta; \Delta \vdash L^{I} \tau : \star}$$

K-Conj
$$\frac{\Theta; \Delta \vdash \Phi \ \texttt{wf} \qquad \Psi; \Theta; \Delta \vdash \tau : \star}{\Psi; \Theta; \Delta \vdash \Phi \& \tau : \star}$$

K-Impl
$$\frac{\Theta; \Delta \vdash \Phi \ \texttt{wf} \qquad \Psi; \Theta; \Delta, \Phi \vdash \tau : \star}{\Psi; \Theta; \Delta \vdash \Phi \implies \tau : \star}$$

K-Monad
$$\frac{\Theta; \Delta \vdash I : \mathbb{N} \qquad \Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^{+}} \qquad \Psi; \Theta; \Delta \vdash \tau : \star}{\Psi; \Theta; \Delta \vdash \mathbb{M}(I, \vec{p}) \tau : \star}$$

K-Pot
$$\frac{\Theta; \Delta \vdash I : \mathbb{N} \qquad \Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^{+}} \qquad \Psi; \Theta; \Delta \vdash \tau : \star}{\Psi; \Theta; \Delta \vdash [I|\vec{p}] \tau : \star}$$

K-ConstPot
$$\frac{\Theta; \Delta \vdash I : \mathbb{R}^{+} \qquad \Psi; \Theta; \Delta \vdash \tau : \star}{\Psi; \Theta; \Delta \vdash [I] \ \tau : \star}$$

K-FamLam
$$\frac{\Psi; \Theta, i : S; \Delta \vdash \tau : K}{\Psi; \Theta; \Delta \vdash \lambda i : S.\tau : S \to K}$$

K-FamApp
$$\frac{\Psi; \Theta; \Delta \vdash \tau : S \to K \qquad \Theta; \Delta \vdash I : S}{\Psi; \Theta; \Delta \vdash \tau \ I : K}$$

2.5. Subtyping.

S-Refl
$$\frac{}{\Psi; \Theta; \Delta \vdash \tau <: \tau : K}$$

S-Trans
$$\frac{\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : K \qquad \Psi; \Theta; \Delta \vdash \tau_2 <: \tau_3 : K}{\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_3 : K}$$

S-Arr
$$\frac{\Psi; \Theta; \Delta \vdash \tau_1' <: \tau_1 : \star \qquad \Psi; \Theta; \Delta \vdash \tau_2 <: \tau_2' : \star}{\Psi; \Theta; \Delta \vdash \tau_1 \multimap \tau_2 <: \tau_1' \multimap \tau_2' : \star}$$

S-Tensor
$$\frac{\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_1' : \star \qquad \Psi; \Theta; \Delta \vdash \tau_2 <: \tau_2' : \star}{\Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 <: \tau_1' \otimes \tau_2' : \star}$$

S-With
$$\frac{\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_1' : \star \qquad \Psi; \Theta; \Delta \vdash \tau_2 <: \tau_2' : \star}{\Psi; \Theta; \Delta \vdash \tau_1 \& \tau_2 <: \tau_1' \& \tau_2' : \star}$$

S-Sum
$$\frac{\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_1' : \star \qquad \Psi; \Theta; \Delta \vdash \tau_2 <: \tau_2' : \star}{\Psi; \Theta; \Delta \vdash \tau_1 \oplus \tau_2 <: \tau_1' \oplus \tau_2' : \star}$$

S-Bang
$$\frac{\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : \star}{\Psi; \Theta; \Delta \vdash !\tau_1 <: !\tau_2 : \star}$$

S-IForall
$$\frac{\Psi; \Theta, i : S; \Delta \vdash \tau_1 <: \tau_2 : \star}{\Psi; \Theta; \Delta \vdash \forall i : S.\tau_1 <: \forall i : S.\tau_2 : \star}$$

S-IExists
$$\frac{\Psi; \Theta, i : S; \Delta \vdash \tau_1 <: \tau_2 : \star}{\Psi; \Theta; \Delta \vdash \exists i : S.\tau_1 <: \exists i : S.\tau_2 : \star}$$

S-TForall
$$\frac{\Psi, \alpha : K; \Theta; \Delta \vdash \tau_1 <: \tau_2 : \star}{\Psi; \Theta; \Delta \vdash \forall \alpha : K.\tau_1 <: \forall \alpha : K.\tau_2 : \star}$$

S-List

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : \star \qquad \Theta;\Delta \vDash I = J}{\Psi;\Theta;\Delta \vdash L^I \tau_1 <: L^J \tau_2 : \star}$$

S-Impl

$$\frac{\Psi;\Theta;\Delta,\Phi_2 \vdash \tau_1 <: \tau_2 : \star \qquad \Theta;\Delta \vDash \Phi_2 \to \Phi_1}{\Psi;\Theta;\Delta \vdash \Phi_1 \implies \tau_1 <: \Phi_2 \implies \tau_2 : \star}$$

S-Conj

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : \star \Rightarrow \qquad \Theta;\Delta \vDash \Phi_1 \to \Phi_2}{\Psi;\Theta;\Delta \vdash \Phi_1 \& \tau_1 <: \Phi_2 \& \tau_2 : \star}$$

S-Monad

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : \star \qquad \Theta;\Delta \vDash I = J \qquad \Theta;\Delta \vDash \vec{q} \leq \vec{p}}{\Psi;\Theta;\Delta \vdash \mathbb{M}(I,\vec{q})\tau_1 <: \mathbb{M}(J,\vec{p})\tau_2 : \star}$$

S-Pot

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : \star \qquad \Theta;\Delta \vDash I = J \qquad \Theta;\Delta \vDash \vec{p} \leq \vec{q}}{\Psi;\Theta;\Delta \vdash [I|\vec{q}]\tau_1 <: [J|\vec{p}]\tau_2 : \star}$$

S-ConstPot

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : \star \qquad \Theta;\Delta \vDash J \leq I}{\Psi;\Theta;\Delta \vdash [I]\tau_1 <: [J]\tau_2 : \star}$$

S-FamLam

$$\frac{\Psi;\Theta,i:S;\Delta \vdash \tau_1 <: \tau_2 : K}{\Psi;\Theta;\Delta \vdash \lambda i:S.\tau_1 <: \lambda i:S.\tau_2 : S \to K}$$

S-FamApp

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : S \to K \qquad \Theta;\Delta \vDash I = J}{\Psi;\Theta;\Delta \vdash \tau_1\ I <: \tau_2\ J : K}$$

S-Fam-Beta1

$$\frac{}{\Psi;\Theta;\Delta \vdash (\lambda i:S.\tau)\ J <: \tau[J/i] : K}$$

S-Fam-Beta2

$$\frac{}{\Psi;\Theta;\Delta \vdash \tau[J/i] <: (\lambda i:S.\tau)\ J : K}$$

2.6. Context Subsumption.

CS-Emp

$$\frac{}{\Psi;\Theta;\Delta \vdash \Gamma \sqsubseteq \cdot}$$

CS-Var

$$\frac{x:\tau' \in \Gamma \qquad \Psi;\Theta;\Delta \vdash \tau' <: \tau \qquad \Psi;\Theta;\Delta \vdash \Gamma \setminus \{x:\tau'\} \sqsubseteq \Gamma'}{\Psi;\Theta;\Delta \vdash \Gamma \sqsubseteq \Gamma', x:\tau}$$

2.7. Type Assignment.

T-Var-1

$$\frac{x:\tau \in \Gamma}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash x:\tau}$$

T-Var-2

$$\frac{x:\tau \in \Omega}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash x:\tau}$$

T-Unit

$$\frac{}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash ():1}$$

T-Base
$$\frac{}{\Psi;\Theta;\Delta;\Omega;\Gamma\vdash c:b}$$

T-Absurd
$$\frac{\Theta;\Delta\vDash\bot}{\Psi;\Theta;\Delta;\Omega;\Gamma\vdash \texttt{absurd}:\tau}$$

T-Nil
$$\frac{}{\Psi;\Theta;\Delta;\Omega;\Gamma\vdash \texttt{nil}:L^0\tau}$$

T-Cons
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma_1\vdash e_1:\tau \qquad \Psi;\Theta;\Omega;\Gamma_2\vdash e_2:L^I\tau}{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2\vdash e_1::e_2:L^{I+1}\tau}$$

T-Match
$$\frac{\begin{array}{c}\Psi;\Theta;\Delta;\Omega;\Gamma_1\vdash e:L^I\tau\\ \Psi;\Theta;\Delta,I=0;\Omega;\Gamma_2\vdash e_1:\tau' \qquad \Psi;\Theta;\Delta,I\geq 1;\Omega;\Gamma_2,h:\tau,t:L^{I-1}\tau\vdash e_2:\tau'\end{array}}{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2\vdash \texttt{match}(e,e_1,h.t.e_2):\tau'}$$

T-ExistI
$$\frac{\Theta;\Delta\vdash I:S \qquad \Psi;\Theta;\Delta;\Omega;\Gamma\vdash e:\tau[I/i]}{\Psi;\Theta;\Delta;\Omega;\Gamma\vdash \texttt{pack}[I](e):\exists i:S.\tau}$$

T-ExistE
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma_1\vdash e:\exists i:S.\tau \qquad \Psi;\Theta,i:S;\Delta;\Omega;\Gamma_2,x:\tau\vdash e':\tau'}{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2\vdash \texttt{unpack}\ (i,x)=e\ \texttt{in}\ e':\tau'}$$

T-Lam
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma,x:\tau_1\vdash e:\tau_2}{\Psi;\Theta;\Delta;\Omega;\Gamma\vdash \lambda x.e:\tau_1\multimap\tau_2}$$

T-App
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma_1\vdash e_1:\tau_1\multimap\tau_2 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_2\vdash e_2:\tau_1}{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2\vdash e_1\,e_2:\tau_2}$$

T-TensorI
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma_1\vdash e_1:\tau_1 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_2\vdash e_2:\tau_2}{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2\vdash \langle\!\langle e_1,e_2\rangle\!\rangle;\tau_1\otimes\tau_2}$$

T-TensorE
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma_1\vdash e:\tau_1\otimes\tau_2 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_2,x:\tau_1,y:\tau_2\vdash e':\tau'}{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2\vdash \texttt{let}\ \langle\!\langle x,y\rangle\!\rangle=e\ \texttt{in}\ e':\tau'}$$

T-WithI
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 : \tau_1 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_2 : \tau_2}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash (e_1, e_2) : \tau_1 \& \tau_2}$$

T-Fst
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \tau_1 \& \tau_2}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{fst}(e) : \tau_1}$$

T-Snd
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \tau_1 \& \tau_2}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{snd}(e) : \tau_2}$$

T-Inl
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \tau_1}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{inl}(e) : \tau_1 \oplus \tau_2}$$

T-Inr
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \tau_2}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{inr}(e) : \tau_1 \oplus \tau_2}$$

T-Case
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e : \tau_1 \oplus \tau_2 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_2, x : \tau_1 \vdash e_1 : \tau \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_2, y : \tau_2 \vdash e_2 : \tau}{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash \texttt{case}(e, x.e_1, y.e_2) : \tau}$$

T-ExpI
$$\frac{\Psi;\Theta;\Delta;\Omega;\cdot \vdash e : \tau}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash !e : !\tau}$$

T-ExpE
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e : !\tau \qquad \Psi;\Theta;\Delta;\Omega, x : \tau;\Gamma_2 \vdash e' : \tau'}{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash \texttt{let } !x = e \texttt{ in } e' : \tau'}$$

T-TAbs
$$\frac{\Psi, \alpha : K;\Theta;\Delta;\Omega;\Gamma \vdash e : \tau}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \Lambda\alpha.e : \forall\alpha : K.\tau}$$

T-TApp
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \forall\alpha : K.\tau \qquad \Psi;\Theta;\Delta \vdash \tau' : K}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e[\tau'] : \tau[\tau'/\alpha]}$$

T-IAbs
$$\frac{\Psi;\Theta, i : S;\Delta;\Omega;\Gamma \vdash e : \tau}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \Lambda i.e : \forall i : S.\tau}$$

T-IApp
$$\frac{\Theta;\Delta \vdash I : S \qquad \Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \forall i : S.\tau}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e[I] : \tau[I/i]}$$

T-Fix
$$\frac{\Psi;\Theta;\Delta;\Omega, x : \tau;\cdot \vdash e : \tau}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{fix } x.e : \tau}$$

T-CImpI
$$\frac{\Psi;\Theta;\Delta,\Phi';\Omega;\Gamma \vdash e : \tau}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \Lambda.e : (\Phi' \Rightarrow \tau)}$$

T-CImpE
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \Phi' \Rightarrow \tau \qquad \Theta;\Delta \vDash \Phi'}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e\{\} : \tau}$$

T-CAndI
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \tau \qquad \Theta;\Delta \vDash \Phi'}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash < e > : \Phi' \& \tau}$$

T-CAndE
$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e : \Phi' \& \tau \qquad \Psi;\Theta;\Delta,\Phi';\Omega;\Gamma_2, x : \tau \vdash e' : \tau'}{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash \texttt{clet } x = e \texttt{ in } e' : \tau'}$$

T-Tick
$$\frac{\Theta; \Delta \vdash I : \mathbb{N} \qquad \Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+}}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{tick}[I|\vec{p}] : \mathbb{M}\,(I, \vec{p})\,1}$$

T-Ret
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{ret}\ e : \mathbb{M}\,(I, \vec{0})\,\tau}$$

T-Bind
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : \mathbb{M}\,(I, \vec{p})\,\tau_1 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau_1 \vdash e_2 : \mathbb{M}\,(I, \vec{q})\,\tau_2}{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{bind}\ \ x = e_1\ \ \texttt{in}\ \ e_2 : \mathbb{M}\,(I, \vec{p} + \vec{q})\,\tau_2}$$

T-Release
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : [I|\vec{q}]\tau_1 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau \vdash e_2 : \mathbb{M}\,(I, \vec{p} + \vec{q})\,\tau_2}{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{release}\ \ x = e_1\ \ \texttt{in}\ \ e_2 : \mathbb{M}\,(I, \vec{p})\,\tau_2}$$

T-Store
$$\frac{\Theta; \Delta \vdash I : \mathbb{N} \qquad \Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \qquad \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{store}[I|\vec{p}](e) : \mathbb{M}\,(I, \vec{p})\,([I|\vec{p}]\,\tau)}$$

T-StoreConst
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \qquad \Theta; \Delta \vdash I : \mathbb{N}}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{store}[I](e) : \mathbb{M}\,(K, \texttt{const}(I))\,([I]\,\tau)}$$

T-ReleaseConst
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : [J]\tau_1 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau \vdash e_2 : \mathbb{M}\,(I, \vec{p} + \texttt{const}(J))\,\tau_2}{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{release}\ \ x = e_1\ \ \texttt{in}\ \ e_2 : \mathbb{M}\,(I, \vec{p})\,\tau_2}$$

T-Shift
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \mathbb{M}\,(I - 1, \lhd\,\vec{p})\,\tau \qquad \Theta; \Delta \vDash I \geq 1}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{shift}(e) : \mathbb{M}\,(I, \vec{p})\,\tau}$$

T-Sub
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau' \qquad \Psi; \Theta; \Delta \vdash \tau' <: \tau}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau}$$

T-Weaken
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \qquad \Theta; \Delta \vdash \Omega' \sqsubseteq \Omega \qquad \Theta; \Delta \vdash \Gamma' \sqsubseteq \Gamma}{\Psi; \Theta; \Delta; \Omega'; \Gamma' \vdash e : \tau}$$

2.8. Normal Forms and Normalization.

$$\textsc{N-Var}\ \frac{}{\alpha\ \texttt{ne}} \qquad \textsc{N-Unit}\ \frac{}{1\ \texttt{nf}} \qquad \textsc{N-Arr}\ \frac{\tau_1\ \texttt{nf} \quad \tau_2\ \texttt{nf}}{\tau_1 \multimap \tau_2\ \texttt{nf}} \qquad \textsc{N-Tensor}\ \frac{\tau_1\ \texttt{nf} \quad \tau_2\ \texttt{nf}}{\tau_1 \otimes \tau_2\ \texttt{nf}} \qquad \textsc{N-With}\ \frac{\tau_1\ \texttt{nf} \quad \tau_2\ \texttt{nf}}{\tau_1 \& \tau_2\ \texttt{nf}} \qquad \textsc{N-Sum}\ \frac{\tau_1\ \texttt{nf} \quad \tau_2\ \texttt{nf}}{\tau_1 \oplus \tau_2\ \texttt{nf}}$$

$$\textsc{N-Bang}\ \frac{\tau\ \texttt{nf}}{!\tau\ \texttt{nf}} \qquad \textsc{N-IForall}\ \frac{\tau\ \texttt{nf}}{\forall i : S.\tau\ \texttt{nf}} \qquad \textsc{N-IExists}\ \frac{\tau\ \texttt{nf}}{\exists i : S.\tau\ \texttt{nf}} \qquad \textsc{N-TForall}\ \frac{\tau\ \texttt{nf}}{\forall \alpha : K.\tau\ \texttt{nf}} \qquad \textsc{N-List}\ \frac{\tau\ \texttt{nf}}{L^I \tau\ \texttt{nf}} \qquad \textsc{N-Conj}\ \frac{\tau\ \texttt{nf}}{\Phi \& \tau\ \texttt{nf}} \qquad \textsc{N-Impl}\ \frac{\tau\ \texttt{nf}}{\Phi \implies \tau\ \texttt{nf}}$$

$$\textsc{N-Monad}\ \frac{\tau\ \texttt{nf}}{\mathbb{M}(I, \vec{p})\tau\ \texttt{nf}} \qquad \textsc{N-Pot}\ \frac{\tau\ \texttt{nf}}{[I|\vec{p}]\tau\ \texttt{nf}} \qquad \textsc{N-ConstPot}\ \frac{\tau\ \texttt{nf}}{[I]\tau\ \texttt{nf}} \qquad \textsc{N-FamLam}\ \frac{\tau\ \texttt{nf}}{\lambda i : S.\tau\ \texttt{nf}} \qquad \textsc{N-FamApp}\ \frac{\tau\ \texttt{ne}}{(\tau\ I)\ \texttt{ne}} \qquad \textsc{N-NeNf}\ \frac{\tau\ \texttt{ne}}{\tau\ \texttt{nf}}$$

$$\begin{aligned}
\texttt{eval}(\alpha) &= \alpha \\
\texttt{eval}(1) &= 1 \\
\texttt{eval}(\tau_1 \multimap \tau_2) &= \texttt{eval}(\tau_1) \multimap \texttt{eval}(\tau_2) \\
\texttt{eval}(\tau_1 \otimes \tau_2) &= \texttt{eval}(\tau_1) \otimes \texttt{eval}(\tau_2) \\
\texttt{eval}(\tau_1 \& \tau_2) &= \texttt{eval}(\tau_1) \& \texttt{eval}(\tau_2) \\
\texttt{eval}(\tau_1 \oplus \tau_2) &= \texttt{eval}(\tau_1) \oplus \texttt{eval}(\tau_2) \\
\texttt{eval}(!\tau) &= !\texttt{eval}(\tau) \\
\texttt{eval}(\forall i : S.\tau) &= \forall i : S.\texttt{eval}(\tau) \\
\texttt{eval}(\exists i : S.\tau) &= \exists i : S.\texttt{eval}(\tau) \\
\texttt{eval}(\forall \alpha : K.\tau) &= \forall \alpha : K.\texttt{eval}(\tau) \\
\texttt{eval}(L^I \tau) &= L^I(\texttt{eval}(\tau)) \\
\texttt{eval}(\Phi \& \tau) &= \Phi \& \texttt{eval}(\tau) \\
\texttt{eval}(\Phi \implies \tau) &= \Phi \implies \texttt{eval}(\tau) \\
\texttt{eval}(\mathbb{M}(I, \vec{p})\tau) &= \mathbb{M}(I, \vec{p})(\texttt{eval}(\tau)) \\
\texttt{eval}([I|\vec{p}]\tau) &= [I|\vec{p}](\texttt{eval}(\tau))
\end{aligned}$$

$$\texttt{eval}([I]\tau) = [I](\texttt{eval}(\tau))$$

$$\texttt{eval}(\lambda i : S.\tau) = \lambda i : S.\texttt{eval}(\tau)$$

$$\texttt{eval}(\tau\ I) = \begin{cases} \tau'[I/i], & \texttt{eval}(\tau) = \lambda i : S.\tau' \\ \texttt{eval}(\tau)\ I & \texttt{eval}(\tau)\ \texttt{ne} \end{cases}$$

3. Rules of biλ-Amor

3.1. Algorithmic Wellformedness Judgments.

AWF-CCtxE
$$\frac{}{\Theta \vdash \cdot\ \mathtt{wf} \Rightarrow \top}$$

AWF-CCtxNe
$$\frac{\Theta \vdash \Delta\ \mathtt{wf} \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash \Phi\ \mathtt{wf} \Rightarrow \Phi_2}{\Theta \vdash \Delta, \Phi\ \mathtt{wf} \Rightarrow \Phi_1 \wedge (\bigwedge \Delta \to \Phi_2)}$$

AWF-TCtxE
$$\frac{}{\Psi; \Theta; \Delta \vdash \cdot\ \mathtt{wf} \Rightarrow \top}$$

AWF-TCtxNE
$$\frac{\Psi; \Theta; \Delta \vdash \Gamma\ \mathtt{wf} \Rightarrow \Phi_1 \qquad \Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_2}{\Psi; \Theta; \Delta \vdash \Gamma, x : \tau\ \mathtt{wf} \Rightarrow \Phi_1 \wedge \Phi_2}$$

3.2. Algorithmic Sort Checking/Inference.

AI-Var
$$\frac{i : S \in \Theta}{\Theta; \Delta \vdash i : S \Rightarrow \top}$$

AI-Plus
$$\frac{\Theta; \Delta \vdash I : bS \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash J : bS \Rightarrow \Phi_2}{\Theta; \Delta \vdash I + J : bS \Rightarrow \Phi_1 \wedge \Phi_2}$$

AI-Minus
$$\frac{\Theta; \Delta \vdash I : bS \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash J : bS \Rightarrow \Phi_2}{\Theta; \Delta \vdash I - J : bS \Rightarrow \Phi_1 \wedge \Phi_2 \wedge (I \geq J)}$$

AI-Times-$\mathbb{R}$
$$\frac{c \in \mathbb{R}^+ \qquad \Theta; \Delta \vdash I : \mathbb{R}^+ \Rightarrow \Phi}{\Theta; \Delta \vdash c \cdot I : \mathbb{R}^+ \Rightarrow \Phi}$$

AI-Times-$\vec{\mathbb{R}}$
$$\frac{c \in \mathbb{R}^+ \qquad \Theta; \Delta \vdash I : \vec{\mathbb{R}}^+ \Rightarrow \Phi}{\Theta; \Delta \vdash c \cdot I : \vec{\mathbb{R}}^+ \Rightarrow \Phi}$$

AI-Times-$\mathbb{N}$
$$\frac{c \in \mathbb{N} \qquad \Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi}{\Theta; \Delta \vdash c \cdot I : \mathbb{N} \Rightarrow \Phi}$$

AI-Shift
$$\frac{\Theta; \Delta \vdash I : \vec{\mathbb{R}}^+ \Rightarrow \Phi}{\Theta; \Delta \vdash \triangleleft I : \vec{\mathbb{R}}^+ \Rightarrow \Phi}$$

AI-Lam
$$\frac{\Theta, i : bS; \Delta \vdash I : S \Rightarrow \Phi}{\Theta; \Delta \vdash \lambda i : bS.I : bS \to S \Rightarrow \forall i : S.\Phi}$$

AI-App
$$\frac{\Theta; \Delta \vdash I : bS \to S \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash J : bS \Rightarrow \Phi_2}{\Theta; \Delta \vdash I\ J : S \Rightarrow \Phi_1 \wedge \Phi_2}$$

AI-Sum
$$\frac{\Theta; \Delta \vdash I_0 : \mathbb{N} \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash I_1 : \mathbb{N} \Rightarrow \Phi_2 \qquad \Theta, i : \mathbb{N}; \Delta, I_0 \leq i < I_1 \vdash J : bS \Rightarrow \Phi_3}{\Theta; \Delta \vdash \sum_{i=I_0}^{I_1} J : bS \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \forall i : \mathbb{N}.(I_0 \leq i < I_1 \to \Phi_3)}$$

AI-ConstVec
$$\frac{\Theta;\Delta \vdash I : \mathbb{R}^+ \Rightarrow \Phi}{\Theta;\Delta \vdash \texttt{const}(I) : \vec{\mathbb{R}^+} \Rightarrow \Phi}$$

AI-Vec-Lit
$$\frac{\Phi = \bigwedge_i c_i \geq 0}{\Theta;\Delta \vdash (c_0,\ldots,c_k) : \mathbb{R}^+ \Rightarrow \Phi}$$

AI-Nat-Lit
$$\frac{}{\Theta;\Delta \vdash n : \mathbb{N} \Rightarrow n \geq 0}$$

AI-PosReal-Lit
$$\frac{}{\Theta;\Delta \vdash r : \mathbb{R}^+ \Rightarrow r \geq 0}$$

3.3. Algorithmic Constraint Wellformedness.

AC-Top
$$\frac{}{\Theta;\Delta \vdash \top\ \texttt{wf} \Rightarrow \top}$$

AC-Bot
$$\frac{}{\Theta;\Delta \vdash \bot\ \texttt{wf} \Rightarrow \top}$$

AC-Conj
$$\frac{\Theta;\Delta \vdash \Phi_1\ \texttt{wf} \Rightarrow \Phi_1' \qquad \Theta;\Delta \vdash \Phi_2\ \texttt{wf} \Rightarrow \Phi_2'}{\Theta;\Delta \vdash \Phi_1 \wedge \Phi_2\ \texttt{wf} \Rightarrow \Phi_1' \wedge \Phi_2'}$$

AC-Disj
$$\frac{\Theta;\Delta \vdash \Phi_1\ \texttt{wf} \Rightarrow \Phi_1' \qquad \Theta;\Delta \vdash \Phi_2\ \texttt{wf} \Rightarrow \Phi_2'}{\Theta;\Delta \vdash \Phi_1 \vee \Phi_2\ \texttt{wf} \Rightarrow \Phi_1' \wedge \Phi_2'}$$

AC-Impl
$$\frac{\Theta;\Delta \vdash \Phi_1\ \texttt{wf} \Rightarrow \Phi_1' \qquad \Theta;\Delta \vdash \Phi_2\ \texttt{wf} \Rightarrow \Phi_2'}{\Theta;\Delta,\Phi_1 \vdash \Phi_1 \to \Phi_2\ \texttt{wf} \Rightarrow \Phi_1' \wedge (\Phi_1 \to \Phi_2')}$$

AC-Forall
$$\frac{\Theta, i : S;\Delta \vdash \Phi\ \texttt{wf} \Rightarrow \Phi'}{\Theta;\Delta \vdash \forall i : S.\Phi\ \texttt{wf} \Rightarrow \forall i : S.\Phi'}$$

AC-Exists
$$\frac{\Theta, i : S;\Delta,\Phi \vdash \Phi\ \texttt{wf} \Rightarrow \Phi'}{\Theta;\Delta \vdash \exists i : S.\Phi\ \texttt{wf} \Rightarrow \forall i : S.(\Phi \to \Phi')}$$

AC-Leq
$$\frac{\Theta;\Delta \vdash I : bS \Rightarrow \Phi_1 \qquad \Theta;\Delta \vdash J : bS \Rightarrow \Phi_2}{\Theta;\Delta \vdash I \leq J\ \texttt{wf} \Rightarrow \Phi_1 \wedge \Phi_2}$$

AC-Lt
$$\frac{\Theta;\Delta \vdash I : bS \Rightarrow \Phi_1 \qquad \Theta;\Delta \vdash J : bS \Rightarrow \Phi_2}{\Theta;\Delta \vdash I < J\ \texttt{wf} \Rightarrow \Phi_1 \wedge \Phi_2}$$

AC-Eq
$$\frac{\Theta;\Delta \vdash I : bS \Rightarrow \Phi_1 \qquad \Theta;\Delta \vdash J : bS \Rightarrow \Phi_2}{\Theta;\Delta \vdash I = J\ \texttt{wf} \Rightarrow \Phi_1 \wedge \Phi_2}$$

3.4. Algorithmic Kind Checking/Inference.

AK-Var
$$\frac{\alpha : K \in \Psi}{\Psi;\Theta;\Delta \vdash \alpha : K \Rightarrow \top}$$

AK-Unit
$$\frac{}{\Psi;\Theta;\Delta \vdash 1 : \star \Rightarrow \top}$$

AK-Arr

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 : \star \Rightarrow \Phi_1 \qquad \Psi;\Theta;\Delta \vdash \tau_2 : \star \Rightarrow \Phi_2}{\Psi;\Theta;\Delta \vdash \tau_1 \multimap \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

AK-Tensor

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 : \star \Rightarrow \Phi_1 \qquad \Psi;\Theta;\Delta \vdash \tau_2 : \star \Rightarrow \Phi_2}{\Psi;\Theta;\Delta \vdash \tau_1 \otimes \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

AK-With

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 : \star \Rightarrow \Phi_1 \qquad \Psi;\Theta;\Delta \vdash \tau_2 : \star \Rightarrow \Phi_2}{\Psi;\Theta;\Delta \vdash \tau_1 \& \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

AK-Sum

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 : \star \Rightarrow \Phi_1 \qquad \Psi;\Theta;\Delta \vdash \tau_2 : \star \Rightarrow \Phi_2}{\Psi;\Theta;\Delta \vdash \tau_1 \oplus \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

AK-Bang

$$\frac{\Psi;\Theta;\Delta \vdash \tau : \star \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash !\tau : \star \Rightarrow \Phi}$$

AK-IForall

$$\frac{\Psi;\Theta, i : S;\Delta \vdash \tau : \star \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \forall i : S.\tau : \star \Rightarrow \forall i : S.\Phi}$$

AK-IExists

$$\frac{\Psi;\Theta, i : S;\Delta \vdash \tau : \star \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \exists i : S.\tau : \star \Rightarrow \forall i : S.\Phi}$$

AK-TForall

$$\frac{\Psi, \alpha : K;\Theta;\Delta \vdash \tau : \star \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \forall \alpha : K.\tau : \star \Rightarrow \Phi}$$

AK-List

$$\frac{\Theta;\Delta \vdash I : \mathbb{N} \Rightarrow \Phi_1 \qquad \Psi;\Theta;\Delta \vdash \tau : \star \Rightarrow \Phi_2}{\Psi;\Theta;\Delta \vdash L^I\tau : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

AK-Conj

$$\frac{\Theta;\Delta \vdash \Phi\ \texttt{wf} \Rightarrow \Phi_1 \qquad \Psi;\Theta;\Delta \vdash \tau : \star \Rightarrow \Phi_2}{\Psi;\Theta;\Delta \vdash \Phi \& \tau : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

AK-Impl

$$\frac{\Theta;\Delta \vdash \Phi\ \texttt{wf} \Rightarrow \Phi_1 \qquad \Psi;\Theta;\Delta, \Phi \vdash \tau : \star \Rightarrow \Phi_2}{\Psi;\Theta;\Delta \vdash \Phi \implies \tau : \star \Rightarrow \Phi_1 \wedge (\Phi \to \Phi_2)}$$

AK-Monad

$$\frac{\Theta;\Delta \vdash I : \mathbb{N} \Rightarrow \Phi_1 \qquad \Theta;\Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \Rightarrow \Phi_2 \qquad \Psi;\Theta;\Delta \vdash \tau : \star \Rightarrow \Phi_3}{\Psi;\Theta;\Delta \vdash \mathbb{M}(I, \vec{p})\tau : \star \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3}$$

AK-Pot

$$\frac{\Theta;\Delta \vdash I : \mathbb{N} \Rightarrow \Phi_1 \qquad \Theta;\Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \Rightarrow \Phi_2 \qquad \Psi;\Theta;\Delta \vdash \tau : \star \Rightarrow \Phi_3}{\Psi;\Theta;\Delta \vdash [I|\vec{p}]\tau : \star \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3}$$

AK-ConstPot

$$\frac{\Theta; \Delta \vdash I : \mathbb{R}^{+} \Rightarrow \Phi_1 \qquad \Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_2}{\Psi; \Theta; \Delta \vdash [I]\, \tau : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

AK-FamLam

$$\frac{\Psi; \Theta, i : S; \Delta \vdash \tau : K \Rightarrow \Phi}{\Psi; \Theta; \Delta \vdash \lambda i : S.\tau : S \to K \Rightarrow \forall i : S.\Phi}$$

AK-FamApp

$$\frac{\Theta; \Delta \vdash I : S \Rightarrow \Phi_1 \qquad \Psi; \Theta; \Delta \vdash \tau : S \to K \Rightarrow \Phi_2}{\Psi; \Theta; \Delta \vdash \tau\ I : K \Rightarrow \Phi_1 \wedge \Phi_2}$$

3.5. Algorithmic Subtyping.

AS-Unit

$$\frac{}{\Psi; \Theta; \Delta \vdash 1 <:_{\mathtt{nf}} 1 : \star \Rightarrow \top}$$

AS-Var

$$\frac{}{\Psi; \Theta; \Delta \vdash \alpha <:_{\mathtt{nf}} \alpha : \star \Rightarrow \top}$$

AS-Arr

$$\frac{\Psi; \Theta; \Delta \vdash \tau_1' <:_{\mathtt{nf}} \tau_1 : \star \Rightarrow \Phi_1 \qquad \Psi; \Theta; \Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_2' : \star \Rightarrow \Phi_2}{\Psi; \Theta; \Delta \vdash \tau_1 \multimap \tau_2 <:_{\mathtt{nf}} \tau_1' \multimap \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

AS-Tensor

$$\frac{\Psi; \Theta; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_1' : \star \Rightarrow \Phi_1 \qquad \Psi; \Theta; \Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_2' : \star \Rightarrow \Phi_2}{\Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 <:_{\mathtt{nf}} \tau_1' \otimes \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

AS-With

$$\frac{\Psi; \Theta; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_1' : \star \Rightarrow \Phi_1 \qquad \Psi; \Theta; \Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_2' : \star \Rightarrow \Phi_2}{\Psi; \Theta; \Delta \vdash \tau_1 \& \tau_2 <:_{\mathtt{nf}} \tau_1' \& \tau_2' \Rightarrow \Phi_1 \wedge \Phi_2}$$

AS-Sum

$$\frac{\Psi; \Theta; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_1' : \star \Rightarrow \Phi_1 \qquad \Psi; \Theta; \Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_2' : \star \Rightarrow \Phi_2}{\Psi; \Theta; \Delta \vdash \tau_1 \oplus \tau_2 <:_{\mathtt{nf}} \tau_1' \oplus \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2}$$

AS-Bang

$$\frac{\Psi; \Theta; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi; \Theta; \Delta \vdash !\tau_1 <:_{\mathtt{nf}} !\tau_2 : \star \Rightarrow \Phi}$$

AS-IForall

$$\frac{\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi; \Theta; \Delta \vdash \forall i : S.\tau_1 <:_{\mathtt{nf}} \forall i : S.\tau_2 : \star \Rightarrow \forall i : S.\Phi}$$

AS-IExists

$$\frac{\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi; \Theta; \Delta \vdash \exists i : S.\tau_1 <:_{\mathtt{nf}} \exists i : S.\tau_2 : \star \Rightarrow \forall i : S.\Phi}$$

AS-TForall

$$\frac{\Psi, \alpha : K; \Theta; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi; \Theta; \Delta \vdash \forall \alpha : K.\tau_1 <:_{\mathtt{nf}} \forall \alpha : K.\tau_2 : \star \Rightarrow \Phi}$$

AS-List

$$\frac{\Psi; \Theta; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi; \Theta; \Delta \vdash L^{I} \tau_1 <:_{\mathtt{nf}} L^{J} \tau_2 : \star \Rightarrow I = J \wedge \Phi}$$

AS-Impl

$$\frac{\Psi;\Theta;\Delta,\Phi_2 \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \Phi_1 \implies \tau_1 <:_{\texttt{nf}} \Phi_2 \implies \tau_2 : \star \Rightarrow (\Phi_2 \to \Phi) \wedge (\Phi_2 \to \Phi_1)}$$

AS-Conj

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \Phi_1 \& \tau_1 <:_{\texttt{nf}} \Phi_2 \& \tau_2 : \star \Rightarrow \Phi \wedge (\Phi_1 \to \Phi_2)}$$

AS-Monad

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \mathbb{M}(I,\vec{q})\tau_1 <:_{\texttt{nf}} \mathbb{M}(J,\vec{p})\tau_2 : \star \Rightarrow (I = J) \wedge (\vec{q} \le \vec{p}) \wedge \Phi}$$

AS-Pot

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash [I|\vec{q}]\tau_1 <:_{\texttt{nf}} [J|\vec{p}]\tau_2 : \star \Rightarrow (I = J) \wedge (\vec{p} \le \vec{q}) \wedge \Phi}$$

AS-ConstPot

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash [I]\tau_1 <:_{\texttt{nf}} [J]\tau_2 : \star \Rightarrow \Phi \wedge (J \le I)}$$

AS-FamLam

$$\frac{\Psi;\Theta, i : S;\Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : K \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \lambda i : S.\tau_1 <:_{\texttt{nf}} \lambda i : S.\tau_2 : S \to K \Rightarrow \forall i : S.\Phi}$$

AS-FamApp

$$\frac{\Psi;\Theta;\Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : S \to K \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \tau_1\ I <:_{\texttt{nf}} \tau_2\ J : K \Rightarrow (I = J) \wedge \Phi}$$

AS-Normalize

$$\frac{\Psi;\Theta;\Delta \vdash \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : K \Rightarrow \Phi}{\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : K \Rightarrow \Phi}$$

3.6. Algorithmic Type Checking/Inference.

AT-Var-1

$$\frac{x : \tau \in \Gamma}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash x \uparrow \tau \Rightarrow \top, \Gamma \setminus \{x : \tau\}}$$

AT-Var-2

$$\frac{x : \tau \in \Omega}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash x \uparrow \tau \Rightarrow \top, \Gamma}$$

AT-Unit

$$\frac{}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash () \uparrow 1 \Rightarrow \top, \Gamma}$$

AT-Base

$$\frac{}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash c \uparrow b \Rightarrow \top, \Gamma}$$

AT-Absurd

$$\frac{}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{absurd} \downarrow \tau \Rightarrow \bot, \Gamma}$$

AT-Nil

$$\frac{}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{nil} \downarrow L^I\tau \Rightarrow I = 0, \Gamma}$$

AT-Cons

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \downarrow \tau \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_2 \downarrow L^{I-1}\tau \Rightarrow \Phi_2, \Gamma_2}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 :: e_2 \downarrow L^I\tau \Rightarrow (I \geq 1) \wedge \Phi_1 \wedge \Phi_2, \Gamma_2}$$

AT-Match

$$\frac{\begin{array}{c}\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow L^I\tau \Rightarrow \Phi_1, \Gamma_1 \\ \Psi;\Theta;\Delta, I = 0;\Omega;\Gamma_1 \vdash e_1 \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2 \qquad \Psi;\Theta;\Delta, I \geq 1;\Omega;\Gamma_1, h:\tau, t:L^{I-1}\tau \vdash e_2 \downarrow \tau' \Rightarrow \Phi_3, \Gamma_3 \\ \Phi_{\texttt{body}} = (I = 0 \to \Phi_2) \wedge (I \geq 1 \to \Phi_3) \qquad \Gamma' = \Gamma_2 \cap (\Gamma_3 \setminus \{h, t\})\end{array}}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{match}(e, e_1, h.t.e_2) \downarrow \tau' \Rightarrow \Phi_1 \wedge \Phi_{\texttt{body}}, \Gamma'}$$

AT-ExistI

$$\frac{\Theta;\Delta \vdash I : S \Rightarrow \Phi_1 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau[I/i] \Rightarrow \Phi_2, \Gamma'}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{pack}[I](e) \downarrow \exists i : S.\tau \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma'}$$

AT-ExistE

$$\frac{\begin{array}{c}\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \exists i : S.\tau \Rightarrow \Phi_1, \Gamma_1 \\ \Psi;\Theta, i : S;\Delta;\Omega;\Gamma_1, x : \tau \vdash e' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2 \qquad \Phi = \Phi_1 \wedge (\forall i : S.\Phi_2)\end{array}}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{unpack}\ \ (i, x) = e\ \ \texttt{in}\ \ e' \downarrow \tau' \Rightarrow \Phi, \Gamma_2 \setminus \{x : \tau\}}$$

AT-Lam

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma, x : \tau_1 \vdash e \downarrow \tau_2, \Rightarrow \Phi, \Gamma'}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \lambda x.e \downarrow \tau_1 \multimap \tau_2 \Rightarrow \Phi, \Gamma' \setminus \{x : \tau_1\}}$$

AT-App

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \uparrow \tau_1 \multimap \tau_2 \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_2 \downarrow \tau_1 \Rightarrow \Phi_2, \Gamma_2}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1\, e_2 \uparrow \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2}$$

AT-TensorI

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \downarrow \tau_1 \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_2 \downarrow \tau_2 \Rightarrow \Phi_2, \Gamma_2}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \langle\!\langle e_1, e_2 \rangle\!\rangle \downarrow \tau_1 \otimes \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2}$$

AT-TensorE

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \tau_1 \otimes \tau_2 \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_1, x:\tau_1, y:\tau_2 \vdash e' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{let}\ \ \langle\!\langle x, y \rangle\!\rangle = e\ \ \texttt{in}\ \ e' \downarrow \tau' \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2 \setminus \{x, y\}}$$

AT-WithI

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \downarrow \tau_1 \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_2 \downarrow \tau_2 \Rightarrow \Phi_2, \Gamma_2}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash (e_1, e_2) \downarrow \tau_1 \& \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_1 \cap \Gamma_2}$$

AT-Fst

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \tau_1 \& \tau_2 \Rightarrow \Phi, \Gamma'}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{fst}(e) \uparrow \tau_1 \Rightarrow \Phi, \Gamma'}$$

AT-Snd

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \tau_1 \& \tau_2 \Rightarrow \Phi, \Gamma'}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{snd}(e) \uparrow \tau_2 \Rightarrow \Phi, \Gamma'}$$

AT-Inl

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau_1 \Rightarrow \Phi, \Gamma'}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{inl}(e) \downarrow \tau_1 \oplus \tau_2 \Rightarrow \Phi, \Gamma'}$$

AT-Inr

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau_2 \Rightarrow \Phi, \Gamma'}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{inr}(e) \downarrow \tau_1 \oplus \tau_2 \Rightarrow \Phi, \Gamma'}$$

AT-Case

$$\frac{\begin{array}{c}\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \tau_1 \oplus \tau_2 \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi;\Theta;\Delta;\Omega;\Gamma_1, x:\tau_1 \vdash e_1 \downarrow \tau \Rightarrow \Phi_2, \Gamma_2 \\ \Psi;\Theta;\Delta;\Omega;\Gamma_1, y:\tau_2 \vdash e_2 \downarrow \tau \Rightarrow \Phi_3, \Gamma_3 \qquad \Gamma' = (\Gamma_2 \setminus \{x:\tau_1\}) \cap (\Gamma_3 \setminus \{y:\tau_2\})\end{array}}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{case}(e, x.e_1, y.e_2) \downarrow \tau \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3, \Gamma'}$$

AT-ExpI

$$\frac{\Psi;\Theta;\Delta;\Omega;\cdot \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash !e \downarrow !\tau \Rightarrow \Phi, \Gamma}$$

AT-ExpE

$$\frac{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow !\tau \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi;\Theta;\Delta;\Omega, x:\tau;\Gamma_1 \vdash e' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2}{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{let}\ \ !x = e\ \ \texttt{in}\ \ e' \downarrow \tau' \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2}$$

AT-TAbs
$$\frac{\Psi, \alpha : K; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda\alpha.e \downarrow \forall\alpha : K.\tau \Rightarrow \Phi, \Gamma'}$$

AT-TApp
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \forall\alpha : K.\tau \Rightarrow \Phi_1, \Gamma' \qquad \Psi; \Theta; \Delta \vdash \tau' : K \Rightarrow \Phi_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e[\tau'] \uparrow \tau[\tau'/\alpha] \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma'}$$

AT-IAbs
$$\frac{\Psi; \Theta, i : S; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda i.e \downarrow \forall i : S.\tau \Rightarrow \forall i : S.\Phi, \Gamma'}$$

AT-IApp
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \forall i : S.\tau \Rightarrow \Phi_1, \Gamma' \qquad \Theta; \Delta \vdash I : S \Rightarrow \Phi_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e[I] \uparrow \tau[I/i] \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma'}$$

AT-Fix
$$\frac{\Psi; \Theta; \Delta; \Omega, x : \tau; \cdot \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{fix}\ \ x.e \downarrow \tau \Rightarrow \Phi, \Gamma}$$

AT-CImpI
$$\frac{\Psi; \Theta; \Delta, \Phi'; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda.e \downarrow (\Phi' \Rightarrow \tau) \Rightarrow (\Phi' \to \Phi), \Gamma'}$$

AT-CImpE
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow (\Phi' \Rightarrow \tau) \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e\{\} \uparrow \tau \Rightarrow \Phi \wedge \Phi', \Gamma'}$$

AT-CAndI
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash < e > \downarrow \Phi' \& \tau \Rightarrow \Phi \wedge \Phi', \Gamma'}$$

AT-CAndE
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \Phi' \& \tau \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi; \Theta; \Delta, \Phi'; \Omega; \Gamma_1, x : \tau \vdash e' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{clet}\ \ x = e\ \ \texttt{in}\ \ e' \downarrow \tau' \Rightarrow \Phi_1 \wedge (\Phi' \to \Phi_2), \Gamma_2 \setminus \{x : \tau\}}$$

AT-Ret
$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{ret}\ \ e \downarrow \mathbb{M}\,(I, \vec{p})\,\tau \Rightarrow \Phi, \Gamma'}$$

AT-Bind
$$\frac{\begin{array}{c}\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 \uparrow \mathbb{M}\,(J, \vec{p})\,\tau_1 \Rightarrow \Phi_1, \Gamma_1 \\ \Psi; \Theta; \Delta; \Omega; \Gamma_1, x : \tau_1 \vdash e_2 \downarrow \mathbb{M}\,(I, \vec{q} - \vec{p})\,\tau_2 \Rightarrow \Phi_2, \Gamma_2 \qquad \Phi = (\vec{q} \geq \vec{p}) \wedge (I = J) \wedge \Phi_1 \wedge \Phi_2\end{array}}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{bind}\ \ x = e_1\ \ \texttt{in}\ \ e_2 \downarrow \mathbb{M}\,(I, \vec{q})\,\tau_2 \Rightarrow \Phi, \Gamma_2 \setminus \{x : \tau_1\}}$$

AT-Tick

$$\frac{\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \Rightarrow \Phi_1}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{tick}[I|\vec{p}] \uparrow \mathbb{M}\,(I, \vec{p})\,1 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma}$$

AT-Release

$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 \uparrow [J|\vec{q}]\tau_1 \Rightarrow \Phi_1, \Gamma_1 \qquad \Psi; \Theta; \Delta; \Omega; \Gamma_1, x : \tau \vdash e_2 \downarrow \mathbb{M}\,(I, \vec{p} + \vec{q})\,\tau_2 \Rightarrow \Phi_2, \Gamma_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{release}\ \ x = e_1\ \ \texttt{in}\ \ e_2 \downarrow \mathbb{M}\,(I, \vec{p})\,\tau_2 \Rightarrow (I = J \wedge \Phi_1 \wedge \Phi_2), \Gamma_2 \setminus \{x\}}$$

AT-Store

$$\frac{\begin{array}{c}\Theta; \Delta \vdash K : \mathbb{N} \Rightarrow \Phi_1 \qquad \Theta; \Delta \vdash \vec{w} : \vec{\mathbb{R}^+} \Rightarrow \Phi_2 \\ \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi_3, \Gamma' \qquad \Phi = \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge (\vec{p} \leq \vec{w} \leq \vec{q}) \wedge (I = J = K)\end{array}}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{store}[K|\vec{w}](e) \downarrow \mathbb{M}\,(I, \vec{q})\,([J|\vec{p}]\,\tau) \Rightarrow \Phi, \Gamma'}$$

AT-StoreConst

$$\frac{\begin{array}{c}\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi_1, \Gamma' \\ \Theta; \Delta \vdash J : \mathbb{R} \Rightarrow \Phi_2 \qquad \Phi = (\texttt{const}(I) \leq \texttt{const}(J) \leq \vec{p}) \wedge \Phi_1 \wedge \Phi_2\end{array}}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{store}[J](e) \downarrow \mathbb{M}\,(K, \vec{p})\,([I]\,\tau) \Rightarrow \Phi, \Gamma'}$$

AT-ReleaseConst

$$\frac{\begin{array}{c}\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 \uparrow [J]\tau_1 \Rightarrow \Phi_1, \Gamma_1 \\ \Psi; \Theta; \Delta; \Omega; \Gamma_1, x : \tau_1 \vdash e_2 \downarrow \mathbb{M}\,(I, \vec{p} + \texttt{const}(J))\,\tau_2 \Rightarrow \Phi_2, \Gamma_2\end{array}}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{release}\ \ x = e_1\ \ \texttt{in}\ \ e_2 \downarrow \mathbb{M}\,(I, \vec{p})\,\tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2 \setminus \{x\}}$$

AT-Shift

$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \mathbb{M}\,(I - 1, \triangleleft\,\vec{q})\,\tau \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{shift}(e) \downarrow \mathbb{M}\,(I, \vec{q})\,\tau \Rightarrow (I \geq 1) \wedge \Phi, \Gamma'}$$

AT-Sub

$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \tau' \Rightarrow \Phi_1, \Gamma' \qquad \Psi; \Theta; \Delta \vdash \tau' <: \tau : \star \Rightarrow \Phi_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma'}$$

AT-Anno

$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash (e : \tau) \uparrow \tau \Rightarrow \Phi, \Gamma'}$$

4. Theorems

THEOREM 5.1. *If* $\Psi;\Theta;\Delta \vdash \Gamma$ *wf and* $\Gamma' \subseteq \Gamma$ *then* $\Psi;\Theta;\Delta \vdash \Gamma'$ *wf*

PROOF. Immediate from the fact that $\Psi;\Theta;\Delta \vdash \Gamma$ `wf` exactly when $\Psi;\Theta;\Delta \vdash \tau : \star$ for every $x : \tau \in \Gamma$. □

THEOREM 5.2. *If* $\Theta \vdash \Delta, \Phi$ *wf then* $\Theta \vdash \Delta$ *wf*

THEOREM A.1 (Raw Admissibility of Weakening for Sort Checking). *If* $\Theta;\Delta \vdash I : S$ *and* $\Theta' \supseteq \Theta$*, then* $\Theta';\Delta \vdash I : S$

THEOREM A.2 (Raw Admissibility of Weakening for Constraint Well-Formedness). *If* $\Theta;\Delta \vdash \Phi$ *wf and* $\Theta' \supseteq \Theta$*, then* $\Theta';\Delta \vdash \Phi$ *wf*

THEOREM A.3 (Admissibility of Weakening for Constraint Context Well-Formedness). *If* $\Theta \vdash \Delta$ *wf and* $\Theta' \supseteq \Theta$*, then* $\Theta' \vdash \Delta$ *wf*

THEOREM A.4 (Admissibility of Weakening for Sort Checking). *If* $\Theta;\Delta \vdash_p I : S$ *and* $\Theta' \supseteq \Theta$*, then* $\Theta';\Delta \vdash_p I : S$

THEOREM A.5 (Admissibility of Weakening for Constraint Well-Formedness). *If* $\Theta;\Delta \vdash_p \Phi$ *wf and* $\Theta' \supseteq \Theta$*, then* $\Theta';\Delta \vdash_p \Phi$ *wf*

THEOREM A.6 (Raw Index Substitution for Constraint-Well-Formedness). *If* $\Theta, j : S_1;\Delta \vdash I : S_2$ *and* $\Theta;\Delta \vdash_p J : S_1$*, then* $\Theta;\Delta[J/j] \vdash I[J/j] : S_2$

THEOREM A.7 (Index Substitution for Sort Checking). *If* $\Theta, j : S_1;\Delta \vdash_p I : S_2$ *and* $\Theta;\Delta \vdash_p J : S_1$ *then* $\Theta;\Delta \vdash_p I[J/j] : S_2$

PROOF. Immediate by Theorem A.6, using the fact that $\Theta \vdash \Delta$ `wf`. □

THEOREM A.8 (Raw Index Substitution for Constraint Well-formedness). *If* $\Theta, i : S;\Delta \vdash \Phi$ *wf and* $\Theta;\Delta \vdash I : S$ *then* $\Theta;\Delta[I/i] \vdash \Phi[I/i]$ *wf*

THEOREM A.9 (Index Substitution for Constraint Well-Formedness). *If* $\Theta, i : S;\Delta \vdash_p \Phi$ *wf and* $\Theta;\Delta \vdash_p I : S$ *then* $\Theta;\Delta \vdash_p \Phi[I/i]$ *wf.*

Proof. Immediate by Theorem A.8, using the fact that $\Theta \vdash \Delta\ \texttt{wf}$. □

Theorem A.10 (Admissibility of Weakening for Type Formation). *If* $\Psi; \Theta; \Delta \vdash_p \tau : K$, $\Psi' \supseteq \Psi$, *and* $\Theta' \supseteq \Theta$, *then* $\Psi'; \Theta'; \Delta \vdash_p \tau : K$.

Theorem A.11 (Admissibility of Weakening for Term Context Wellformedness). *If* $\Psi; \Theta; \Delta \vdash \Gamma\ \mathit{wf}$, $\Psi' \supseteq \Psi$, *and* $\Theta' \supseteq \Theta$, *then* $\Psi'; \Theta'; \Delta \vdash \Gamma\ \mathit{wf}$.

Theorem 5.4. *If* $\Psi; \Theta, i : S; \Delta \vdash_p \tau : K$ *and* $\Theta; \Delta \vdash_p I : S$ *then* $\Psi; \Theta; \Delta \vdash_p \tau[I/i] : K$

Theorem A.12 (Admissibility of Weakening for Subtyping). *Suppose* $\Psi; \Theta; \Delta \vdash_p \tau <: \tau' : K$, $\Psi' \supseteq \Psi$, *and* $\Theta' \supseteq \Theta$. *Then,* $\Psi'; \Theta'; \Delta \vdash_p \tau <: \tau'$.

Theorem 5.5. *Suppose* $\Psi; \Theta; \Delta \vdash \tau : K$ *and* $\Psi; \Theta; \Delta \vdash \tau' : K$. *If* $\Psi'; \Theta'; \Delta \vdash_p \tau <: \tau' : K$ *with* $\Theta' \supseteq \Theta$ *and* $\Psi' \supseteq \Psi$, *then* $\Psi; \Theta; \Delta \vdash_p \tau <: \tau' : K$

Theorem A.13 (Context Subsumption Includes Subset). *If* $\Psi; \Theta; \Delta \vdash_p \Gamma'$ *and* $\Gamma \subseteq \Gamma'$ *as sets, then* $\Psi; \Theta; \Delta \vdash_p \Gamma' \sqsubseteq \Gamma$

Proof. By induction on $|\Gamma|$. If $\Gamma = \varnothing$, this is immediate by CS-Emp. Now suppose $\Gamma = \Gamma'', x : \tau$. Then, since $\Gamma \subseteq \Gamma'$, we have $x : \tau \in \Gamma'$, and $\Gamma'' \subseteq \Gamma' \setminus \{x : \tau\}$. Moreover, by S-Refl, $\Psi; \Theta; \Delta \vdash \tau <: \tau : \star$, and so we are done by IH. □

Theorem 5.6. $\Psi; \Theta; \Delta \vdash \Gamma \sqsubseteq \Gamma'$ *if and only if for all* $x : \tau' \in \Gamma'$, *there is some* τ *so that* $x : \tau \in \Gamma$ *and* $\Psi; \Theta; \Delta \vdash \tau <: \tau' : \star$.

Proof. By an easy induction on $|\Gamma|$. □

Theorem A.14 (Admissibility of Weakening for Context Subsumption). *If* $\Psi; \Theta; \Delta \vdash_p \Gamma \sqsubseteq \Gamma'$ *and* $\Psi' \supseteq \Psi$, $\Theta' \supseteq \Theta$, *and* $\Delta' \supseteq \Delta$, *then* $\Psi'; \Theta'; \Delta' \vdash_p \Gamma \sqsubseteq \Gamma'$

Theorem A.15 (Strengthening for Context Subsumption). *Suppose* $\Psi; \Theta; \Delta \vdash \Gamma, \Gamma'\ \mathit{wf}$. *If* $\Psi'; \Theta'; \Delta \vdash_p \Gamma \sqsubseteq \Gamma'$ *with* $\Theta' \supseteq \Theta$ *and* $\Psi' \supseteq \Psi$, *then* $\Psi; \Theta; \Delta \vdash_p \Gamma \sqsubseteq \Gamma'$.

Proof. Immediate by Theorem 5.6 and Theorem 5.5 □

Theorem 5.7. *Suppose that*

(1) $\Psi; \Theta; \Delta \vdash_p \Gamma_1 \sqsubseteq \Gamma_1'$

(2) $\Psi'; \Theta'; \Delta' \vdash_p \Gamma_2 \sqsubseteq \Gamma_2'$

(3) $\Gamma_1 \supseteq \Gamma_2$ *and* $\Gamma_1' \supseteq \Gamma_2'$

Then, $\Psi; \Theta; \Delta \vdash_p \Gamma_2 \sqsubseteq \Gamma_2'$.

PROOF. By Theorem 5.6, it suffices to show that for every $x : \tau \in \Gamma_2'$, there is some τ' such that $x : \tau' \in \Gamma_2'$, and $\Psi; \Theta; \Delta \vdash \tau' <: \tau : \star$. Suppose $x : \tau \in \Gamma_2'$. Then, $x : \tau \in \Gamma_1'$. Further, since $\Psi'; \Theta'; \Delta' \vdash \Gamma_2 \sqsubseteq \Gamma_2'$, there is some τ' such that $x : \tau' \in \Gamma_2$. But then, $x : \tau' \in \Gamma_1$, and so since $\Psi; \Theta; \Delta \vdash \Gamma_1 \sqsubseteq \Gamma_1'$, we have that $\Psi; \Theta; \Delta \vdash \tau' <: \tau : \star$, by Theorem 5.6. □

4.1. Normalization.

THEOREM 7.2 (Canonical Forms for $S \to K$). *If* $\Psi; \Theta; \Delta \vdash \tau : S \to K$ *and* τ ***nf****, then either:*

(1) $\tau = \lambda i : S.\tau'$ *with* τ' ***nf***

(2) τ ***ne***

PROOF. Inversion on $\Psi; \Theta; \Delta \vdash \tau : S \to K$ and then τ `nf`. □

THEOREM 7.1.

(1) If τ ***ne*** *then* $\tau[I/i]$ ***ne***

(2) If τ ***nf*** *then* $\tau[I/i]$ ***nf***

PROOF. By an easy simultaneous induction. This is only true because we don't require index terms inside a type to be in normal form for the type to be in normal form. □

THEOREM 7.4. $\textit{eval}(\tau[J/i]) = \textit{eval}(\tau)[J/i]$

THEOREM 7.3 (Normalization Theorem). *If* $\Psi; \Theta; \Delta \vdash_p \tau : K$*, then:*

(1) $\Psi; \Theta; \Delta \vdash_p \textit{eval}(\tau) : K$

(2) $\Psi; \Theta; \Delta \vdash_p \tau \equiv \textit{eval}(\tau) : K$

(3) $\textit{eval}(\tau)$ ***nf***

THEOREM A.16.

- *If* τ ***nf****, then* $\textit{eval}(\tau) = \tau$
- *If* τ ***ne****, then* $\textit{eval}(\tau) = \tau$

Theorem A.17 (Raw Soundness of Sort Checking/Inference). *If* $\Theta; \Delta \vdash I : S \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$*, then* $\Theta; \Delta \vdash I : S$

Theorem A.18 (Raw Soundness of Constraint Well-Formedness). *If* $\Theta; \Delta \vdash \Phi\ \mathit{wf} \Rightarrow \Phi'$ *and* $\Theta; \Delta \vDash \Phi'$ *then* $\Theta; \Delta \vdash \Phi\ \ \mathit{wf}$

Theorem 8.1 (Soundness of Index Context Well-Formedness). *If* $\Theta \vdash \Delta\ \mathit{wf} \Rightarrow \Phi$ *and* $\Theta; \cdot \vDash \Phi$*, then* $\Theta \vdash \Delta\ \mathit{wf}$

Proof. By induction on $\Theta \vdash \Delta\ \mathtt{wf} \Rightarrow \Phi$. The base case is immediate. Suppose $\Theta \vdash \Delta, \Phi\ \mathtt{wf} \Rightarrow \Phi_1 \wedge (\bigwedge \Delta \rightarrow \Phi_2)$ with $\Theta; \cdot \vDash \Phi_1 \wedge (\bigwedge \Delta \rightarrow \Phi_2)$, by way of $\Theta \vdash \Delta\ \mathtt{wf} \Rightarrow \Phi_1$ and $\Theta; \Delta \vdash \Phi\ \mathtt{wf} \Rightarrow \Phi_2$. Since $\Theta; \cdot \vDash \Phi_1$, we have by IH that $\Theta \vdash \Delta\ \mathtt{wf}$. By Theorem A.17, since $\Theta; \Delta \vDash \Phi_2$, we have that $\Theta; \Delta \vdash \Phi_2\ \mathtt{wf}$, and so $\Theta \vdash \Delta, \Phi\ \mathtt{wf}$, as required. □

Theorem 8.2 (Soundness of Sort Checking). *If* $\Theta; \Delta \vdash_p I : S \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$*, then* $\Theta; \Delta \vdash_p I : S$

Proof. Immediate by Theorem A.17 and Theorem 8.1 □

Theorem 8.3 (Soundness of Constraint Well-Formedness). *If* $\Theta; \Delta \vdash_p \Phi\ \ \mathit{wf} \Rightarrow \Phi'$ *and* $\Theta; \Delta \vDash \Phi'$ *then* $\Theta; \Delta \vdash_p \Phi\ \ \mathit{wf}$

Proof. Immediate by Theorem A.18 and Theorem 8.1 □

Theorem A.19 (Raw Soundness of Kind Checking/Inference). *If* $\Psi; \Theta; \Delta \vdash \tau : K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$ *then* $\Psi; \Theta; \Delta \vdash \tau : K$*.*

Theorem 8.4 (Soundness of Kind Checking). *If* $\Psi; \Theta; \Delta \vdash_p \tau : K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$ *then* $\Psi; \Theta; \Delta \vdash_p \tau : K$*.*

Proof. Immediate by Theorem A.19 and Theorem 8.1 □

Theorem A.20 (Raw Soundness of Subtyping for Normal Forms). *If* $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\mathit{nf}} \tau_2 : K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$ *then* $\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : K$

Theorem 8.5 (Soundness of Subtyping for Normal Forms). *If* $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{\mathit{nf}} \tau_2 : K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$ *then* $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K$

Proof. Immediate by Theorem A.20 and Theorem 8.1 □

Theorem 8.6 (Soundness of Subtyping). *If* $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash \Phi$*, then* $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K$

Proof. There is only one case: $\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : K \Rightarrow \Phi$ by way of $\Psi; \Theta; \Delta \vdash \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : K \Rightarrow \Phi$ with $\Theta; \Delta \vDash \Phi$. By Theorem 8.5, $\Psi; \Theta; \Delta \vdash \texttt{eval}(\tau_1) <: \texttt{eval}(\tau_2) : K$. By Theorem 7.3 and two uses of S-Trans, $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K$, as required. □

Theorem A.21 (Output Context is Uniquely Determined). *For any* $\Psi, \Theta, \Delta, \Gamma, e, \Phi$*, there is at most one* Γ' *so that* $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \updownarrow \tau \Rightarrow \Phi, \Gamma'$

Theorem 8.7.

- *If* $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'$ *then* $\Gamma' \subseteq \Gamma$
- *If* $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \tau \Rightarrow \Phi, \Gamma'$ *then* $\Gamma' \subseteq \Gamma$

Proof. Induction on $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \updownarrow \tau \Rightarrow \Phi, \Gamma'$. □

Theorem A.22 (Output Context is Well-Formed). *If* $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e \updownarrow \tau \Rightarrow \Phi, \Gamma'$ *then* $\Psi; \Theta; \Delta \vdash \Gamma'$ *wf* $\Rightarrow \Phi'$ *with* $\Theta; \Delta \vDash \Phi'$.

Proof. An easy induction on Γ', using the fact that $\Psi; \Theta; \Delta \vdash \Gamma\ \texttt{wf} \Rightarrow \Phi''$ with $\Theta; \Delta \vDash \Phi''$, and Theorem 8.7 □

Theorem A.23 (Algorithmic Well-Formedness of Context Operations). *If* $\Gamma_1, \Gamma_2 \subseteq \Gamma$ *such that* $\Psi; \Theta; \Delta \vdash \Gamma$ *wf, then the following are true:*

(1) $\Psi; \Theta; \Delta \vdash \Gamma_1, \Gamma_2$ *wf*
(2) $\Psi; \Theta; \Delta \vdash \Gamma_1 \cap \Gamma_2$ *wf*
(3) $\Psi; \Theta; \Delta \vdash \Gamma_1 \setminus \Gamma_2$ *wf*

Theorem A.24 (Raw Soundness of Type Checking/Inference).

(1) If $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'$ *and* $\Theta; \Delta \vDash \Phi$ *then* $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : \tau$
(2) If $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \tau \Rightarrow \Phi, \Gamma'$ *and* $\Theta; \Delta \vDash \Phi$ *then* $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : \tau$

Theorem 8.9 (Soundness of Type Checking/Inference).

(1) If $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e \downarrow \tau \Rightarrow \Phi, \Gamma'$ *and* $\Theta;\Delta \vDash \Phi$ *then* $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma' \vdash_p |e| : \tau$

(2) If $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e \uparrow \tau \Rightarrow \Phi, \Gamma'$ *and* $\Theta;\Delta \vDash \Phi$ *then* $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma' \vdash_p |e| : \tau$

Proof. Immediate by A.24. □

Theorem A.25 (Raw Completeness of Sort Checking/Inference). *If* $\Theta;\Delta \vdash I : S$, *then* $\Theta;\Delta \vdash I : S \Rightarrow \Phi$ *and* $\Theta;\Delta \vDash \Phi$.

Theorem A.26 (Raw Completeness of Constraint Well-Formedness). *If* $\Theta;\Delta \vdash \Phi$ *wf, then* $\Theta;\Delta \vdash \Phi$ *wf* $\Rightarrow \Phi'$ *with* $\Theta;\Delta \vDash \Phi'$

Theorem 8.11. *If* $\Theta \vdash \Delta$ *wf then* $\Theta \vdash \Delta$ *wf* $\Rightarrow \Phi$ *with* $\Theta; \cdot \vDash \Phi$

Proof. By induction on $\Theta \vdash \Delta$ `wf`. The base case is immediate. Suppose $\Theta \vdash \Delta, \Phi$ `wf` by way of $\Theta \vdash \Delta$ `wf` and $\Theta;\Delta \vdash \Phi$ `wf`. By IH, there is some Φ_1 such that $\Theta \vdash \Delta$ `wf` $\Rightarrow \Phi_1$ with $\Theta;\cdot \vDash \Phi_1$. By Theorem A.26, there is Φ_2 such that $\Theta;\Delta \vdash \Phi$ `wf` $\Rightarrow \Phi_2$ with $\Theta;\Delta \vDash \Phi_2$. Equivalently, $\Theta;\cdot \vDash \bigwedge\Delta \to \Phi_2$, and so $\Theta;\cdot \vDash \Phi_1 \wedge (\bigwedge\Delta \to \Phi_2)$. Then, by AWF-CCtx-Ne, $\Theta \vdash \Delta, \Phi$ `wf` $\Rightarrow \Phi_1 \wedge (\bigwedge\Delta \to \Phi_2)$, as required. □

Theorem 8.10. *If* $\Theta;\Delta \vdash_p I : S$, *then* $\Theta;\Delta \vdash_p I : S \Rightarrow \Phi$ *and* $\Theta;\Delta \vDash \Phi$.

Proof. Immediate by Theorem A.25 and Theorem 8.11 □

Theorem 8.12. *If* $\Theta;\Delta \vdash_p \Phi$ *wf, then* $\Theta;\Delta \vdash_p \Phi$ *wf* $\Rightarrow \Phi'$ *with* $\Theta;\Delta \vDash \Phi'$

Proof. Immediate by Theorem A.26 and Theorem 8.11 □

Theorem A.27 (Raw Completeness of Kind Checking/Inference). *If* $\Phi;\Theta;\Delta \vdash \tau : K$, *then* $\Phi;\Theta;\Delta \vdash \tau : K \Rightarrow \Phi$ *with* $\Theta;\Delta \vDash \Phi$

Theorem 8.13. *If* $\Psi;\Theta;\Delta \vdash_p \tau : K$, *then* $\Psi;\Theta;\Delta \vdash_p \tau : K \Rightarrow \Phi$ *with* $\Theta;\Delta \vDash \Phi$

Proof. Immediate by Theorem A.27 and Theorem 8.11. □

Theorem A.28 (Assumption Precomposition for Sort Assignment). *If* $\Theta;\Delta,\Phi_1 \vdash_p I : S \Rightarrow \Phi$ *and* $\Theta;\Delta \vDash (\Phi_2 \to \Phi_1) \wedge \Phi$ *then there exists* Φ' *such that* $\Psi;\Theta;\Delta,\Phi_2 \vdash_p I : S \Rightarrow \Phi'$ *and* $\Theta;\Delta \vDash \Phi'$.

Theorem A.29 (Assumption Precomposition for Constraint Well-Formedness). *If* $\Theta; \Delta, \Phi_1 \vdash_p \Phi''$ *wf* $\Rightarrow \Phi$ *and* $\Theta; \Delta \vDash (\Phi_2 \to \Phi_1) \wedge \Phi$ *then there exists* Φ' *such that* $\Psi; \Theta; \Delta, \Phi_2 \vdash_p \Phi''$ *wf* $\Rightarrow \Phi'$ *and* $\Theta; \Delta \vDash \Phi'$.

Theorem A.30 (Assumption Precomposition for Kind Assignment). *If* $\Psi; \Theta; \Delta, \Phi_1 \vdash_p \tau : K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash (\Phi_2 \to \Phi_1) \wedge \Phi$ *then there exists* Φ' *such that* $\Psi; \Theta; \Delta, \Phi_2 \vdash_p \tau : K \Rightarrow \Phi'$ *and* $\Theta; \Delta \vDash \Phi'$.

Theorem A.31 (Assumption Precomposition for Subtyping). *If* $\Psi; \Theta; \Delta, \Phi_1 \vdash_p \tau <:_{nf} \tau' : K \Rightarrow \Phi$ *and* $\Theta; \Delta \vDash (\Phi_2 \to \Phi_1) \wedge \Phi$ *then there exists* Φ' *such that* $\Psi; \Theta; \Delta, \Phi_2 \vdash_p \tau <:_{nf} \tau' : K \Rightarrow \Phi'$ *and* $\Theta; \Delta \vDash \Phi'$.

Theorem 8.14 (Reflexivity of Algorithmic Subtyping for Neutral Forms). *If* $\Psi; \Theta; \Delta \vdash_p \tau : K$ *and* τ ***ne****, then* $\Psi; \Theta; \Delta \vdash_p \tau <:_{nf} \tau : K \Rightarrow \Phi$ *with* $\Theta; \Delta \vDash \Phi$

Theorem 8.15 (Reflexivity of Algorithmic Subtyping for Normal Forms). *If* $\Psi; \Theta; \Delta \vdash_p \tau : K$ *and* τ ***nf****, then* $\Psi; \Theta; \Delta \vdash_p \tau <:_{nf} \tau : K \Rightarrow \Phi$ *with* $\Theta; \Delta \vDash \Phi$

Theorem 8.17 (Transitivity of Algorithmic Subtyping for Normal Forms). *If* $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{nf} \tau_2 : K \Rightarrow \Phi_1$ *and* $\Psi; \Theta; \Delta \vdash_p \tau_2 <:_{nf} \tau_3 : K \Rightarrow \Phi_2$ *with* $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$*, then* $\Psi; \Theta; \Delta \vdash \tau_1 <:_{nf} \tau_3 : K \Rightarrow \Phi$ *such that* $\Theta; \Delta \vDash \Phi$.

Theorem A.32 (Index Substitution for Algorithmic Sort Checking). *If* $\Theta, i : S; \Delta \vdash_p J : S' \Rightarrow \Phi_1$ *and* $\Theta; \Delta \vdash_p I : S \Rightarrow \Phi_2$ *with* $\Theta, i : S; \Delta \vDash \Phi_1$ *and* $\Theta; \Delta \vDash \Phi_2$*, then* $\Theta; \Delta \vdash_p J[I/i] : S' \Rightarrow \Phi$ *for some* $\Theta; \Delta \vDash \Phi$

Proof. By Theorem 8.2, $\Theta, i : S; \Delta \vdash_p J : S'$ and $\Theta; \Delta \vdash_p I : S$. By Theorem A.7, $\Theta; \Delta \vdash_p J[I/i] : S'$. By Theorem 8.10, $\Theta; \Delta \vdash_p J[I/i] : S' \Rightarrow \Phi'$ for some Φ' such that $\Theta; \Delta \vDash \Phi'$. $\square$

Theorem A.33 (Index Substitution for Algorithmic Constraint Well-Formedness). *If* $\Theta, i : S; \Delta \vdash_p \Phi$ *wf* $\Rightarrow \Phi_1$ *and* $\Theta; \Delta \vdash_p I : S \Rightarrow \Phi_2$ *with* $\Theta, i : S; \Delta \vDash \Phi_1$ *and* $\Theta; \Delta \vDash \Phi_2$*, then* $\Theta; \Delta \vdash_p \Phi[I/i]$ *wf* $\Rightarrow \Phi'$ *for some* $\Theta; \Delta \vDash \Phi'$

Proof. By Theorem 8.3, $\Theta, i : S; \Delta \vdash_p \Phi$ `wf`. By Theorem 8.2, $\Theta; \Delta \vdash_p I : S$. By Theorem A.9, $\Theta; \Delta \vdash_p \Phi[I/i]$ `wf`. By Theorem 8.12, $\Theta; \Delta \vdash_p \Phi[I/i]$ `wf` $\Rightarrow \Phi'$ for some $\Theta; \Delta \vDash \Phi'$ $\square$

THEOREM A.34 (Index Substitution for Algorithmic Type Formation). *If* $\Psi;\Theta,i:S;\Delta \vdash_p \tau : K \Rightarrow \Phi_1$ *and* $\Theta;\Delta \vdash_p I : S \Rightarrow \Phi_2$ *with* $\Theta,i:S;\Delta \vDash \Phi_1$ *and* $\Theta;\Delta \vDash \Phi_2$*, then* $\Psi;\Theta;\Delta \vdash_p \tau[I/i] : K \Rightarrow \Phi$ *for some* $\Theta;\Delta \vDash \Phi$

PROOF. By Theorem 8.4, $\Psi;\Theta,i:S;\Delta \vdash_p \tau : K$. By Theorem 8.2, $\Theta;\Delta \vdash_p I : S$. By Theorem 5.4, $\Psi;\Theta;\Delta \vdash_p \tau[I/i] : K$. Finally, by Theorem 8.13, $\Psi;\Theta;\Delta \vdash_p \tau[I/i] : K \Rightarrow \Phi$ for some $\Theta;\Delta \vDash \Phi$ □

THEOREM 8.19 (Admissibility of Normal Form Subtyping Substitution). *Suppose the following:*

- $\Psi;\Theta,i:S;\Delta \vdash_p \tau_1 <:_{nf} \tau_2 : K \Rightarrow \Phi$ *with* $\Theta;\Delta \vDash \Phi$ *and* $\Theta \vdash \Delta$ *wf.*
- $\Theta;\Delta \vdash_p I : S \Rightarrow \Phi_1$ *with* $\Theta;\Delta \vDash \Phi_1$
- $\Theta;\Delta \vdash_p J : S \Rightarrow \Phi_2$ *with* $\Theta;\Delta \vDash \Phi_2$
- $\Theta;\Delta \vDash I = J$

Then, $\Psi;\Theta;\Delta \vdash_p \tau_1[I/i] <:_{nf} \tau_2[J/i] : K \Rightarrow \Phi'$ *for some* Φ' *with* $\Theta;\Delta \vDash \Phi'$*.*

THEOREM 8.20 (Type Family Application Commutes with Evaluation). *If* $\Psi;\Theta;\Delta \vdash_p \mathit{eval}(\tau_1) <:_{nf} \mathit{eval}(\tau_2) : S \to K \Rightarrow \Phi$ *and* $\Theta;\Delta \vDash \Phi \wedge I = J$ *with* $\Theta;\Delta \vdash_p I : S$ *and* $\Theta;\Delta \vdash_p J : S$ *then* $\Psi;\Theta;\Delta \vdash_p \mathit{eval}(\tau_1\ I) <:_{nf} \mathit{eval}(\tau_2\ J) : K \Rightarrow \Phi'$ *for some* $\Theta;\Delta \vDash \Phi'$*.*

PROOF. By inversion on $\Psi;\Theta;\Delta \vdash \mathtt{eval}(\tau_1) <:_{\mathtt{nf}} \mathtt{eval}(\tau_2) : S \to K \Rightarrow \Phi$.

- For the first case, suppose the derivation was $\Psi;\Theta;\Delta \vdash \lambda i : S.\tau_1' <:_{\mathtt{nf}} \tau_2' : S \to K \Rightarrow \Phi$ from $\Psi;\Theta,i:S;\Delta \vdash \tau_1' <:_{\mathtt{nf}} \tau_2' : K \Rightarrow \Phi'$. By Theorem 8.19, $\Psi;\Theta;\Delta \vdash_p \tau_1'[I/i] <:_{\mathtt{nf}} \tau_2'[J/i] : K \Rightarrow \Phi'$, for some $\Theta;\Delta \vDash \Phi'$. But $\mathtt{eval}(\tau_1\ I) = \tau_1'[I/i]$ and $\mathtt{eval}(\tau_2\ J) = \tau_2'[J/i]$.
- Now, suppose the derivation was $\Psi;\Theta;\Delta \vdash \tau_1'\ L_1 <:_{\mathtt{nf}} \tau_2'\ L_2 : S \to K \Rightarrow \Phi \wedge (L_1 = L_2)$, where $\mathtt{eval}(\tau_1) = \tau_1'\ L_1$ and $\mathtt{eval}(\tau_2) = \tau_2\ L_2$. These must both be $\mathtt{ne}$, since they are both applications, and therefore $\mathtt{eval}(\tau_1)\ I = \mathtt{eval}(\tau_1\ I)$ and $\mathtt{eval}(\tau_2)\ J = \mathtt{eval}(\tau_2\ J)$, as required.

□

THEOREM 8.21 (Completeness of Algorithmic Subtyping). *If* $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_2 : K$ *then* $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_2 : K \Rightarrow \Phi$ *and* $\Theta;\Delta \vDash \Phi$.

THEOREM 8.22 (Admissibility of Algorithmic Weakening).

(1) If $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e \downarrow \tau \Rightarrow \Phi, \Gamma''$ with $\Theta;\Delta \vDash \Phi$, then whenever $\Psi;\Theta;\Delta \vdash_p \Gamma' \sqsubseteq \Gamma$ and $\Psi;\Theta;\Delta \vdash_p \Omega' \sqsubseteq \Omega$, there are Φ_1, e_1, Γ_1 so that $|e_1| = |e|$, $\Theta;\Delta \vDash \Phi_1$, $\Psi;\Theta;\Delta \vdash_p \Gamma_1 \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash_p \Gamma_1 \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash_p e_1 \downarrow \tau \Rightarrow \Phi_1, \Gamma_1$.

(2) If $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e \uparrow \tau \Rightarrow \Phi, \Gamma''$ with $\Theta;\Delta \vDash \Phi$, then whenever $\Psi;\Theta;\Delta \vdash_p \Gamma' \sqsubseteq \Gamma$ and $\Psi;\Theta;\Delta \vdash_p \Omega' \sqsubseteq \Omega$, there are Φ_2, e_2, Γ_2 so that $|e_2| = |e|$, $\Theta;\Delta \vDash \Phi_2$, $\Psi;\Theta;\Delta \vdash_p \Gamma_2 \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash_p \Gamma_2 \sqsubseteq \Gamma''$ and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash_p e_2 \uparrow \tau \Rightarrow \Phi_2, \Gamma_2$.

THEOREM 8.23 (Completeness of Type Checking/Inference). *If $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e : \tau$, then:*

(1) There are e', Φ', Γ' such that $|e'| = e$, $\Theta;\Delta \vDash \Phi'$, and $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e' \downarrow \tau \Rightarrow \Phi', \Gamma'$.

(2) There are e'', Φ'', Γ'' such that $|e''| = e$, $\Theta;\Delta \vDash \Phi''$, and $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e'' \uparrow \tau \Rightarrow \Phi'', \Gamma''$

5. Proofs

Proof of Theorem A.6

By induction on the derivation of $\Theta, j : S_1; \Delta \vdash I : S_2$. The case for I-Var is immediate.

▸ **Case 1:** *I-Plus.*

▸ **Given:**

(1) $\Theta, j : S_1; \Delta \vdash I_1 + I_2 : bS$

(2) $\Theta, j : S_1; \Delta \vdash I_1 : bS$

(3) $\Theta, j : S_1; \Delta \vdash I_2 : bS$

(4) $\Theta \vdash \Delta\ \mathtt{wf}$

▸ **Goal:**

$$\boxed{\Theta; \Delta[J/j] \vdash (I + J)[J/j] : bS}$$

By IH on (2) and (3)

(5) $\Theta; \Delta[J/j] \vdash I_1[J/j] : bS$

(6) $\Theta; \Delta[J/j] \vdash I_2[J/j] : bS$

By I-Plus

(7) $\Theta; \Delta[J/j] \vdash I_1[J/j] + I_2[J/j] : bS$

Goal follows by (7).

▸ **Case 2:** *I-Minus.*

- **Given:**

  (1) $\Theta, j : S_1; \Delta \vdash I_1 - I_2 : bS$

  (2) $\Theta, j : S_1; \Delta \vdash I_1 : bS$

  (3) $\Theta, j : S_1; \Delta \vdash I_2 : bS$

  (4) $\Theta, j : S_1; \Delta \vDash I_1 \geq I_2$

  (5) $\Theta \vdash \Delta$ `wf`

- **Goal:**

  $\boxed{\Theta; \Delta[J/j] \vdash (I + 1 - I_2)[J/j] : bS}$

By IH on (2) and (3)

(5) $\Theta; \Delta[J/j] \vdash I_1[J/j] : bS$

(6) $\Theta; \Delta[J/j] \vdash I_2[J/j] : bS$

Instantiating the quantifier in (4) and using (5)

(7) $\Theta; \Delta[J/j] \vDash I_1[J/j] \geq I_2[J/j]$

By I-Minus on (5) (6) (7)

(8) $\Theta; \Delta[J/j] \vdash I_1[J/j] - I_2[J/j] : bS$

Goal follows by (8).

- **Case 3:** *I-Times-*$\mathbb{R}^+$.

- **Given:**

  (1) $\Theta, j : S_1; \Delta \vdash c \cdot I : \mathbb{R}^+$

  (2) $\Theta, j : S_1; \Delta \vdash I : \mathbb{R}^+$

  (3) $c \in \mathbb{R}^+$

- **Goal:**

  $\boxed{\Theta; \Delta[I/i] \vdash (c \cdot I)[J/j] : \mathbb{R}^+}$

By IH on (2)

(4) $\Theta; \Delta[J/j] \vdash I[J/j] : \mathbb{R}^+$

By I-Times-$\mathbb{R}^+$ on (3) and (4)

(5) $\Theta; \Delta[J/j] \vdash c \cdot I[J/j] : \mathbb{R}^+$

Goal follows by (5)

- **Case 4:** *I-Times-*$\vec{\mathbb{R}}^+$.

Identical to I-Times-$\mathbb{R}^+$

- **Case 5:** *I-Times-*$\mathbb{N}$*.*

  Identical to I-Times-$\mathbb{R}^+$

- **Case 6:** *I-Shift.*

  - **Given:**

    (1) $\Theta, j : S; \Delta \vdash_{\triangleleft} I : \vec{\mathbb{R}^+}$

    (2) $\Theta, j : S; \Delta \vdash I : \vec{\mathbb{R}^+}$

  - **Goal:**

    $\boxed{\Theta; \Delta[J/j] \vdash (\triangleleft\, I)\,[J/j] : \vec{\mathbb{R}^+}}$

  By IH on (2)

  (3) $\Theta; \Delta[J/j] \vdash I[J/j] : \vec{\mathbb{R}^+}$

  By I-Shift on (3)

  (4) $\Theta; \Delta[J/j] \vdash_{\triangleleft} I[J/j] : \vec{\mathbb{R}^+}$

  Goal follows immediately from (4)

- **Case 7:** *I-Lam.*

  - **Given:**

    (1) $\Theta, j : S_1; \Delta \vdash \lambda i : S_2.I : S_2 \to S_3$

    (2) $\Theta, j : S_1, i : S_2; \Delta \vdash I : S_3$

  - **Goal:**

    $\boxed{\Theta; \Delta[J/j] \vdash \lambda i : S_2.I[J/j] : S_2 \to S_3}$

  By IH on (2)

  (3) $\Theta, i : S_2; \Delta[J/j] \vdash_p I[J/j] : S_3$

  By I-Lam on (3)

  (4) $\Theta; \Delta[J/j] \vdash \lambda i : S_2.I[J/j] : S_2 \to S_3$

  Goal follows immediately by (4)

- **Case 8:** *I-App.*

  - **Given:**

    (1) $\Theta, j : S; \Delta \vdash I_1\ I_2 : S_2$

(2) $\Theta, j : S; \Delta \vdash I_1 : S_1 \to S_2$

(3) $\Theta, j : S; \Delta \vdash I_2 : S_1$

▸ **Goal:**

$\boxed{\Theta; \Delta[J/j] \vdash (I_1\ I_2)[J/j] : S_2,}$

By IH on (2) and (3)

(4) $\Theta; \Delta[J/j] \vdash I_1[J/j] : S_1 \to S_2$

(5) $\Theta; \Delta[J/j] \vdash I_2[J/j] : S_1$

By I-App on (4) and (5)

(6) $\Theta; \Delta[J/j] \vdash I_1[J/j]\ I_2[J/j] : S_2$

Goal follows from (6)

▸ **Case 9:** *I-Const.*

▸ **Given:**

(1) $\Theta, j : S; \Delta \vdash \mathtt{const}(I) : \vec{\mathbb{R}}^{+}$

(2) $\Theta, j : S; \Delta \vdash I : \mathbb{R}^{+}$

▸ **Goal:**

$\boxed{\Theta; \Delta[J/j] \vdash \mathtt{const}(I)[J/j] : \vec{\mathbb{R}}^{+}}$

By IH on (2)

(3) $\Theta; \Delta[J/j] \vdash I[J/j] : \mathbb{R}^{+}$

By I-Const on (3)

(4) $\Theta; \Delta[J/j] \vdash \mathtt{const}(I[J/j]) : \vec{\mathbb{R}}^{+}$

Goal follows from (4)

▸ **Case 10:** *I-$\mathbb{N}$-Lit.*

Immediate.

▸ **Case 11:** *I-$\mathbb{R}^{+}$-Lit.*

Immediate.

▸ **Case 12:** *I-$\vec{\mathbb{R}}^{+}$-Lit.*

Immediate.

□

Proof of Theorem A.8

▸ **Given:**

(1) $\Theta, i : S; \Delta \vdash \Phi\,\texttt{wf}$

(2) $\Theta; \Delta \vdash I : S$

▸ **Goal:**

$\boxed{\Theta; \Delta[I/i] \vdash_p \Phi[I/i]\,\texttt{wf}}$

▸ **Case 1:** *C-Top.*

Immediate.

▸ **Case 2:** *C-Bot.*

Immediate.

▸ **Case 3:** *C-Conj.*

▸ **Given:**

(1) $\Theta, i : S; \Delta \vdash \Phi_1 \wedge \Phi_2\,\texttt{wf}$

(3) $\Theta, i : S; \Delta \vdash \Phi_1\,\texttt{wf}$

(4) $\Theta, i : S; \Delta \vdash \Phi_2\,\texttt{wf}$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash_p (\Phi_1 \wedge \Phi_2)[I/i]\,\texttt{wf}}$

By IH on (3) and (4)

(5) $\Theta; \Delta[I/i] \vdash \Phi_1[I/i]\,\texttt{wf}$

(6) $\Theta; \Delta[I/i] \vdash \Phi_2[I/i]\,\texttt{wf}$

The result follows by C-Conj on (5) and (6)

▸ **Case 4:** *C-Disj.*

▸ **Given:**

(1) $\Theta, i : S; \Delta \vdash \Phi_1 \vee \Phi_2\,\texttt{wf}$

(3) $\Theta, i : S; \Delta \vdash \Phi_1\,\texttt{wf}$

(4) $\Theta, i : S; \Delta \vdash \Phi_2\,\texttt{wf}$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash_p (\Phi_1 \vee \Phi_2)[I/i]\,\texttt{wf}}$

By IH on (3) and (4)

(5) $\Theta; \Delta[I/i] \vdash \Phi_1[I/i]\,\texttt{wf}$

(6) $\Theta; \Delta[I/i] \vdash \Phi_2[I/i]\,\texttt{wf}$

The result follows by C-Disj on (5) and (6)

- **Case 5:** *C-Impl.*

  - **Given:**

    (1) $\Theta, i : S; \Delta \vdash \Phi_1 \to \Phi_2\,\texttt{wf}$

    (3) $\Theta, i : S; \Delta \vdash \Phi_1\,\texttt{wf}$

    (4) $\Theta, i : S; \Delta, \Phi_1 \vdash \Phi_2\,\texttt{wf}$

  - **Goal:**

    $\boxed{\Theta; \Delta[I/i] \vdash (\Phi_1 \vee \Phi_2)[I/i]\,\texttt{wf}}$

  By IH on (3) and (4)

  (5) $\Theta; \Delta[I/i] \vdash \Phi_1[I/i]\,\texttt{wf}$

  (6) $\Theta; \Delta[I/i], \Phi_1[I/i] \vdash \Phi_2[I/i]\,\texttt{wf}$

  By C-Impl on (5) and (6)

  (7) $\Theta; \Delta[I/i] \vdash \Phi_1[I/i] \to \Phi_2[I/i]\,\texttt{wf}$

  The result follows by C-Disj on (5) and (6)

- **Case 6:** *C-Forall.*

  - **Given:**

    (1) $\Theta, i : S; \Delta \vdash \forall j : S'.\Phi\,\texttt{wf}$

    (3) $\Theta, i : S, j : S'; \Delta \vdash \Phi\,\texttt{wf}$

  - **Goal:**

    $\boxed{\Theta; \Delta[I/i] \vdash (\forall j : S'.\Phi)[I/i]}$

  By IH on (3)

  (4) $\Theta, j : S'; \Delta[I/i] \vdash \Phi[I/i]\,\texttt{wf}$

  By C-Forall on (4)

  (5) $\Theta; \Delta[I/i] \vdash \forall j : S'.\Phi[I/i]\,\texttt{wf}$

  The goal follows by (5)

- **Case 6:** *C-Exists.*

  - **Given:**

    (1) $\Theta, i : S; \Delta \vdash \exists j : S'.\Phi\,\texttt{wf}$

(3) $\Theta, i : S, j : S'; \Delta \vdash \Phi\,\texttt{wf}$

- **Goal:**

$\boxed{\Theta; \Delta[I/i] \vdash (\exists j : S'.\Phi)[I/i]}$

By IH on (3)

(4) $\Theta, j : S'; \Delta[I/i] \vdash \Phi[I/i]\,\texttt{wf}$

By C-Exists on (4)

(5) $\Theta; \Delta[I/i] \vdash \exists j : S'.\Phi[I/i]\,\texttt{wf}$

The goal follows by (5)

- **Case 7:** *C-Eq.*

Immediate by Theorem A.6.

- **Case 8:** *C-Leq.*

Immediate by Theorem A.6.

- **Case 9:** *C-Lt.*

Immediate by Theorem A.6.

□

Proof of Theorem 5.4:

By induction on $\Psi; \Theta, i : S; \Delta \vdash \tau : K$

- **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p \tau : K$

(2) $\Theta; \Delta \vdash_p I : S$

- **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \tau[I/i] : K}$

- **Case 1:** *K-Var.*

Immediate.

- **Case 2:** *K-Unit.*

Immediate.

- **Case 3:** *K-Arr.*

- **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 \multimap \tau_2 : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 : \star$

(4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_2 : \star$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] \multimap \tau_2[I/i] : \star}$$

By IH on (3) and (4)

(5) $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] : \star$

(6) $\Psi; \Theta; \Delta \vdash_p \tau_2[I/i] : \star$

Goal follows by K-Arr on (5) and (6)

▸ **Case 4:** *K-Tensor.*

▸ **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 \otimes \tau_2 : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 : \star$

(4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_2 : \star$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] \otimes \tau_2[I/i] : \star}$$

By IH on (3) and (4)

(5) $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] : \star$

(6) $\Psi; \Theta; \Delta \vdash_p \tau_2[I/i] : \star$

Goal follows by K-Tensor on (5) and (6)

▸ **Case 5:** *K-With.*

▸ **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 \& \tau_2 : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 : \star$

(4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_2 : \star$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] \& \tau_2[I/i] : \star}$$

By IH on (3) and (4)

(5) $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] : \star$

(6) $\Psi; \Theta; \Delta \vdash_p \tau_2[I/i] : \star$

Goal follows by K-With on (5) and (6)

▸ **Case 6:** *K-Sum.*

▸ **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 \oplus \tau_2 : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 : \star$

(4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_2 : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] \oplus \tau_2[I/i] : \star}$

By IH on (3) and (4)

(5) $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] : \star$

(6) $\Psi; \Theta; \Delta \vdash_p \tau_2[I/i] : \star$

Goal follows by K-Sum on (5) and (6)

▸ **Case 7:** *K-Bang.*

▸ **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p !\tau : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p !\tau[I/i] : \star}$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash_p \tau[I/i] : \star$

Goal follows by K-Bang on (5)

▸ **Case 8:** *K-IForall.*

▸ **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p \forall j : S'.\tau : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S, j : S'; \Delta \vdash_p \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \forall j : S'.\tau[I/i] : \star}$

By IH on (3)

(4) $\Psi; \Theta, j : S'; \Delta \vdash_p \tau[I/i] : \star$

Goal follows by K-IForall on (4)

▸ **Case 9:** *K-IExists.*

▸ **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p \exists j : S'.\tau : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S, j : S'; \Delta \vdash_p \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \exists j : S'.\tau[I/i] : \star}$

By IH on (3)

(4) $\Psi; \Theta, j : S'; \Delta \vdash_p \tau[I/i] : \star$

Goal follows by K-IExists on (4)

▸ **Case 10:** *K-TForall.*

▸ **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p \forall \alpha : K.\tau : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi, \alpha : K; \Theta, i : S; \Delta \vdash_p \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \forall \alpha : K.\tau[I/i] : \star}$

By IH on (3)

(4) $\Psi, \alpha : K; \Theta; \Delta \vdash_p \tau[I/i] : \star$

Goal follows by K-TForall on (4)

▸ **Case 11:** *K-List.*

- **Given:**

  (1) $\Psi; \Theta, i : S; \Delta \vdash_p L^J \tau : \star$

  (2) $\Theta; \Delta \vdash_p I : S$

  (3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau : \star$

  (4) $\Theta, i : S; \Delta \vdash_p J : \mathbb{N}$

- **Goal:**

  $\boxed{\Psi; \Theta; \Delta \vdash_p L^{J[I/i]} (\tau[I/i]) : \star}$

  By IH on (3)

  (5) $\Psi; \Theta; \Delta \vdash_p \tau[I/i] : \star$

  By Theorem A.7 on (4)

  (6) $\Theta; \Delta \vdash_p J[I/i] : \mathbb{N}$

  Goal follows by K-List on (5) and (6)

- **Case 12:** *K-Impl.*

  - **Given:**

    (1) $\Psi; \Theta, i : S; \Delta \vdash_p \Phi \implies \tau : \star$

    (2) $\Theta; \Delta \vdash_p I : S$

    (3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau : \star$

    (4) $\Theta, i : S; \Delta \vdash_p \Phi\ \mathtt{wf}$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \Phi[I/i] \implies \tau[I/i] : \star}$

    By IH on (3)

    (5) $\Psi; \Theta; \Delta \vdash_p \tau[I/i] : \star$

    By Theorem A.9 on (4)

    (6) $\Theta; \Delta \vdash_p \Phi[I/i]\ \mathtt{wf}$

    Goal follows by K-Impl on (5) and (6)

- **Case 13:** *K-Conj.*

  - **Given:**

    (1) $\Psi; \Theta, i : S; \Delta \vdash_p \Phi \& \tau : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau : \star$

(4) $\Theta, i : S; \Delta \vdash_p \Phi \ \texttt{wf}$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash_p \Phi[I/i] \& \tau[I/i] : \star}$$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash_p \tau[I/i] : \star$

By Theorem A.9 on (4)

(6) $\Theta; \Delta \vdash_p \Phi[I/i] \ \texttt{wf}$

Goal follows by K-Conj on (5) and (6)

▸ **Case 14:** *K-Monad.*

▸ **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p \mathbb{M}\,(J, \vec{p})\,\tau : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau : \star$

(4) $\Theta, i : S; \Delta \vdash_p J : \mathbb{N}$

(5) $\Theta, i : S; \Delta \vdash_p \vec{p} : \vec{\mathbb{R}^+}$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash_p \mathbb{M}\,(J[I/i], \vec{p}[I/i])\ (\tau[I/i]) : \star}$$

By IH on (3)

(6) $\Psi; \Theta; \Delta \vdash_p \tau[I/i] : \star$

By Theorem A.7 on (4) and (5)

(7) $\Theta; \Delta \vdash_p J[I/i] : \mathbb{N}$

(8) $\Theta; \Delta \vdash_p \vec{p}[I/i] : \vec{\mathbb{R}^+}$

Goal follows by K-Monad on (6), (7), and (8)

▸ **Case 15:** *K-Pot.*

▸ **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p [J|\vec{p}]\,\tau : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau : \star$

(4) $\Theta, i : S; \Delta \vdash_p J : \mathbb{N}$

(5) $\Theta, i : S; \Delta \vdash_p \vec{p} : \vec{\mathbb{R}^+}$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash_p [J[I/i]|\vec{p}[I/i]]\ (\tau[I/i]) : \star}$$

By IH on (3)

(6) $\Psi; \Theta; \Delta \vdash_p \tau[I/i] : \star$

By Theorem A.7 on (4) and (5)

(7) $\Theta; \Delta \vdash_p J[I/i] : \mathbb{N}$

(8) $\Theta; \Delta \vdash_p \vec{p}[I/i] : \vec{\mathbb{R}^+}$

Goal follows by K-Pot on (6), (7), and (8)

▸ **Case 16:** *K-ConstPot.*

▸ **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p [J]\, \tau : \star$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau : \star$

(4) $\Theta, i : S; \Delta \vdash_p J : \mathbb{R}$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash_p [J[I/i]]\ (\tau[I/i]) : \star}$$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash_p \tau[I/i] : \star$

By Theorem A.7 on (4)

(6) $\Theta; \Delta \vdash_p J[I/i] : \mathbb{N}$

Goal follows by K-ConstPot on (5) and (6)

▸ **Case 17:** *K-FamLam.*

▸ **Given:**

(1) $\Psi; \Theta, i : S; \Delta \vdash_p \lambda j : S'.\tau : S' \to K$

(2) $\Theta; \Delta \vdash_p I : S$

(3) $\Psi; \Theta, i : S, j : S' \vdash_p \tau : K$

- **Goal:**

  $\boxed{\Psi; \Theta; \Delta \vdash_p \lambda j : S'.\tau[I/i] : S' \to K}$

  By IH on (3)

  (4) $\Psi; \Theta, j : S' \vdash_p \tau[I/i] : K$

  Goal follows by K-FamLam on (4)

- **Case 18:** *K-FamApp.*

  - **Given:**

    (1) $\Psi; \Theta, i : S; \Delta \vdash_p \tau\ J : K$

    (2) $\Theta; \Delta \vdash_p I : S$

    (3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau : S' \to K$

    (4) $\Theta, i : S; \Delta \vdash_p J : S'$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p (\tau[I/i])\ (J[I/i]) : K}$

  By IH on (3)

  (5) $\Psi; \Theta; \Delta \vdash_p \tau[I/i] : S' \to K$

  By Theorem A.7 on (4)

  (6) $\Theta; \Delta \vdash_p J[I/i] : S'$

  Goal follows by K-FamApp on (5) and (6)

□

Proof of Theorem 7.3

- **Given:**

  (1) $\Psi; \Theta; \Delta \vdash_p \tau : K$

- **Goal 1:**

  $\boxed{\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau) : K}$

- **Goal 2:**

  $\boxed{\Psi; \Theta; \Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : K}$

- **Goal 3:**

  $\boxed{\texttt{eval}(\tau)\ \texttt{nf}}$

By induction on the derivation of $\Psi; \Theta; \Delta \vdash \tau : K$

▸ **Case 1:** *K-Var.*

Immediate.

▸ **Case 2:** *K-Unit.*

Immediate.

▸ **Case 3:** *K-Arr.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \tau_1 \multimap \tau_2 : \star$

(2) $\Psi;\Theta;\Delta \vdash_p \tau_1 : \star$

(3) $\Psi;\Theta;\Delta \vdash_p \tau_2 : \star$

▸ **Goal 1:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) \multimap \texttt{eval}(\tau_1) : \star}$

▸ **Goal 2:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \tau_1 \multimap \tau_2 \equiv \texttt{eval}(\tau_1) \multimap \texttt{eval}(\tau_1) : \star}$

▸ **Goal 3:**

$\boxed{\texttt{eval}(\tau_1) \multimap \texttt{eval}(\tau_1)\ \texttt{nf}}$

By IH on (2)

(4) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) : \star$

(5) $\Psi;\Theta;\Delta \vdash_p \tau_1 \equiv \texttt{eval}(\tau_1) : \star$

(6) $\texttt{eval}(\tau_1)\ \texttt{nf}$

By IH on (3)

(7) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_2) : \star$

(8) $\Psi;\Theta;\Delta \vdash_p \tau_2 \equiv \texttt{eval}(\tau_2) : \star$

(9) $\texttt{eval}(\tau_2)\ \texttt{nf}$

Goal 1 follows by K-Arr on (4) and (7)

Goal 2 follows by S-Arr twice on (5) and (8)

Goal 3 follows from (6) and (9)

▸ **Case 4:** *K-Tensor.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \tau_1 \otimes \tau_2 : \star$

(2) $\Psi; \Theta; \Delta \vdash_p \tau_1 : \star$

(3) $\Psi; \Theta; \Delta \vdash_p \tau_2 : \star$

▸ **Goal 1:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau_1) \otimes \texttt{eval}(\tau_1) : \star}$

▸ **Goal 2:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1 \otimes \tau_2 \equiv \texttt{eval}(\tau_1) \otimes \texttt{eval}(\tau_1) : \star}$

▸ **Goal 3:**

$\boxed{\texttt{eval}(\tau_1) \otimes \texttt{eval}(\tau_1)\ \texttt{nf}}$

By IH on (2)

(4) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau_1) : \star$

(5) $\Psi; \Theta; \Delta \vdash_p \tau_1 \equiv \texttt{eval}(\tau_1) : \star$

(6) $\texttt{eval}(\tau_1)\ \texttt{nf}$

By IH on (3)

(7) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau_2) : \star$

(8) $\Psi; \Theta; \Delta \vdash_p \tau_2 \equiv \texttt{eval}(\tau_2) : \star$

(9) $\texttt{eval}(\tau_2)\ \texttt{nf}$

Goal 1 follows by K-Tensor on (4) and (7)

Goal 2 follows by S-Tensor twice on (5) and (8)

Goal 3 follows from (6) and (9)

▸ **Case 5:** *K-With.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \tau_1 \& \tau_2 : \star$

(2) $\Psi; \Theta; \Delta \vdash_p \tau_1 : \star$

(3) $\Psi; \Theta; \Delta \vdash_p \tau_2 : \star$

▸ **Goal 1:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau_1) \& \texttt{eval}(\tau_1) : \star}$

▸ **Goal 2:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1 \& \tau_2 \equiv \texttt{eval}(\tau_1) \& \texttt{eval}(\tau_1) : \star}$

▸ **Goal 3:**

$\boxed{\texttt{eval}(\tau_1) \& \texttt{eval}(\tau_1)\ \texttt{nf}}$

By IH on (2)

(4) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau_1) : \star$

(5) $\Psi; \Theta; \Delta \vdash_p \tau_1 \equiv \texttt{eval}(\tau_1) : \star$

(6) $\texttt{eval}(\tau_1)\ \texttt{nf}$

By IH on (3)

(7) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau_2) : \star$

(8) $\Psi; \Theta; \Delta \vdash_p \tau_2 \equiv \texttt{eval}(\tau_2) : \star$

(9) $\texttt{eval}(\tau_2)\ \texttt{nf}$

Goal 1 follows by K-With on (4) and (7)

Goal 2 follows by S-With twice on (5) and (8)

Goal 3 follows from (6) and (9)

▸ **Case 6:** *K-Sum.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \tau_1 \oplus \tau_2 : \star$

(2) $\Psi; \Theta; \Delta \vdash_p \tau_1 : \star$

(3) $\Psi; \Theta; \Delta \vdash_p \tau_2 : \star$

▸ **Goal 1:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau_1) \oplus \texttt{eval}(\tau_1) : \star}$

▸ **Goal 2:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1 \oplus \tau_2 \equiv \texttt{eval}(\tau_1) \oplus \texttt{eval}(\tau_1) : \star}$

▸ **Goal 3:**

$\boxed{\texttt{eval}(\tau_1) \oplus \texttt{eval}(\tau_1)\ \texttt{nf}}$

By IH on (2)

(4) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau_1) : \star$

(5) $\Psi; \Theta; \Delta \vdash_p \tau_1 \equiv \texttt{eval}(\tau_1) : \star$

(6) $\texttt{eval}(\tau_1)\ \texttt{nf}$

By IH on (3)

(7) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau_2) : \star$

(8) $\Psi; \Theta; \Delta \vdash_p \tau_2 \equiv \texttt{eval}(\tau_2) : \star$

(9) $\texttt{eval}(\tau_2)\ \texttt{nf}$

Goal 1 follows by K-Sum on (4) and (7)

Goal 2 follows by S-Sum twice on (5) and (8)

Goal 3 follows from (6) and (9)

- **Case 7:** *K-Bang.*
  - **Given:**

    (1) $\Psi;\Theta;\Delta \vdash_p !\tau : \star$

    (2) $\Psi;\Theta;\Delta \vdash_p \tau\star$
  - **Goal 1:**

    $\boxed{\Psi;\Theta;\Delta \vdash_p !\texttt{eval}(\tau) : \star}$
  - **Goal 2:**

    $\boxed{\Psi;\Theta;\Delta \vdash_p !\tau \equiv !\texttt{eval}(\tau) : \star}$
  - **Goal 3:**

    $\boxed{!\texttt{eval}(\tau)\ \texttt{nf}}$

  By IH on (2)

  (4) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau) : \star$

  (5) $\Psi;\Theta;\Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : \star$

  (6) $\texttt{eval}(\tau)\ \texttt{nf}$

  Goal 1 follows by K-Bang on (4)

  Goal 2 follows by S-Sum on (5)

  Goal 3 follows from (6)

- **Case 8:** *K-IForall.*
  - **Given:**

    (1) $\Psi;\Theta;\Delta \vdash_p \forall i : S.\tau : \star$

    (2) $\Psi;\Theta, i : S;\Delta \vdash_p \tau : \star$
  - **Goal 1:**

    $\boxed{\Psi;\Theta;\Delta \vdash_p \forall i : S.\texttt{eval}(\tau) : \star}$
  - **Goal 2:**

    $\boxed{\Psi;\Theta;\Delta \vdash_p \forall i : S.\tau \equiv \forall i : S.\texttt{eval}(\tau) : \star}$
  - **Goal 3:**

$\boxed{\forall i : S.\texttt{eval}(\tau)\ \texttt{nf}}$

By IH on (2)

(3) $\Psi; \Theta, i : S; \Delta \vdash_p \texttt{eval}(\tau) : \star$

(4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : \star$

(5) $\texttt{eval}(\tau)\ \texttt{nf}$

Goal 1 follows by K-IForall on (3)

Goal 2 follows by S-IForall on (4)

Goal 3 is immediate from (5)

- **Case 9:** *K-IExists.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \exists i : S.\tau : \star$

    (2) $\Psi; \Theta, i : S; \Delta \vdash_p \tau : \star$

  - **Goal 1:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \exists i : S.\texttt{eval}(\tau) : \star}$

  - **Goal 2:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \exists i : S.\tau \equiv \exists i : S.\texttt{eval}(\tau) : \star}$

  - **Goal 3:**

    $\boxed{\exists i : S.\texttt{eval}(\tau)\ \texttt{nf}}$

  By IH on (2)

  (3) $\Psi; \Theta, i : S; \Delta \vdash_p \texttt{eval}(\tau) : \star$

  (4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : \star$

  (5) $\texttt{eval}(\tau)\ \texttt{nf}$

  Goal 1 follows by K-IExists on (3)

  Goal 2 follows by S-IExists on (4)

  Goal 3 is immediate from (5)

- **Case 10:** *K-TForall.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \forall \alpha : K.\tau : \star$

    (2) $\Psi, \alpha : K; \Theta; \Delta \vdash_p \tau : \star$

  - **Goal 1:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \forall\alpha : K.\texttt{eval}(\tau) : \star}$

▸ **Goal 2:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \forall\alpha : K.\tau \equiv \forall\alpha : K.\texttt{eval}(\tau) : \star}$

▸ **Goal 3:**

$\boxed{\forall\alpha : K.\texttt{eval}(\tau)\ \texttt{nf}}$

By IH on (2)

(3) $\Psi, \alpha : K;\Theta;\Delta \vdash_p \texttt{eval}(\tau) : \star$

(4) $\Psi, \alpha : K;\Theta;\Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : \star$

(5) $\texttt{eval}(\tau)\ \texttt{nf}$

Goal 1 follows by K-TForall on (3)

Goal 2 follows by S-TForall on (4)

Goal 3 is immediate from (5)

▸ **Case 11:** *K-List.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p L^I\tau : \star$

(2) $\Psi;\Theta;\Delta \vdash_p \tau : \star$

(3) $\Theta;\Delta \vdash_p I : \mathbb{N}$

▸ **Goal 1:**

$\boxed{\Psi;\Theta;\Delta \vdash_p L^I\,(\texttt{eval}(\tau)) : \star}$

▸ **Goal 2:**

$\boxed{\Psi;\Theta;\Delta \vdash_p L^I\tau \equiv L^I\,(\texttt{eval}(\tau)) : \star}$

▸ **Goal 3:**

$\boxed{L^I\,(\texttt{eval}(\tau))\ \texttt{nf}}$

By IH on (2)

(4) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau) : \star$

(5) $\Psi;\Theta;\Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : \star$

(6) $\texttt{eval}(\tau)\ \texttt{nf}$

Goal 1 follows by K-List on (4) and (3)

Goal 2 follows by S-List on (5), and the fact that $\Theta;\Delta \vDash I = I$

Goal 3 is immediate from (6)

- **Case 11:** *K-Conj.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \Phi \& \tau : \star$

    (2) $\Psi; \Theta; \Delta \vdash_p \tau : \star$

    (3) $\Theta; \Delta \vdash_p \Phi \, \texttt{wf}$
  - **Goal 1:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \Phi \& \texttt{eval}(\tau) : \star}$
  - **Goal 2:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \Phi \& \tau \equiv \Phi \& \texttt{eval}(\tau) : \star}$
  - **Goal 3:**

    $\boxed{\Phi \& \texttt{eval}(\tau) \, \texttt{nf}}$

  By IH on (2)

  (4) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau) : \star$

  (5) $\Psi; \Theta; \Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : \star$

  (6) $\texttt{eval}(\tau) \, \texttt{nf}$

  Goal 1 follows by K-Conj from (4) and (3)

  Goal 2 follows by S-Conj from (5) and the fact that $\Theta; \Delta \vDash \Phi \to \Phi$

  Goal 3 follows from (6)

- **Case 12:** *K-Impl.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \Phi \implies \tau : \star$

    (2) $\Psi; \Theta; \Delta \vdash_p \tau : \star$

    (3) $\Theta; \Delta \vdash_p \Phi \, \texttt{wf}$
  - **Goal 1:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \Phi \implies \texttt{eval}(\tau) : \star}$
  - **Goal 2:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \Phi \implies \tau \equiv \Phi \implies \texttt{eval}(\tau) : \star}$
  - **Goal 3:**

    $\boxed{\Phi \& \texttt{eval}(\tau) \, \texttt{nf}}$

By IH on (2)

(4) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau) : \star$

(5) $\Psi; \Theta; \Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : \star$

(6) $\texttt{eval}(\tau)\ \texttt{nf}$

Goal 1 follows by K-Impl from (4) and (3)

Goal 2 follows by S-Impl from (5) and the fact that $\Theta; \Delta \vDash \Phi \rightarrow \Phi$

Goal 3 follows from (6)

- **Case 13:** *K-Monad.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \mathbb{M}\,(I, \vec{p})\,\tau : \star$

    (2) $\Psi; \Theta; \Delta \vdash_p \tau : \star$

    (3) $\Theta; \Delta \vdash_p I : \mathbb{N}$

    (4) $\Theta; \Delta \vdash_p \vec{p} : \vec{\mathbb{R}^+}$
  - **Goal 1:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \mathbb{M}\,(I, \vec{p})\,\texttt{eval}(\tau) : \star}$
  - **Goal 2:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \mathbb{M}\,(I, \vec{p})\,\tau \equiv \mathbb{M}\,(I, \vec{p})\,\texttt{eval}(\tau) : \star}$
  - **Goal 3:**

    $\boxed{\mathbb{M}\,(I, \vec{p})\,\texttt{eval}(\tau)\ \texttt{nf}}$

  By IH on (2)

  (5) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau) : \star$

  (6) $\Psi; \Theta; \Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : \star$

  (7) $\texttt{eval}(\tau)\ \texttt{nf}$

  Goal 1 follows by K-Monad on (5), (3), (5)

  Goal 2 follows by S-Monad on (6) and the fact that $\Theta; \Delta \vDash I = I \wedge \vec{p} \leq \vec{p}$

  Goal 3 follows from (7)

- **Case 14:** *K-Pot.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p [I|\vec{p}]\,\tau : \star$

    (2) $\Psi; \Theta; \Delta \vdash_p \tau : \star$

(3) $\Theta; \Delta \vdash_p I : \mathbb{N}$

(4) $\Theta; \Delta \vdash_p \vec{p} : \vec{\mathbb{R}^+}$

▸ **Goal 1:**

$\boxed{\Psi; \Theta; \Delta \vdash_p [I|\vec{p}]\, \texttt{eval}(\tau) : \star}$

▸ **Goal 2:**

$\boxed{\Psi; \Theta; \Delta \vdash_p [I|\vec{p}]\, \tau \equiv [I|\vec{p}]\, \texttt{eval}(\tau) : \star}$

▸ **Goal 3:**

$\boxed{[I|\vec{p}]\, \texttt{eval}(\tau)\ \texttt{nf}}$

By IH on (2)

(5) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau) : \star$

(6) $\Psi; \Theta; \Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : \star$

(7) $\texttt{eval}(\tau)\ \texttt{nf}$

Goal 1 follows by K-Pot on (5), (3), (4)

Goal 2 follows by S-Pot on (6) and the fact that $\Theta; \Delta \vDash I = I \wedge \vec{p} \leq \vec{p}$

Goal 3 follows from (7)

▸ **Case 15:** *K-ConstPot.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p [I]\, \tau : \star$

(2) $\Psi; \Theta; \Delta \vdash_p \tau : \star$

(3) $\Theta; \Delta \vdash_p I : \mathbb{R}$

▸ **Goal 1:**

$\boxed{\Psi; \Theta; \Delta \vdash_p [I]\, \texttt{eval}(\tau) : \star}$

▸ **Goal 2:**

$\boxed{\Psi; \Theta; \Delta \vdash_p [I]\, \tau \equiv [I]\, \texttt{eval}(\tau) : \star}$

▸ **Goal 3:**

$\boxed{[I]\, \texttt{eval}(\tau)\ \texttt{nf}}$

By IH on (2)

(4) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau) : \star$

(5) $\Psi; \Theta; \Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : \star$

(6) $\texttt{eval}(\tau)\ \texttt{nf}$

Goal 1 follows by K-ConstPot on (4), (3)

Goal 2 follows by S-Pot on (5) and the fact that $\Theta; \Delta \vDash I = I \wedge \vec{p} \leq \vec{p}$

Goal 3 follows from (6)

- **Case 15:** *K-FamLam.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \lambda i : S.\tau : S \to K$

    (2) $\Psi; \Theta, i : S; \Delta \vdash_p \tau : K$

  - **Goal 1:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \lambda i : S.\texttt{eval}(\tau) : S \to K}$

  - **Goal 2:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \lambda i : S.\tau \equiv \lambda i : S.\texttt{eval}(\tau) : S \to K}$

  - **Goal 3:**

    $\boxed{\lambda i : S.\texttt{eval}(\tau)\ \texttt{nf}}$

  By IH on (2)

  (3) $\Psi; \Theta, i : S; \Delta \vdash_p \texttt{eval}(\tau) : K$

  (4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : K$

  (5) $\texttt{eval}(\tau)\ \texttt{nf}$

  Goal 1 follows from K-FamLam on (3)

  Goal 2 follows from S-FamLam on (4)

  Goal 3 follows from (5)

- **Case 15:** *K-FamApp.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \tau\ I : K$

    (2) $\Psi; \Theta; \Delta \vdash_p \tau : S \to K$

    (3) $\Theta; \Delta \vdash_p I : S$

  - **Goal 1:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau\ I) : K}$

  - **Goal 2:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \tau\ I \equiv \texttt{eval}(\tau\ I) : K}$

  - **Goal 3:**

$\boxed{\texttt{eval}(\tau\ I)\ \texttt{nf}}$

By IH on (2)

(4) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau) : S \to K$

(5) $\Psi; \Theta; \Delta \vdash_p \tau \equiv \texttt{eval}(\tau) : S \to K$

(6) $\texttt{eval}(\tau)\ \texttt{nf}$

By Theorem 7.2 on (4), there are two possibilities for $\texttt{eval}(\tau)$.

▸ **Subcase 1:** *$\texttt{eval}(\tau)\ \texttt{ne}$.*

Here, $\texttt{eval}(\tau\ I) = \texttt{eval}(\tau)\ I$

Goal 1 follows from T-FamApp on (4) and (3)

Goal 2 follows from S-FamApp on (5) and (3) with $\Theta; \Theta \vDash I = I$

Goal 3 follows from the fact that $\texttt{eval}(\tau)\,\texttt{ne}$

▸ **Subcase 2:** *$\texttt{eval}(\tau) = \lambda i : S.\tau'$ and τ' $\texttt{nf}$.*

In this case, $\texttt{eval}(\tau\ I) = \tau'[I/i]$

From Theorem 7.2

(8) $\tau'\ \texttt{nf}$

We also know from (4) that

(9) $\Psi; \Theta; \Delta \vdash_p \lambda i : S.\tau' : S \to K$

Inverting (9)

(10) $\Psi; \Theta, i : S; \Delta \vdash_p \tau' : K$

Goal 1 follows from Theorem A.7 on (10) and (3)

By S-FamApp on (5)

(11) $\Psi; \Theta; \Delta \vdash_p \tau\ I \equiv (\lambda i : S.\tau')\ I : K$

By S-FamBeta{1,2}

(12) $\Psi; \Theta; \Delta \vdash_p (\lambda i : S.\tau')\ I \equiv \tau'[I/i] : K$

Goal 2 follows from applying S-Trans to (11) and (12)

Goal 3 follows from applying Theorem 7.1 to (8)

□

Proof of Theorem A.17

▸ **Given:**

(1) $\Theta; \Delta \vdash I : S \Rightarrow \Phi$

(2) $\Theta; \Delta \vDash \Phi$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash I : S}$

▸ **Case 1:** *AI-Var.*

Immediate.

▸ **Case 2:** *AI-Plus.*

▸ **Given:**

(1) $\Theta; \Delta \vdash I + J : bS \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Theta; \Delta \vdash I : bS \Rightarrow \Phi_1$

(4) $\Theta; \Delta \vdash J : bS \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash I + J : bS}$

By IH on (3), since by (2) $\Theta; \Delta \vDash \Phi_1$

(5) $\Theta; \Delta \vdash I : bS$

By IH on (4), since by (2) $\Theta; \Delta \vDash \Phi_2$

(6) $\Theta; \Delta \vdash J : bS$

Goal follows by I-Plus

▸ **Case 3:** *AI-Minus.*

▸ **Given:**

(1) $\Theta; \Delta \vdash I - J : bS \Rightarrow \Phi_1 \wedge \Phi_2 \wedge (I \geq J)$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2 \wedge (I \geq J)$

(3) $\Theta; \Delta \vdash I : bS \Rightarrow \Phi_1$

(4) $\Theta; \Delta \vdash J : bS \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash I - J : bS}$

By IH on (3), since by (2) $\Theta; \Delta \vDash \Phi_1$

(5) $\Theta; \Delta \vdash I : bS$

By IH on (4), since by (2) $\Theta; \Delta \vDash \Phi_2$

(6) $\Theta; \Delta \vdash J : bS$

Goal follows by I-Plus, using the fact that $\Theta; \Delta \vDash I \geq J$ from (2)

▸ **Case 4:** *AI-Times-*.*

Immediate by IH.

▸ **Case 5:** *AI-Shift.*

▸ **Given:**

(1) $\Theta; \Delta \vdash_{\triangleleft} I : \vec{\mathbb{R}}^{+} \Rightarrow \Phi$

(2) $\Theta; \Delta \vDash \Phi$

(3) $\Theta; \Delta \vdash I : \vec{\mathbb{R}}^{+} \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash_{\triangleleft} I : \vec{\mathbb{R}}^{+}}$

By IH on (3)

(4) $\Theta; \Delta \vdash I : \vec{\mathbb{R}}^{+}$

Goal follows by I-Shift on (4)

▸ **Case 6:** *AI-Lam.*

▸ **Given:**

(1) $\Theta; \Delta \vdash \lambda i : S.I : S \to S' \Rightarrow \forall i : S.\Phi$

(2) $\Theta; \Delta \vDash \forall i : S.\Phi$

(3) $\Theta, i : S; \Delta \vdash I : S' \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash \lambda i : S.I : S \to S'}$

Equivalently to (2)

(4) $\Theta, i : S; \Delta \vDash \Phi$

By IH on (3) with (4)

(5) $\Theta, i : S; \Delta \vdash I : S'$

Goal follows by I-Lam on (5)

▸ **Case 7:** *AI-App.*

▸ **Given:**

(1) $\Theta; \Delta \vdash I\ J : S' \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Theta; \Delta \vdash I : S \to S' \Rightarrow \Phi_1$

(4) $\Theta; \Delta \vdash J : S \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash I\ J : S'}$

By IH on (3)

(5) $\Theta; \Delta \vdash I : S \to S'$

By IH on (4)

(6) $\Theta; \Delta \vdash J : S$

Goal follows by I-App on (5) and (6)

▸ **Case 8:** *AI-Sum.*

▸ **Given:**

(1) $\Theta; \Delta \vdash \sum_{i=I_0}^{I_1} J : bS \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \forall i : \mathbb{N}.(I_0 \leq i \leq I_1 \to \Phi_3)$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2 \wedge \forall i : \mathbb{N}.(I_0 \leq i \leq I_1 \to \Phi_3)$

(3) $\Theta; \Delta \vdash I_0 : \mathbb{N} \Rightarrow \Phi_1$

(4) $\Theta; \Delta \vdash I_1 : \mathbb{N} \Rightarrow \Phi_2$

(5) $\Theta, i : \mathbb{N}; \Delta, I_0 \leq i \leq I_1 \vdash J : bS \Rightarrow \Phi_3$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash \sum_{i=I_0}^{I_1} J : bS}$

From (2)

(6) $\Theta, i : \mathbb{N}; \Delta, (I_0 \leq i \leq I_1) \vDash \Phi_3$

By IH on (5) using (6)

(7) $\Theta, i : \mathbb{N}; \Delta, I_0 \leq i \leq I_1 \vdash J : bS$

By IH on (3)

(8) $\Theta; \Delta \vdash I_0 : \mathbb{N}$

By IH on (4)

(9) $\Theta; \Delta \vdash I_1 : \mathbb{N}$

Goal follows by I-Sum on (7), (8), (9)

- **Case 9:** *AI-*-Lit.*

  Immediate.

□

Proof of Theorem A.18

- **Given:**

  (1) $\Theta; \Delta \vdash \Phi \texttt{ wf} \Rightarrow \Phi'$

  (2) $\Theta; \Delta \vDash \Phi'$

- **Goal:**

  $\boxed{\Theta; \Delta \vdash \Phi \texttt{ wf}}$

By induction on $\Theta; \Delta \vdash \Phi \texttt{ wf} \Rightarrow \Phi'$

- **Case 1:** *AC-Top.*

  Immediate.

- **Case 2:** *AC-Bot.*

  Immediate.

- **Case 3:** *AC-Conj.*

  - **Given:**

    (1) $\Theta; \Delta \vdash \Phi_1 \wedge \Phi_2 \texttt{ wf} \Rightarrow \Phi_1' \wedge \Phi_2'$

    (2) $\Theta; \Delta \vDash \Phi_1' \wedge \Phi_2'$

    (3) $\Theta; \Delta \vdash \Phi_1 \texttt{ wf} \Rightarrow \Phi_1'$

    (4) $\Theta; \Delta \vdash \Phi_2 \texttt{ wf} \Rightarrow \Phi_2'$

  - **Goal:**

    $\boxed{\Theta; \Delta \vdash \Phi_1 \wedge \Phi_2 \texttt{ wf}}$

  By IH on (3)

  (5) $\Theta; \Delta \vdash \Phi_1 \texttt{ wf}$

  By IH on (4)

  (6) $\Theta; \Delta \vdash \Phi_2 \texttt{ wf}$

  Goal follows by C-Conj on (5) and (6)

▸ **Case 4:** *AC-Disj.*

▸ **Given:**

(1) $\Theta; \Delta \vdash \Phi_1 \vee \Phi_2 \texttt{ wf} \Rightarrow \Phi_1' \wedge \Phi_2'$

(2) $\Theta; \Delta \vDash \Phi_1' \wedge \Phi_2'$

(3) $\Theta; \Delta \vdash \Phi_1 \texttt{ wf} \Rightarrow \Phi_1'$

(4) $\Theta; \Delta \vdash \Phi_2 \texttt{ wf} \Rightarrow \Phi_2'$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash \Phi_1 \vee \Phi_2 \texttt{ wf}}$

By IH on (3)

(5) $\Theta; \Delta \vdash \Phi_1 \texttt{ wf}$

By IH on (4)

(6) $\Theta; \Delta \vdash \Phi_2 \texttt{ wf}$

Goal follows by C-Disj on (5) and (6)

▸ **Case 5:** *AC-Impl.*

▸ **Given:**

(1) $\Theta; \Delta \vdash \Phi_1 \to \Phi_2 \texttt{ wf} \Rightarrow \Phi_1' \wedge (\Phi_1 \to \Phi_2')$

(2) $\Theta; \Delta \vDash \Phi_1' \wedge (\Phi_1 \to \Phi_2')$

(3) $\Theta; \Delta \vdash \Phi_1 \texttt{ wf} \Rightarrow \Phi_1'$

(4) $\Theta; \Delta, \Phi_1 \vdash \Phi_2 \texttt{ wf} \Rightarrow \Phi_2'$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash \Phi_1 \to \Phi_2 \texttt{ wf}}$

By IH on (3)

(5) $\Theta; \Delta \vdash \Phi_1 \texttt{ wf}$

From (2)

(6) $\Theta; \Delta, \Phi_1 \vDash \Phi_2'$

By IH on (4) with (6)

(7) $\Theta; \Delta, \Phi_1 \vdash \Phi_2 \texttt{ wf}$

Goal follows by C-Impl on (5) and (7)

▸ **Case 6:** *AC-Forall.*

- **Given:**

  (1) $\Theta; \Delta \vdash \forall i : S.\Phi\ \texttt{wf} \Rightarrow \forall i : S.\Phi'$

  (2) $\Theta; \Delta \vDash \forall i : S.\Phi'$

  (3) $\Theta, i : S; \Delta \vdash \Phi\ \texttt{wf} \Rightarrow \Phi'$

- **Goal:**

  $\boxed{\Theta; \Delta \vdash \forall i : S.\Phi\ \texttt{wf}}$

Equivalently to (2)

(4) $\Theta, i : S; \Delta \vDash \Phi'$

By IH on (3) with (4)

(5) $\Theta, i : S; \Delta \vdash \Phi\ \texttt{wf}$

Goal follows from C-Forall on (5)

- **Case 7:** *AC-Exists.*

- **Given:**

  (1) $\Theta; \Delta \vdash \exists i : S.\Phi\ \texttt{wf} \Rightarrow \forall i : S.\Phi'$

  (2) $\Theta; \Delta \vDash \exists i : S.\Phi'$

  (3) $\Theta, i : S; \Delta \vdash \Phi\ \texttt{wf} \Rightarrow \Phi'$

- **Goal:**

  $\boxed{\Theta; \Delta \vdash \exists i : S.\Phi\ \texttt{wf}}$

Equivalently to (2)

(4) $\Theta, i : S; \Delta \vDash \Phi'$

By IH on (3) with (4)

(5) $\Theta, i : S; \Delta \vdash \Phi\ \texttt{wf}$

Goal follows from C-Exists on (5)

- **Case 8:** *AC-Eq.*

- **Given:**

  (1) $\Theta; \Delta \vdash I = J\ \texttt{wf} \Rightarrow \Phi_1 \wedge \Phi_2$

  (2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

  (3) $\Theta; \Delta \vdash I : bS \Rightarrow \Phi_1$

(4) $\Theta; \Delta \vdash J : bS \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Theta; \Delta \vdash I = J \ \mathtt{wf}}$

By Theorem A.17 on (3)

(5) $\Theta; \Delta \vdash I : bS$

By Theorem A.17 on (4)

(6) $\Theta; \Delta \vdash J : bS$

Goal follows by C-Eq on (5) and (6)

▸ **Case 9:** *AC-Leq.*

Identical to Case 8

▸ **Case 10:** *AC-Lt.*

Identical to Case 8

□

Proof of Theorem A.19

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau : K \Rightarrow \Phi$

(2) $Theta; \Delta \vDash \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \ tau : K}$

By Induction on $\Psi; \Theta; \Delta \vdash \tau : K \Rightarrow \Phi$

▸ **Case 1:** *AK-Var.*

Immediate.

▸ **Case 2:** *AK-Unit.*

Immediate.

▸ **Case 3:** *AK-Arr.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1 \multimap \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 : \star \Rightarrow \Phi_1$

(4) $\Psi; \Theta; \Delta \vdash \tau_2 : \star \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \tau_1 \multimap \tau_2 : \star}$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash \tau_1 : \star$

By IH on (4)

(6) $\Psi; \Theta; \Delta \vdash \tau_2 : \star$

Goal follows by K-Arr on (5) and (6)

▸ **Case 4:** *AK-Tensor.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 : \star \Rightarrow \Phi_1$

(4) $\Psi; \Theta; \Delta \vdash \tau_2 : \star \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 : \star}$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash \tau_1 : \star$

By IH on (4)

(6) $\Psi; \Theta; \Delta \vdash \tau_2 : \star$

Goal follows by K-Tensor on (5) and (6)

▸ **Case 5:** *AK-With.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1 \& \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 : \star \Rightarrow \Phi_1$

(4) $\Psi; \Theta; \Delta \vdash \tau_2 : \star \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash \tau_1 \& \tau_2 : \star}$

By IH on (3)

(5) $\Psi;\Theta;\Delta \vdash \tau_1 : \star$

By IH on (4)

(6) $\Psi;\Theta;\Delta \vdash \tau_2 : \star$

Goal follows by K-With on (5) and (6)

▸ **Case 6:** *AK-Sum.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash \tau_1 \oplus \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi;\Theta;\Delta \vdash \tau_1 : \star \Rightarrow \Phi_1$

(4) $\Psi;\Theta;\Delta \vdash \tau_2 : \star \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash \tau_1 \oplus \tau_2 : \star}$

By IH on (3)

(5) $\Psi;\Theta;\Delta \vdash \tau_1 : \star$

By IH on (4)

(6) $\Psi;\Theta;\Delta \vdash \tau_2 : \star$

Goal follows by K-Sum on (5) and (6)

▸ **Case 7:** *AK-Bang.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash !\tau : \star \Rightarrow \Phi$

(2) $\Theta;\Delta \vDash \Phi$

(3) $\Psi;\Theta;\Delta \vdash \tau : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash !\tau : \star}$

By IH on (3)

(4) $\Psi;\Theta;\Delta \vdash \tau : \star$

Goal follows by K-Bang on (5)

▸ **Case 8:** *AK-IForall.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \forall i : S.\tau : \star \Rightarrow \forall i : S.\Phi$

(2) $\Theta; \Delta \vDash \forall i : S.\Phi$

(3) $\Psi; \Theta, i : S; \Delta \vdash \tau : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \forall i : S.\tau : \star}$

Equivalently to (2)

(5) $\Theta, i : S; \Delta \vDash \Phi$

By IH on (3) using (5)

(6) $\Psi; \Theta, i : S; \Delta \vdash \tau : \star$

Goal follows by K-IForall on (6)

▸ **Case 9:** *AK-IExists.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \exists i : S.\tau : \star \Rightarrow \forall i : S.\Phi$

(2) $\Theta; \Delta \vDash \forall i : S.\Phi$

(3) $\Psi; \Theta, i : S; \Delta \vdash \tau : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \exists i : S.\tau : \star}$

Equivalently to (2)

(5) $\Theta, i : S; \Delta \vDash \Phi$

By IH on (3) using (5)

(6) $\Psi; \Theta, i : S; \Delta \vdash \tau : \star$

Goal follows by K-IExists on (6)

▸ **Case 10:** *AK-TForall.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \forall \alpha : K.\tau : \star \Rightarrow \Phi$

(2) $\Theta; \Delta \vDash \Phi$

(3) $\Psi, \alpha : K; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi$

- **Goal:**

  $\boxed{\Psi; \Theta; \Delta \vdash \forall \alpha : K.\tau : \star}$

  By IH on (3)

  (4) $\Psi, \alpha : K; \Theta; \Delta \vdash \tau : \star$

  Goal follows by K-TForall on (5)

- **Case 11:** *AK-List.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash L^I \tau : \star \Rightarrow \Phi_1 \wedge \Phi_2$

    (2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

    (3) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_1$

    (4) $\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi 2$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash L^I \tau : \star}$

  By IH on (3)

  (5) $\Psi; \Theta; \Delta \vdash \tau : \star$

  By Theorem A.17 on (4)

  (6) $\Theta; \Delta \vdash I : \mathbb{N}$

  Goal follows by K-List on (5) and (6)

- **Case 12:** *AK-Conj.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash \Phi \& \tau : \star \Rightarrow \Phi_1 \wedge \Phi_2$

    (2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

    (3) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_1$

    (4) $\Theta; \Delta \vdash \Phi \ \texttt{wf} \Rightarrow \Phi 2$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash \Phi \& \tau : \star}$

  By IH on (3)

  (5) $\Psi; \Theta; \Delta \vdash \tau : \star$

  By Theorem A.18 on (4)

(6) $\Theta; \Delta \vdash \Phi\ \texttt{wf}$

Goal follows by K-Conj on (5) and (6)

▸ **Case 13:** *AK-Impl.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \Phi \implies \tau : \star \Rightarrow \Phi_1 \land (\Phi \rightarrow \Phi_2)$

(2) $\Theta; \Delta \vDash \Phi_1 \land (\Phi \rightarrow \Phi_2)$

(3) $\Theta; \Delta \vdash \Phi\ \texttt{wf} \Rightarrow \Phi 1$

(4) $\Psi; \Theta; \Delta, \Phi \vdash \tau : \star \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \Phi \implies \tau : \star}$

By Theorem A.18 on (4)

(5) $\Theta; \Delta \vdash \Phi\ \texttt{wf}$

From (2)

(6) $\Theta; \Delta, \Phi \vDash \Phi_2$

By IH on (4) with (6)

(7) $\Psi; \Theta; \Delta, \Phi \vdash \tau : \star$

Goal follows by K-Impl on (5) and (7)

▸ **Case 14:** *AK-Monad.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \mathbb{M}\,(I, \vec{p})\,\tau : \star \Rightarrow \Phi_1 \land \Phi_2 \land \Phi_3$

(2) $\Theta; \Delta \vDash \Phi_1 \land \Phi_2 \land \Phi_3$

(3) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_1$

(4) $\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi_2$

(5) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \Rightarrow \Phi_3$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \mathbb{M}\,(I, \vec{p})\,\tau : \star}$

By IH on (3)

(6) $\Psi; \Theta; \Delta \vdash \tau : \star$

By Theorem A.17 on (4)

(7) $\Theta; \Delta \vdash I : \mathbb{N}$

By Theorem A.17 on (5)

(8) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+}$

Goal follows by K-Monad on (6), (7), and (8)

- **Case 15:** *AK-Pot.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash [I|\vec{p}]\,\tau : \star \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3$

    (2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2 \wedge \Phi_3$

    (3) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_1$

    (4) $\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi_2$

    (5) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \Rightarrow \Phi_3$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash [I|\vec{p}]\,\tau : \star}$

  By IH on (3)

  (6) $\Psi; \Theta; \Delta \vdash \tau : \star$

  By Theorem A.17 on (4)

  (7) $\Theta; \Delta \vdash I : \mathbb{N}$

  By Theorem A.17 on (5)

  (8) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+}$

  Goal follows by K-Pot on (6), (7), and (8)

- **Case 16:** *AK-ConstPot.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash [I]\,\tau : \star \Rightarrow \Phi_1 \wedge \Phi_2$

    (2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

    (3) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_1$

    (4) $\Theta; \Delta \vdash I : \mathbb{R} \Rightarrow \Phi_2$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash [I]\,\tau : \star}$

  By IH on (3)

(5) $\Psi;\Theta;\Delta \vdash \tau : \star$

By Theorem A.17 on (4)

(6) $\Theta;\Delta \vdash I : \mathbb{R}$

Goal follows by K-ConstPot on (5), and (6)

- **Case 17:** *AK-FamLam.*

  - **Given:**

    (1) $\Psi;\Theta;\Delta \vdash \lambda i : S.\tau : S \to K \Rightarrow \forall i : S.\Phi$

    (2) $\Theta;\Delta \vDash \forall i : S.\Phi$

    (3) $\Psi;\Theta, i : S;\Delta \vdash \tau : K \Rightarrow \Phi$

  - **Goal:**

    $\boxed{\Psi;\Theta;\Delta \vdash \lambda i : S.\tau : S \to K}$

  Equivalently to (2)

  (4) $\Theta, i : S;\Delta \vDash \Phi$

  By IH on (3) with (4)

  (5) $\Psi;\Theta, i : S;\Delta \vdash \tau : K$

  Goal follows by K-FamLam on (5)

- **Case 18:** *AK-FamApp.*

  - **Given:**

    (1) $\Psi;\Theta;\Delta \vdash \tau\ I : K \Rightarrow \Phi_1 \wedge \Phi_2$

    (2) $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$

    (3) $\Psi;\Theta;\Delta \vdash \tau : S \to K \Rightarrow \Phi_1$

    (4) $\Theta;\Delta \vdash I : S \Rightarrow \Phi_2$

  - **Goal:**

    $\boxed{\Psi;\Theta;\Delta \vdash \tau\ I : K}$

  By IH on (3)

  (5) $\Psi;\Theta;\Delta \vdash \tau : S \to K$

  By Theorem A.17 on (4)

  (6) $\Theta;\Delta \vdash I : S$

  Goal follows by K-FamApp on (5) and (6)

$\square$

Proof of Theorem A.20

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : K \Rightarrow \Phi$

(2) $\Theta; \Delta \vDash \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : K}$

▸ **Case 1:** *AS-Unit.*

Immediate.

▸ **Case 2:** *AS-Var.*

Immediate.

▸ **Case 3:** *AS-Arr.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1 \multimap \tau_2 <:_{\mathtt{nf}} \tau_1' \multimap \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi; \Theta; \Delta \vdash \tau_1' <:_{\mathtt{nf}} \tau_1 : \star \Rightarrow \Phi_1$

(4) $\Psi; \Theta; \Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_2' : \star \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \tau_1 \multimap \tau_2 <: \tau_1' \multimap \tau_2' : \star}$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash \tau_1' <: \tau_1 : \star$

By IH on (4)

(6) $\Psi; \Theta; \Delta \vdash \tau_2 <: \tau_2' : \star$

Goal follows by S-Arr on (5) and (6)

▸ **Case 4:** *AS-Tensor.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 <:_{\mathtt{nf}} \tau_1' \otimes \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_1' : \star \Rightarrow \Phi_1$

(4) $\Psi; \Theta; \Delta \vdash \tau_2 <:_{\texttt{nf}} \tau_2' : \star \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 <: \tau_1' \otimes \tau_2' : \star}$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash \tau_1' <: \tau_1 : \star$

By IH on (4)

(6) $\Psi; \Theta; \Delta \vdash \tau_2 <: \tau_2' : \star$

Goal follows by S-Tensor on (5) and (6)

▸ **Case 5:** *AS-With.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1 \& \tau_2 <:_{\texttt{nf}} \tau_1' \& \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_1' : \star \Rightarrow \Phi_1$

(4) $\Psi; \Theta; \Delta \vdash \tau_2 <:_{\texttt{nf}} \tau_2' : \star \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \tau_1 \& \tau_2 <: \tau_1' \& \tau_2' : \star}$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash \tau_1' <: \tau_1 : \star$

By IH on (4)

(6) $\Psi; \Theta; \Delta \vdash \tau_2 <: \tau_2' : \star$

Goal follows by S-With on (5) and (6)

▸ **Case 6:** *AS-Sum.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1 \oplus \tau_2 <:_{\texttt{nf}} \tau_1' \oplus \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_1' : \star \Rightarrow \Phi_1$

(4) $\Psi; \Theta; \Delta \vdash \tau_2 <:_{\texttt{nf}} \tau_2' : \star \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \tau_1 \oplus \tau_2 <: \tau_1' \oplus \tau_2' : \star}$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash \tau_1' <: \tau_1 : \star$

By IH on (4)

(6) $\Psi; \Theta; \Delta \vdash \tau_2 <: \tau_2' : \star$

Goal follows by S-Sum on (5) and (6)

▸ **Case 7:** *AS-Bang.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash !\tau_1 <:_{\texttt{nf}} !\tau_2 : \star \Rightarrow \Phi$

(2) $\Theta; \Delta \vDash \Phi$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash !\tau_1 <: !\tau_2 : \star}$

By IH on (3)

(4) $\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : \star$

Goal follows by S-Bang on (4)

▸ **Case 8:** *AS-IForall.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \forall i : S.\tau_1 <:_{\texttt{nf}} \forall i : S.\tau_2 : \star \Rightarrow \forall i : S.\Phi$

(2) $\Theta; \Delta \vDash \forall i : S.\Phi$

(3) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \forall i : S.\tau_1 <: \forall i : S.\tau_2 : \star}$

Equivalently to (2)

(4) $\Theta, i : S; \Delta \vDash \Phi$

By IH on (3), using (4)

(5) $\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : \star$

Goal follows by S-IForall on (5)

▸ **Case 9:** *AS-IExists.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \exists i : S.\tau_1 <:_{\texttt{nf}} \exists i : S.\tau_2 : \star \Rightarrow \forall i : S.\Phi$

(2) $\Theta; \Delta \vDash \forall i : S.\Phi$

(3) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \exists i : S.\tau_1 <: \exists i : S.\tau_2 : \star}$

Equivalently to (2)

(4) $\Theta, i : S; \Delta \vDash \Phi$

By IH on (3), using (4)

(5) $\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : \star$

Goal follows by S-IExists on (5)

▸ **Case 10:** *AS-TForall.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \forall \alpha : K.\tau_1 <:_{\texttt{nf}} \forall \alpha : K.\tau_2 : \star \Rightarrow \Phi$

(2) $\Theta; \Delta \vDash \Phi$

(3) $\Psi, \alpha : K; \Theta; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \forall \alpha : K.\tau_1 <: \forall \alpha : K.\tau_2 : \star}$

By IH on (3)

(4) $\Psi, \alpha : K; \Theta; \Delta \vdash \tau_1 <: \tau_2 : \star$

Goal follows by S-TForall on (4)

▸ **Case 11:** *AS-List.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash L^I \tau_1 <:_{\texttt{nf}} L^J \tau_2 : \star \Rightarrow \Phi \wedge (I = J)$

(2) $\Theta; \Delta \vDash \Phi \wedge (I = J)$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash L^I \tau_1 <: L^J \tau_2 : \star}$

By IH on (3)

(4) $\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

From (2)

(5) $\Theta; \Delta \vDash I = J$

Goal follows by S-List on (4) and (5)

▸ **Case 12:** *AS-Conj.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \Phi_1 \& \tau_1 <:_{\texttt{nf}} \Phi_2 \& \tau_2 : \star \Rightarrow \Phi \wedge (\Phi_1 \rightarrow \Phi_2)$

(2) $\Theta; \Delta \vDash \Phi \wedge (\Phi_1 \rightarrow \Phi_2)$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \Phi_1 \& \tau_1 <: \Phi_2 \& \tau_2 : \star}$

By IH on (3)

(4) $\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

From (2)

(5) $\Theta; \Delta \vDash \Phi_1 \rightarrow \Phi_2$

Goal follows by S-Conj on (4) and (5)

▸ **Case 13:** *AS-Impl.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \Phi_1 \implies \tau_1 <:_{\texttt{nf}} \Phi_2 \implies \tau_2 : \star \Rightarrow (\Phi_2 \rightarrow \Phi) \wedge (\Phi_2 \rightarrow \Phi_1)$

(2) $\Theta; \Delta \vDash (\Phi_2 \rightarrow \Phi) \wedge (\Phi_2 \rightarrow \Phi_1)$

(3) $\Psi; \Theta; \Delta, \Phi_2 \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \Phi_1 \implies \tau_1 <: \Phi_2 \implies \tau_2 : \star}$

By IH on (3)

(4) $\Psi; \Theta; \Delta, \Phi_2 \vdash \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

From (2)

(5) $\Theta; \Delta \vDash (\Phi_2 \rightarrow \Phi) \wedge (\Phi_2 \rightarrow \Phi_1)$

Goal follows by S-Impl on (4) and (5)

▸ **Case 14:** *AS-Monad.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash \mathbb{M}\,(I,\vec{p})\,\tau_1 <:_{\texttt{nf}} \mathbb{M}\,(J,\vec{q})\,\tau_2 : \star \Rightarrow \Phi \wedge (I = J) \wedge (\vec{p} \leq \vec{q})$

(2) $\Theta;\Delta \vDash \Phi \wedge (I = J) \wedge (\vec{p} \leq \vec{q})$

(3) $\Psi;\Theta;\Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash \mathbb{M}\,(I,\vec{p})\,\tau_1 <: \mathbb{M}\,(J,\vec{q})\,\tau_2 : \star}$

By IH on (3)

(4) $\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : \star$

From (2)

(5) $\Theta;\Delta \vDash I = J$

(6) $\Theta;\Delta \vDash \vec{p} \leq \vec{q}$

Goal follows by S-Monad on (4), (5), and (6)

▸ **Case 15:** *AS-Pot.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash [I|\vec{p}]\,\tau_1 <:_{\texttt{nf}} [J|\vec{q}]\,\tau_2 : \star \Rightarrow \Phi \wedge (I = J) \wedge (\vec{p} \geq \vec{q})$

(2) $\Theta;\Delta \vDash \Phi \wedge (I = J) \wedge (\vec{p} \geq \vec{q})$

(3) $\Psi;\Theta;\Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash [I|\vec{p}]\,\tau_1 <: [J|\vec{q}]\,\tau_2 : \star}$

By IH on (3)

(4) $\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : \star$

From (2)

(5) $\Theta;\Delta \vDash I = J$

(6) $\Theta;\Delta \vDash \vec{p} \geq \vec{q}$

Goal follows by S-Pot on (4), (5), and (6)

▸ **Case 16:** *AS-ConstPot.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash [I]\,\tau_1 <:_{\texttt{nf}} [J]\,\tau_2 : \star \Rightarrow \Phi \wedge (I \geq J)$

(2) $\Theta; \Delta \vDash \Phi \wedge (I \geq J)$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi$

- **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash [I]\, \tau_1 <: [J]\, \tau_2 : \star}$

By IH on (3)

(4) $\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : \star$

From (2)

(5) $\Theta; \Delta \vDash I \geq J$

Goal follows by S-Const on (4) and (5)

- **Case 17:** *AS-FamLam.*

- **Given:**

(1) $\Psi; \Theta; \Delta \vdash \lambda i : S.\tau_1 <:_{\mathtt{nf}} \lambda i : S.\tau_2 : S \to K \Rightarrow \forall i : S.\Phi$

(2) $\Theta; \Delta \vDash \forall i : S.\Phi$

(3) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : K \Rightarrow \Phi$

- **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \lambda i : S.\tau_1 <: \lambda i : S.\tau_2 : S \to K}$

Equivalently to (2)

(4) $\Theta, i : S; \Delta \vDash \Phi$

By IH on (3), using (4)

(5) $\Psi; \Theta, i : S : \Delta \vdash \tau_1 <: \tau_2 : K$

Goal follows by S-FamLam on (5)

- **Case 18:** *AS-FamApp.*

- **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1\ I <:_{\mathtt{nf}} \tau_2\ J : K \Rightarrow (I = J) \wedge \Phi$

(2) $\Theta; \Delta \vDash (I = J) \wedge \Phi$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : S \to K \Rightarrow \Phi$

- **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \tau_1\ I <: \tau_2\ J : K}$

By IH on (3)

(4) $\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2; S \to K$

From (2)

(5) $\Theta;\Delta \vDash I = J$

Goal follows by S-FamApp on (4) and (5).

□

Proof of Theorem A.24

We prove both claims simultaneously by induction over $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi,\Gamma'$ and $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \tau \Rightarrow \Phi,\Gamma'$. For brevity, we will often silently invoke Theorem 8.7 and Theorem 5.6 silently to build context weakening judgments for T-Weaken.

- **Case 1:** *AT-Var-1.*

  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash x \uparrow \tau \Rightarrow \top,\Gamma \setminus \{x:\tau\}$

    (2) $\Theta;\Delta \vDash \top$

    (3) $x:\tau \in \Gamma$

  - **Goal:**

    $\boxed{\Psi;\Theta;\Delta;\Omega;x:\tau \vdash x:\tau}$

  Immediate by T-Var-1 on (3)

- **Case 2:** *AT-Var-2.*

  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash x \uparrow \tau \Rightarrow \top,\Gamma$

    (2) $\Theta;\Delta \vDash \top$

    (3) $x:\tau \in \Omega$

  - **Goal:**

    $\boxed{\Psi;\Theta;\Delta;\Omega;\cdot \vdash x:\tau}$

  Immediate by T-Var-2 on (3)

- **Case 3:** *AT-Unit.*

  Immediate.

- **Case 4:** *AT-Base.*

Immediate.

▸ **Case 5:** *AT-Absurd.*

Immediate.

▸ **Case 6:** *AT-Nil.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{nil} \downarrow L^I \tau \Rightarrow I = 0, \Gamma$

(2) $\Theta; \Delta \vDash I = 0$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta; \cdot \vdash \texttt{nil} : L^I \tau}$

By T-Nil

(3) $\Psi; \Theta; \Delta; \cdot \vdash \texttt{nil} : L^0 \tau$

By S-List, S-Refl and (2)

(4) $\Psi; \Theta; \Delta \vdash L^0 \tau <: L^I \tau : \star$

Goal follows by T-Sub on (3) and (4)

▸ **Case 7:** *AT-Cons.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 :: e_2 \downarrow L^I \tau \Rightarrow (I \geq 1) \wedge \Phi_1 \wedge \Phi_2, \Gamma_2$

(2) $\Theta; \Delta \vDash (I \geq 1) \wedge \Phi_1 \wedge \Phi_2$

(3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 \downarrow \tau \Rightarrow \Phi_1, \Gamma_1$

(4) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_2 \downarrow L^{I-1} \tau \Rightarrow \Phi_2, \Gamma_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma_2 \vdash |e_1| :: |e_2| : L^I \tau}$

By IH on (3)

(5) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma_1 \vdash |e_1| : \tau$

By IH on (4)

(6) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \setminus \Gamma_2 \vdash |e_2| : L^{I-1} \tau$

By T-Cons on (5), (6)

(7) $\Psi; \Theta; \Delta; \Omega; (\Gamma \setminus \Gamma_1), (\Gamma_1 \setminus \Gamma_2) \vdash |e_1| :: |e_2| : L^{(I-1)+1} \tau$

Since $\Theta; \Delta \vDash (I - 1) + 1 = I$, by S-List

(8) $\Psi;\Theta;\Delta \vdash L^{(I-1)+1}\,\tau <: L^I\,\tau : \star$

By T-Sub on (7) and (8)

(9) $\Psi;\Theta;\Delta;\Omega;(\Gamma \setminus \Gamma_1),(\Gamma_1 \setminus \Gamma_2) \vdash |e_1| :: |e_2| : L^I\,\tau$

Goal follows by T-Weaken on (9)

▸ **Case 8:** *AT-Match.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{match}(e,e_1,h.t.e_2) \downarrow \tau' \Rightarrow \Phi_1 \wedge \Phi_{\texttt{body}},\Gamma'$

(2) $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_{\texttt{body}}$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow L^I\tau \Rightarrow \Phi_1,\Gamma_1$

(4) $\Psi;\Theta;\Delta,I=0;\Omega;\Gamma_1 \vdash e_1 \downarrow \tau' \Rightarrow \Phi_2,\Gamma_2$

(5) $\Psi;\Theta;\Delta,I\geq 1;\Omega;\Gamma_1,h:\tau,t:L^{I-1}\tau \vdash e_2 \downarrow \tau' \Rightarrow \Phi_3,\Gamma_3$

(6) $\Phi_{\texttt{body}} = (I=0 \to \Phi_2) \wedge (I \geq 1 \to \Phi_3)$

(7) $\Gamma' = \Gamma_2 \cap (\Gamma_3 \setminus \{h,t\})$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma' \vdash \texttt{match}(|e|,|e_1|,h.t.|e_2|) : \tau'}$$

By IH on (3)

(8) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma_1 \vdash |e| : L^I\tau$

By IH on (4)

(9) $\Psi;\Theta;\Delta,I=0;\Omega;\Gamma_1 \setminus \Gamma_2 \vdash |e_1| : \tau'$

By IH on (5)

(10) $\Psi;\Theta;\Delta,I\geq 1;\Omega;(\Gamma_1,h:\tau,t:L^{I-1}\tau) \setminus \Gamma_3 \vdash |e_2| : \tau'$

By two applications of T-Weaken on (9) and (10)

(11) $\Psi;\Theta;\Delta,I=0;\Omega;\Gamma_1 \setminus \Gamma' \vdash |e_1| : \tau'$

(12) $\Psi;\Theta;\Delta,I\geq 1;\Omega;\Gamma_1 \setminus \Gamma',h:\tau,t:L^{I-1}\tau \vdash |e_2| : \tau'$

By T-Match on (8), (11), (12)

(13) $\Psi;\Theta;\Delta;\Omega;(\Gamma \setminus \Gamma_1),(\Gamma_1 \setminus \Gamma') \vdash \texttt{match}(|e|,e_1,h.t.|e_2|) : \tau'$

The Goal follows from T-Weakening on (13)

▸ **Case 9:** *AT-ExistI.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{pack}[I](e) \downarrow \exists i:S.\tau \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma'$

(2) $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Theta;\Delta \vdash I : S \Rightarrow \Phi_1$

(4) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau[I/i] \Rightarrow \Phi_2, \Gamma'$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma' \vdash \texttt{pack}[I](|e|) : \exists i:S.\tau}$

By Theorem A.17 on (3)

(5) $\Theta;\Delta \vdash I : S$

By IH on (4)

(6) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma' \vdash |e| : \tau[I/i]$

Goal follows by T-ExistI on (5) and (6)

▸ **Case 10:** *AT-ExistE.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{unpack}\ (i,x) = e\ \texttt{in}\ e' \downarrow \tau' \Rightarrow \Phi, \Gamma_2 \setminus \{x\}$

(2) $\Theta;\Delta \vDash \Phi$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \exists i:S.\tau \Rightarrow \Phi_1, \Gamma_1$

(4) $\Psi;\Theta,i:S;\Delta;\Omega;\Gamma_1, x:\tau \vdash e' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2$

(5) $\Phi = \Phi_1 \wedge (\forall i:S.\Phi_2)$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \setminus (\Gamma_2 \setminus \{x\}) \vdash \texttt{unpack}\ (i,x) = |e|\ \texttt{in}\ |e'| : \tau'}$

By IH on (3)

(6) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma_1 \vdash |e| : \exists i:S.\tau$

From (2) and (5)

(7) $\Theta, i:S;\Delta \vDash \Phi_2$

By IH on (4) using (7)

(8) $\Psi;\Theta,i:S;\Delta;\Omega;(\Gamma_1, x:\tau) \setminus \Gamma_2 \vdash e' : \tau'$

By T-Weaken on (8)

(9) $\Psi;\Theta,i:S;\Delta;\Omega;(\Gamma_1 \setminus (\Gamma_2 \setminus \{x:\tau\})), x:\tau \vdash |e'| : \tau'$

By T-ExistE on (6) and (9)

(10) $\Psi;\Theta;\Delta;\Omega;(\Gamma\setminus\Gamma_1),(\Gamma_1\setminus(\Gamma_2\setminus\{x:\tau\}))\vdash \texttt{unpack}\ (i,x)=|e|\ \texttt{in}\ |e'|:\tau'$

Goal follows by T-Weaken on (10)

▸ **Case 11:** *AT-Lam.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma\vdash\lambda x.e\downarrow\tau_1\multimap\tau_2\Rightarrow\Phi,\Gamma'\setminus\{x:\tau_1\}$

(2) $\Theta;\Delta\vDash\Phi$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma,x:\tau_1\vdash e\downarrow\tau_2,\Rightarrow\Phi,\Gamma'$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma\setminus(\Gamma'\setminus\{x:\tau_1\})\vdash\lambda x.|e|:\tau_1\multimap\tau_2}$

By IH on (2)

(4) $\Psi;\Theta;\Delta;\Omega;(\Gamma,x:\tau_1)\setminus\Gamma'\vdash|e|:\tau_2$

By T-Weaken on (4)

(5) $\Psi;\Theta;\Delta;\Omega;(\Gamma\setminus(\Gamma'\setminus\{x:\tau_1\})),x:\tau_1\vdash|e|:\tau_2$

Goal follows by T-Lam on (5)

▸ **Case 12:** *AT-App.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma\vdash e_1\,e_2\uparrow\tau_2\Rightarrow\Phi_1\wedge\Phi_2,\Gamma_2$

(2) $\Theta;\Delta\vDash\Phi_1\wedge\Phi_2$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma\vdash e_1\uparrow\tau_1\multimap\tau_2\Rightarrow\Phi_1,\Gamma_1$

(4) $\Psi;\Theta;\Delta;\Omega;\Gamma_1\vdash e_2\downarrow\tau_1\Rightarrow\Phi_2,\Gamma_2$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma\setminus\Gamma_2\vdash|e_1|\,|e_2|:\tau_2}$

By IH on (3)

(5) $\Psi;\Theta;\Delta;\Omega;\Gamma\setminus\Gamma_1\vdash|e_1|:\tau_1\multimap\tau_2$

By IH on (4)

(6) $\Psi;\Theta;\Delta;\Omega;\Gamma_1\setminus\Gamma_2\vdash|e_2|:\tau_1$

By T-App on (5) and (6)

(7) $\Psi;\Theta;\Delta;\Omega;(\Gamma\setminus\Gamma_1),(\Gamma_1\setminus\Gamma_2)\vdash|e_1|\,|e_2|:\tau_2$

Goal follows from T-Weaken on (7)

- **Case 13:** *AT-TensorI.*

  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \langle\!\langle e_1, e_2 \rangle\!\rangle \downarrow \tau_1 \otimes \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2$

    (2) $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$

    (3) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \downarrow \tau_1 \Rightarrow \Phi_1, \Gamma_1$

    (4) $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_2 \downarrow \tau_2 \Rightarrow \Phi_2, \Gamma_2$

  - **Goal:**

    $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \smallsetminus \Gamma_2 \vdash \langle\!\langle |e_1|, |e_2| \rangle\!\rangle : \tau_1 \otimes \tau_2}$

  By IH on (3)

  (5) $\Psi;\Theta;\Delta;\Omega;\Gamma \smallsetminus \Gamma_1 \vdash |e_1| : \tau_1$

  By IH on (4)

  (6) $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \smallsetminus \Gamma_2 \vdash |e_2| : \tau_2$

  By T-TensorI on (5) and (6)

  (7) $\Psi;\Theta;\Delta;\Omega;(\Gamma \smallsetminus \Gamma_1),(\Gamma_1 \smallsetminus \Gamma_2) \vdash \langle\!\langle |e_1|, |e_2| \rangle\!\rangle : \tau_1 \otimes \tau_2$

  Goal follows from T-Weaken on (7)

- **Case 14:** *AT-TensorE.*

  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{let}\ \ \langle\!\langle x, y \rangle\!\rangle = e\ \ \texttt{in}\ \ e' \downarrow \tau' \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2 \smallsetminus \{x : \tau_1, y : \tau_2\}$

    (2) $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$

    (3) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \tau_1 \otimes \tau_2 \Rightarrow \Phi_1, \Gamma_1$

    (4) $\Psi;\Theta;\Delta;\Omega;\Gamma_1, x : \tau_1, y : \tau_2 \vdash e' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2$

  - **Goal:**

    $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \smallsetminus (\Gamma_2 \smallsetminus \{x : \tau_1, y : \tau_2\}) \vdash \texttt{let}\ \ \langle\!\langle x, y \rangle\!\rangle = |e|\ \ \texttt{in}\ \ |e'| : \tau'}$

  By IH on (3)

  (5) $\Psi;\Theta;\Delta;\Omega;\Gamma \smallsetminus \Gamma_1 \vdash |e| : \tau_1 \otimes \tau_2$

  By IH on (4)

  (6) $\Psi;\Theta;\Delta;\Omega;(\Gamma_1, x : \tau_1, y : \tau_2) \smallsetminus \Gamma_2 \vdash |e'| : \tau'$

  By T-Weaken on (6)

(7) $\Psi;\Theta;\Delta;\Omega;(\Gamma_1 \setminus (\Gamma_2 \setminus \{x:\tau_1, y:\tau_2\})), x:\tau_1, y:\tau_2 \vdash |e'| : \tau'$

Goal follows by T-TensorE and T-Weaken on (5) and (7)

▸ **Case 15:** *AT-WithI.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash (e_1, e_2) \downarrow \tau_1 \& \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_1 \cap \Gamma_2$

(2) $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \downarrow \tau_1 \Rightarrow \Phi_1, \Gamma_1$

(4) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_2 \downarrow \tau_2 \Rightarrow \Phi_2, \Gamma_2$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \setminus (\Gamma_1 \cap \Gamma_2) \vdash (|e_1|, |e_2|) : \tau_1 \& \tau_2}$

By IH on (3)

(5) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma_1 \vdash |e_1| : \tau_1$

By IH on (4)

(6) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma_2 \vdash |e_2| : \tau_2$

By T-Weaken on (5) and (6)

(7) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus (\Gamma_1 \cap \Gamma_2) \vdash |e_1| : \tau_1$

(8) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus (\Gamma_1 \cap \Gamma_2) \vdash |e_2| : \tau_2$

Goal follows by T-WithI on (7) and (8)

▸ **Case 16:** *AT-Fst.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \mathtt{fst}(e) \uparrow \tau_1 \Rightarrow \Phi, \Gamma'$

(2) $\Theta;\Delta \vDash \Phi$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \tau_1 \& \tau_2 \Rightarrow \Phi, \Gamma'$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma' \vdash \mathtt{fst}(|e|) : \tau_1}$

By IH on (3)

(4) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma' \vdash |e| : \tau_1 \& \tau_2$

Goal follows by T-Fst on (4)

▸ **Case 17:** *AT-Snd.*

Identical to Case 16.

- **Case 18:** *AT-Inl.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{inl}(e) \downarrow \tau_1 \oplus \tau_2 \Rightarrow \Phi, \Gamma'$

    (2) $\Theta; \Delta \vDash \Phi$

    (3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau_1 \Rightarrow \Phi, \Gamma'$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash \texttt{inl}(|e|) : \tau_1 \oplus \tau_2}$

  By IH on (3)

  (4) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : \tau_1$

  Goal follows by T-Inl on (4)

- **Case 19:** *AT-Inr.*

  Identical to Case 18.

- **Case 20:** *AT-Case.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{case}(e, x.e_1, y.e_2) \downarrow \tau \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3, \Gamma'$

    (2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2 \wedge \Phi_3$

    (3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \tau_1 \oplus \tau_2 \Rightarrow \Phi_1, \Gamma_1$

    (4) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, x : \tau_1 \vdash e_1 \downarrow \tau \Rightarrow \Phi_2, \Gamma_2$

    (5) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, y : \tau_2 \vdash e_2 \downarrow \tau \Rightarrow \Phi_3, \Gamma_3$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash \texttt{case}(|e|, x.|e_1|, y.|e_2|) : \tau}$

  By IH on (3)

  (6) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma_1 \vdash |e| : \tau_1 \oplus \tau_2$

  By IH on (4)

  (7) $\Psi; \Theta; \Delta; \Omega; (\Gamma_1, x : \tau_1) \setminus \Gamma_2 \vdash |e_1| : \tau$

  By IH on (5)

  (8) $\Psi; \Theta; \Delta; \Omega; (\Gamma_1, y : \tau_2) \setminus \Gamma_2 \vdash |e_2| : \tau$

  By T-Weaken on (7) and then (8)

(9) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \setminus \Gamma', x : \tau_1 \vdash |e_1| : \tau$

(10) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \setminus \Gamma', y : \tau_2 \vdash |e_2| : \tau$

By T-Case on (6), (9), and (10)

(11) $\Psi; \Theta; \Delta; \Omega; (\Gamma \setminus \Gamma_1), (\Gamma_1 \setminus \Gamma') \vdash \texttt{case}(|e|, x.|e_1|, y.|e_2|) : \tau$

Goal follows by T-Weaken on (11)

▸ **Case 21:** *AT-ExpI.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash !e \downarrow !\tau \Rightarrow \Phi, \Gamma$

(2) $\Theta; \Delta \vDash \Phi$

(3) $\Psi; \Theta; \Delta; \Omega; \cdot \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \cdot \vdash !|e| : !\tau}$

By IH on (3)

(4) $\Psi; \Theta; \Delta; \Omega; \cdot \vdash |e| : \tau$

Goal follows by T-ExpI on (4)

▸ **Case 22:** *AT-ExpE.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{let}\ !x = e\ \texttt{in}\ e' \downarrow \tau' \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2$

(2) $\Theta; \Delta \vDash \Phi_1 \Phi_2$

(3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow !\tau \Rightarrow \Phi_1, \Gamma_1$

(4) $\Psi; \Theta; \Delta; \Omega, x : \tau; \Gamma_1 \vdash e' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma_2 \vdash \texttt{let}\ !x = |e|\ \texttt{in}\ |e'| : \tau'}$

By IH on (3)

(5) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma_1 \vdash |e| : !\tau$

By IH on (4)

(6) $\Psi; \Theta; \Delta; \Omega, x : \tau; \Gamma_1 \setminus \Gamma_2 \vdash |e'| : \tau'$

By T-ExpE on (5) and (6)

(7) $\Psi; \Theta; \Delta; \Omega; (\Gamma \setminus \Gamma_1), (\Gamma_1 \setminus \Gamma_2) \vdash \texttt{let}\ !x = |e|\ \texttt{in}\ |e'| : \tau'$

Goal follows by T-Weaken on (7)

▸ **Case 23:** *AT-TAbs.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda\alpha.e \downarrow \forall\alpha : K.\tau \Rightarrow \Phi, \Gamma'$

(2) $\Theta; \Delta \vDash \Phi$

(3) $\Psi, \alpha : K; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash \Lambda\alpha.|e| : \forall\alpha : K.\tau}$

By IH on (3)

(4) $\Psi, \alpha : K; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : \tau$

Goal follows by T-TAbs on (4)

▸ **Case 24:** *AT-TApp.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e[\tau'] \uparrow \tau[\tau'/\alpha] \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma'$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \forall\alpha : K.\tau \Rightarrow \Phi_1, \Gamma'$

(4) $\Psi; \Theta; \Delta \vdash \tau' : K \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e|[\tau'] : \tau[\tau'/\alpha]}$

By IH on (3)

(5) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : \forall\alpha : K.\tau$

By Theorem A.19 on (5)

(6) $\Psi; \Theta; \Delta \vdash \tau' : K$

Goal Follows by T-TApp on (5) and (6)

▸ **Case 25:** *AT-TApp.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda i.e \downarrow \forall i : S.\tau \Rightarrow \forall i : S.\Phi, \Gamma'$

(2) $\Theta; \Delta \vDash \forall i : S.\Phi$

(3) $\Psi; \Theta, i : S; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash \Lambda i.|e| : \forall i : S.\tau}$$

Equivalently to (2)

(4) $\Theta, i : S; \Delta \vDash \Phi$

By IH on (3)

(5) $\Psi; \Theta, i : S; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : \tau$

Goal follows by T-IAbs on (5)

▸ **Case 26:** *AT-IApp.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e[I] \uparrow \tau[I/i] \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma'$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \forall i : S.\tau \Rightarrow \Phi_1, \Gamma'$

(4) $\Theta; \Delta \vdash I : S \Rightarrow \Phi_2$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e[I]| : \tau[I/i]}$$

By IH on (3)

(5) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : \forall i : S.\tau$

By Theorem A.17 on (4)

(6) $\Theta; \Delta \vdash I : S$

Goal follows by T-IApp on (5) and (6)

▸ **Case 27:** *AT-Fix.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{fix}\ x.e \downarrow \tau \Rightarrow \Phi, \Gamma$

(2) $\Theta; \Delta \vDash \Phi$

(3) $\Psi; \Theta; \Delta; \Omega, x : \tau; \cdot \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta; \Omega; \cdot \vdash \texttt{fix}\ x.|e| : \tau}$$

By IH on (3)

(4) $\Psi; \Theta; \Delta; \Omega, x : \tau; \cdot \vdash |e| : \tau$

Goal follows by T-Fix on (4)

▸ **Case 28:** *AT-Ret.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{ret}\ \ e \downarrow \mathbb{M}\,(I,\vec{p})\,\tau \Rightarrow \Phi,\Gamma'$

(2) $\Theta;\Delta \vDash \Phi$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi,\Gamma'$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma' \vdash \texttt{ret}\ \ |e| : \mathbb{M}\,(I,\vec{p})\,\tau}$

By IH on (3)

(4) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma' \vdash |e| : \tau$

By T-Ret on (4)

(5) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma' \vdash \texttt{ret}(|e|) : \mathbb{M}\,(I,\vec{0})\,\tau$

By S-Monad and S-Refl using the fact that $\Theta;\Delta \vDash \vec{p} \geq \vec{0}$

(6) $\Psi;\Theta;\Delta \vdash \mathbb{M}\,(I,\vec{0})\,\tau <: \mathbb{M}\,(I,\vec{p})\,\tau : \star$

Goal follows by T-Sub on (5) and (6)

▸ **Case 29:** *AT-Bind.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{bind}\ \ x = e_1\ \ \texttt{in}\ \ e_2 \downarrow \mathbb{M}\,(I,\vec{q})\,\tau_2 \Rightarrow \Phi,\Gamma_2 \setminus \{x:\tau_1\}$

(2) $\Theta;\Delta \vDash \Phi$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \uparrow \mathbb{M}\,(J,\vec{p})\,\tau_1 \Rightarrow \Phi_1,\Gamma_1$

(4) $\Psi;\Theta;\Delta;\Omega;\Gamma_1, x:\tau_1 \vdash e_2 \downarrow \mathbb{M}\,(I,\vec{q}-\vec{p})\,\tau_2 \Rightarrow \Phi_2,\Gamma_2$

(5) $\Phi = (\vec{q} \geq \vec{p}) \wedge (I = J) \wedge \Phi_1 \wedge \Phi_2$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \setminus (\Gamma_2 \setminus \{x:\tau_1\}) \vdash \texttt{bind}\ \ x = |e_1|\ \ \texttt{in}\ \ |e_2| : \mathbb{M}\,(I,\vec{q})\,\tau_2}$

By IH on (3)

(6) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma_1 \vdash |e_1| : \mathbb{M}\,(J,\vec{p})\,\tau_1$

By IH on (4)

(7) $\Psi;\Theta;\Delta;\Omega;(\Gamma_1, x:\tau_1) \setminus \Gamma_2 \vdash |e_2| : \mathbb{M}\,(I,\vec{q}-\vec{p})\,\tau_2$

From (2) and (5)

(8) $\Theta; \Delta \vDash I = J$

By S-Monad and S-Refl, using (8)

(9) $\Psi; \Theta; \Delta \vdash M(J, \vec{p})\,\tau_1 <: M(I, \vec{p})\,\tau_1 \;: \star$

By T-Sub on (6) and (9)

(10) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma_1 \vdash |e_1| : \mathbb{M}\,(I, \vec{p})\,\tau_1$

By T-Weaken on (7)

(11) $\Psi; \Theta; \Delta; \Omega; (\Gamma_1 \setminus (\Gamma_2 \setminus \{x\})), x : \tau_1 \vdash |e_2| : \mathbb{M}\,(I, \vec{q} - \vec{p})\,\tau_2$

By T-Bind on (10) and (11)

(12) $\Psi; \Theta; \Delta; \Omega(\Gamma \setminus \Gamma_1), (\Gamma_1 \setminus (\Gamma_2 \setminus \{x\})) \vdash \texttt{bind}\;\; x = |e_1| \;\; \texttt{in} \;\; |e_2| : \mathbb{M}\,(I, \vec{q} - \vec{p} + \vec{p})\,\tau_2$

By S-Monad and S-Refl, using the fact that $\Theta; \Delta \vDash (\vec{q} - \vec{p}) + \vec{p} = \vec{q}$

(13) $\Psi; \Theta; \Delta \vdash \mathbb{M}\,(I, \vec{q} - \vec{p} + \vec{p})\,\tau_2 <: \mathbb{M}\,(I, \vec{q})\,\tau_2 : \star$

By T-Sub on (12) and (13)

(13) $\Psi; \Theta; \Delta; \Omega(\Gamma \setminus \Gamma_1), (\Gamma_1 \setminus (\Gamma_2 \setminus \{x\})) \vdash \texttt{bind}\;\; x = |e_1| \;\; \texttt{in} \;\; |e_2| : \mathbb{M}\,(I, \vec{q})\,\tau_2$

Goal follows by T-Weaken on (13)

▸ **Case 30:** *AT-Tick.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{tick}[I|\vec{p}] \uparrow \mathbb{M}\,(I, \vec{p})\,1 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi_1$

(4) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \Rightarrow \Phi_1$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \cdot \vdash \texttt{tick}[I|\vec{p}] : \mathbb{M}\,(I, \vec{p})\,1}$

By Theorem A.17 on (3)

(5) $\Theta; \Delta \vdash I : \mathbb{N}$

By Theorem A.17 on (4)

(6) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+}$

Goal Follows by T-Tick on (5) and (6)

▸ **Case 31:** *AT-Release.*

- **Given:**

  (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{release}\ \ x = e_1\ \ \texttt{in}\ \ e_2 \downarrow \mathbb{M}\,(I,\vec{p})\,\tau_2 \Rightarrow (I = J \wedge \Phi_1 \wedge \Phi_2), \Gamma_2 \setminus \{x\}$

  (2) $\Theta;\Delta \vDash (I = J) \wedge \Phi_1 \wedge \Phi_2$

  (3) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \uparrow [J|\vec{q}]\tau_1 \Rightarrow \Phi_1, \Gamma_1$

  (4) $\Psi;\Theta;\Delta;\Omega;\Gamma_1, x : \tau_1 \vdash e_2 \downarrow \mathbb{M}\,(I, \vec{p} + \vec{q})\,\tau_2 \Rightarrow \Phi_2, \Gamma_2$

- **Goal:**

  $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \setminus (\Gamma_2 \setminus \{x : \tau_1\}) \vdash \texttt{release}\ \ x = |e_1|\ \ \texttt{in}\ \ |e_2| : \mathbb{M}\,(I,\vec{p})\,\tau_2}$

By IH on (3)

(5) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma_1 \vdash |e_1| : [J|\vec{q}]\tau_1$

By IH on (4)

(6) $\Psi;\Theta;\Delta;\Omega;(\Gamma_1, x : \tau_1) \setminus \Gamma_2 \vdash |e_2| : \mathbb{M}\,(I, \vec{p} + \vec{q})\,\tau_2$

From (2)

(7) $\Theta;\Delta \vDash I = J$

By S-Pot, S-Refl, and (7)

(8) $\Psi;\Theta;\Delta \vdash [J|\vec{q}]\tau_1 <: [I|\vec{q}]\tau_1 : \star$

By T-Sub on (5) and (8)

(9) $\Psi;\Theta;\Delta;\Omega;\Gamma \setminus \Gamma_1 \vdash |e_1| : [I|\vec{q}]\tau_1$

By T-Weaken on (6)

(10) $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \setminus (\Gamma_2 \setminus x : \tau_1), x : \tau_1 \vdash |e_2| : \mathbb{M}\,(I, \vec{p} + \vec{q})\,\tau_2$

By T-Release on (9) and (10)

(11) $\Psi;\Theta;\Delta;\Omega;(\Gamma \setminus \Gamma_1), (\Gamma_1 \setminus (\Gamma_2 \setminus x : \tau_1)) \vdash \texttt{release}\ \ x = |e_1|\ \ \texttt{in}\ \ |e_2| : \mathbb{M}\,(I,\vec{p})\,\tau_2$

Goal follows by another T-Weaken on (11)

- **Case 32:** *AT-Store.*

  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{store}[K|\vec{w}](e) \downarrow \mathbb{M}\,(I,\vec{q})\,([J|\vec{p}]\,\tau) \Rightarrow \Phi, \Gamma'$

    (2) $\Theta;\Delta \vDash \Phi$

    (3) $\Theta;\Delta \vdash K : \mathbb{N} \Rightarrow \Phi_1$

    (4) $\Theta;\Delta \vdash \vec{w} : \vec{\mathbb{R}^+} \Rightarrow \Phi_2$

    (5) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi_3, \Gamma'$

(6) $\Phi = \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge (\vec{p} \leq \vec{w} \leq \vec{q}) \wedge (I = J = K)$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash \texttt{store}[K|\vec{w}](|e|) : \mathbb{M}\,(I, \vec{q})\,([J|\vec{p}]\,\tau)}$

By Theorem A.17 on (3)

(7) $\Theta; \Delta \vdash K : \mathbb{N}$

By Theorem A.17 on (4)

(8) $\Theta; \Delta \vdash \vec{w} : \vec{\mathbb{R}^+}$

By IH on (5)

(9) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : \tau$

By T-Store on (7), (8), (9)

(10) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash \texttt{store}[K|\vec{w}](|e|) : \mathbb{M}\,(K, \vec{w})\,([K|\vec{w}]\,\tau)$

By S-Monad, S-Pot, and S-Refl using (2), (6)

(11) $\Psi; \Theta; \Delta \vdash \mathbb{M}\,(K, \vec{w})\,([K|\vec{w}]\,\tau) <: \mathbb{M}\,(I, \vec{q})\,([J|\vec{p}]\,\tau) : \star$

Goal follows by T-Sub on (10) and (11)

▸ **Case 33:** *AT-StoreConst.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{store}[J](e) \downarrow \mathbb{M}\,(K, \vec{p})\,([I]\,\tau) \Rightarrow \Phi, \Gamma'$

(2) $\Theta; \Delta \models \Phi$

(3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi_1, \Gamma'$

(4) $\Theta; \Delta \vdash J \downarrow \mathbb{R} \Rightarrow \Phi_2$

(5) $\Phi = (\texttt{const}(I) \leq \texttt{const}(J) \leq \vec{p}) \wedge \Phi_1 \wedge \Phi_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash \texttt{store}[J](|e|) : \mathbb{M}\,(K, \vec{p})\,([I]\,\tau)}$

By IH on (3)

(6) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : \tau$

By Theorem A.17 on (4)

(7) $\Theta; \Delta \vdash J : \mathbb{R}$

By T-StoreConst on (6) and (7)

(8) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash \texttt{store}[J](|e|) : \mathbb{M}\,(J, \texttt{const}(J))\,([J]\,\tau)$

By (2) and (5)

(9) $\Theta; \Delta \vDash \texttt{const}(I) \leq \texttt{const}(J) \leq \vec{p}$

By S-Monad, S-Pot, S-Refl, and (9)

(10) $\Psi; \Theta; \Delta \vdash \mathbb{M}\,(J, \texttt{const}(J))\,([J]\,\tau) <: \mathbb{M}\,(K, \vec{p})\,([I]\,\tau)$

Goal follows by T-Sub on (8) and (10)

▸ **Case 34:** *AT-ReleaseConst.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{release}\ \ x = e_1\ \ \texttt{in}\ \ e_2 \downarrow \mathbb{M}\,(I, \vec{p})\,\tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2 \setminus \{x\}$

(2) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

(3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 \uparrow [J]\tau_1 \Rightarrow \Phi_1, \Gamma_1$

(4) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, x : \tau_1 \vdash e_2 \downarrow \mathbb{M}\,(I, \vec{p} + \texttt{const}(J))\,\tau_2 \Rightarrow \Phi_2, \Gamma_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus (\Gamma_2 \setminus \{x\}) \vdash \texttt{release}\ \ x = |e_1|\ \ \texttt{in}\ \ |e_2| : \mathbb{M}\,(I, \vec{p})\,\tau_2}$

By IH on (3)

(5) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma_1 \vdash |e_1| : [J]\tau_1$

By IH on (4)

(6) $\Psi; \Theta; \Delta; \Omega; (\Gamma_1, x : \tau_1) \setminus \Gamma_2 \vdash |e_2| : \mathbb{M}\,(I, \vec{p} + \texttt{const}(J))\,\tau_2$

By T-Weaken on (6)

(7) $\Psi; \Theta; \Delta; \Omega; (\Gamma_1 \setminus (\Gamma_2 \setminus \{x : \tau_1\})), x : \tau_1 \vdash |e_2| : \mathbb{M}\,(I, \vec{p} + \texttt{const}(J))\,\tau_2$

By T-ReleaseConst on (5) and (7)

(8) $\Psi; \Theta; \Delta; \Omega; (\Gamma \setminus \Gamma_1), (\Gamma_1 \setminus (\Gamma_2 \setminus \{x\})) \vdash \texttt{release}\ \ x = |e_1|\ \ \texttt{in}\ \ |e_2| : \mathbb{M}\,(I, \vec{p})\,\tau_2$

Goal follows by T-Weaken on (8)

▸ **Case 35:** *AT-Shift.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{shift}(e) \downarrow M\,(I, \vec{q})\,\tau \Rightarrow (I \geq 1) \wedge \Phi, \Gamma'$

(2) $\Theta; \Delta \vDash (I \geq 1) \wedge \Phi$

(3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow M\,(I - 1, \triangleleft\, \vec{q})\,\tau \Rightarrow \Phi, \Gamma'$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash \texttt{shift}(|e|) : M\,(I, \vec{q})\,\tau}$

By IH on (3)

(4) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : M\,(I - 1, \triangleleft\, \vec{q})\,\tau$

From (2)

(5) $\Theta; \Delta \vDash I \geq 1$

Goal follows by T-Shift on (4) and (5)

- **Case 36:** *AT-CImpI.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda.e \downarrow (\Phi' \Rightarrow \tau) \Rightarrow (\Phi' \to \Phi), \Gamma'$

    (2) $\Theta; \Delta \vDash \Phi' \to \Phi$

    (3) $\Psi; \Theta; \Delta, \Phi'; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash \Lambda.|e| : \Phi' \Rightarrow \tau}$

  Equivalently to (2)

  (4) $\Theta; \Delta, \Phi' \vDash \Phi$

  By IH on (3)

  (5) $\Psi; \Theta; \Delta, \Phi'; \Omega; \Gamma \setminus \Gamma' \vdash |e| : \tau$

  Goal is immediate by T-CImpI on (5)

- **Case 37:** *AT-CImpE.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e\{\} \uparrow \tau \Rightarrow \Phi \wedge \Phi', \Gamma'$

    (2) $\Theta; \Delta \vDash \Phi \wedge \Phi'$

    (3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow (\Phi' \Rightarrow \tau) \Rightarrow \Phi, \Gamma'$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e|\{\} : \tau}$

  By IH on (3)

  (4) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : (\Phi' \Rightarrow \tau)$

  From (2)

  (5) $\Theta; \Delta \vDash \Phi'$

  Goal follows from T-CImpI on (4) and (5)

- **Case 38:** *AT-CAndI.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash < e > \downarrow \Phi' \& \tau \Rightarrow \Phi \wedge \Phi', \Gamma'$

    (2) $\Theta; \Delta \vDash \Phi \wedge \Phi'$

    (3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma'$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash < |e| >: \Phi' \& \tau}$

  By IH on (3)

  (4) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma' \vdash |e| : \tau$

  From (2)

  (5) $\Theta; \Delta \vDash \Phi'$

  Goal follows from T-CAndI on (4) and (5)

- **Case 39:** *AT-CAndE.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{clet}\ x = e\ \texttt{in}\ e' \downarrow \tau' \Rightarrow \Phi_1 \wedge (\Phi' \to \Phi_2), \Gamma_2 \setminus \{x : \tau\}$

    (2) $\Theta; \Delta \vDash \Phi_1 \wedge (\Phi' \to \Phi_2)$

    (3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \Phi' \& \tau \Rightarrow \Phi_1, \Gamma_1$

    (4) $\Psi; \Theta; \Delta, \Phi'; \Omega; \Gamma_1, x : \tau \vdash e' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \setminus (\Gamma_2 \setminus \{x : \tau\}) \vdash \texttt{clet}\ x = |e|\ \texttt{in}\ |e'| : \tau'}$

  By IH on (3)

  (5) $\Psi; \Theta; \Delta; \Omega; \Gamma \setminus \Gamma_1 \vdash |e| : \Phi' \& \tau$

  From (2)

  (6) $\Theta; \Delta, \Phi' \vDash \Phi_2$

  By IH on (4) using (6)

  (7) $\Psi; \Theta; \Delta, \Phi'; \Omega; (\Gamma_1, x : \tau) \setminus \Gamma_2 \vdash |e'| : \tau'$

  By T-Weaken on (7)

  (8) $\Psi; \Theta; \Delta, \Phi'; \Omega; (\Gamma_1 \setminus (\Gamma_2 \setminus \{x : \tau\})), x : \tau \vdash |e'| : \tau'$

  By T-CAndE on (5) and (8)

(9) $\Psi;\Theta;\Delta;\Omega;(\Gamma\setminus\Gamma_1),(\Gamma_1\setminus(\Gamma_2\setminus\{x:\tau\}))\vdash \texttt{clet}\ x=|e|\ \texttt{in}\ |e'|:\tau'$

Goal follows by T-Weaken on (9)

▸ **Case 40:** *AT-Sub.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma\vdash e\downarrow\tau\Rightarrow\Phi_1\wedge\Phi_2,\Gamma'$

(2) $\Theta;\Delta\vDash\Phi_1\wedge\Phi_2$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma\vdash e\uparrow\tau'\Rightarrow\Phi_1,\Gamma'$

(4) $\Psi;\Theta;\Delta\vdash\tau'<:\tau:\star\Rightarrow\Phi_2$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma\setminus\Gamma'\vdash|e|:\tau}$

By IH on (3)

(5) $\Psi;\Theta;\Delta;\Omega;\Gamma\setminus\Gamma'\vdash|e|:\tau'$

By Theorem 8.6 on (4)

(6) $\Psi;\Theta;\Delta\vdash\tau'<:\tau:\star$

Goal follows by T-Sub on (5) and (6)

▸ **Case 41:** *AT-Anno.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma\vdash(e:\tau)\uparrow\tau\Rightarrow\Phi,\Gamma'$

(2) $\Theta;\Delta\vDash\Phi$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma\vdash e\downarrow\tau\Rightarrow\Phi,\Gamma'$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma\setminus\Gamma'\setminus\vdash|(e:\tau)|:\tau}$

By IH on (3)

(4) $\Psi;\Theta;\Delta;\Omega;\Gamma\setminus\Gamma'\vdash|e|:\tau$

Goal follows immediately by (4) since $|(e:\tau)|=|e|$

$\square$

Proof. By induction on $\Theta;\Delta\vdash I:S$.

(I-Var) Immediate by AI-Var with $\Phi=\top$.

(I-Plus) Suppose $\Theta;\Delta \vdash I + J : bS$ from $\Theta;\Delta \vdash I : bS$ and $\Theta;\Delta \vdash J : bS$. By IH, $\Theta;\Delta \vdash I : bS \Rightarrow \Phi_1$ and $\Theta;\Delta \vdash J : bS \Rightarrow \Phi_2$ with $\Theta;\Delta \vDash \Phi_1$ and $\Theta;\Delta \vDash \Phi_2$. By AI-Plus, $\Theta;\Delta \vdash I + J : bS \Rightarrow \Phi_1 \wedge \Phi_2$

(I-Minus) Suppose $\Theta;\Delta \vdash I - J : bS$ from $\Theta;\Delta \vdash I : bS$ and $\Theta;\Delta \vdash J : bS$, and $\Theta;\Delta \vDash I \geq J$. By IH, $\Theta;\Delta \vdash I : bS \Rightarrow \Phi_1$ and $\Theta;\Delta \vdash J : bS \Rightarrow \Phi_2$ with $\Theta;\Delta \vDash \Phi_1$ and $\Theta;\Delta \vDash \Phi_2$. By AI-Minus, $\Theta;\Delta \vdash I - J : bS \Rightarrow \Phi_1 \wedge \Phi_2 \wedge (I \geq J)$

(I-Times-$\mathbb{R}$)

(I-Times-$\vec{\mathbb{R}}$)

(I-Times-$\mathbb{N}$)

(I-Shift) Suppose $\Theta;\Delta \vdash \triangleleft I : \vec{\mathbb{R}}^{+}$ from $\Theta;\Delta \vdash I : \vec{\mathbb{R}}^{+}$. By IH, $\Theta;\Delta \vdash I : \vec{\mathbb{R}}^{+} \Rightarrow \Phi$, and $\Theta;\Delta \vDash \Phi$. By AI-Shift, $\Theta;\Delta \vdash \triangleleft I : \vec{\mathbb{R}}^{+} \Rightarrow \Phi$, as required.

(I-Lam) Suppose $\Theta;\Delta \vdash \lambda i : bS.I : bS \to S$ from $\Theta, i : bS;\Delta \vdash I : S$. By IH, $\Theta, i : bS;\Delta \vdash I : S \Rightarrow \Phi$ with $\Theta;\Delta \vDash \Phi$. By AI-Lam, $\Theta;\Delta \vdash \lambda i : bS.I : bS \to S \Rightarrow \Phi$.

(I-App)

(I-Sum)

□

Proof of Theorem A.27

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash \tau : K$

▸ **Goal:**

$\boxed{\Phi;\Theta;\Delta \vdash \tau : K \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

▸ **Case 1:** *K-Var.*

Immediate.

▸ **Case 2:** *K-Unit.*

Immediate.

▸ **Case 3:** *K-Arr.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash \tau_1 \multimap \tau_2 : \star$

(2) $\Psi; \Theta; \Delta \vdash \tau_1 : \star$

(3) $\Psi; \Theta; \Delta \vdash \tau_2 : \star$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash \tau_1 \multimap \tau_2 : \star \Rightarrow \Phi' \text{ and } \Theta; \Delta \vDash \Phi'}$$

By IH on (2) and (3)

(4) $\Psi; \Theta; \Delta \vdash \tau_1 : \star \Rightarrow \Phi_1$

(5) $\Psi; \Theta; \Delta \vdash \tau_2 : \star \Rightarrow \Phi_2$

(6) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

By AK-Arr on (4) and (5)

(7) $\Psi; \Theta; \Delta \vdash \tau_1 \multimap \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

The Goal follows by (6) and (7), with $\Phi' = \Phi_1 \wedge \Phi_2$

▸ **Case 4:** *K-Tensor.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 : \star$

(2) $\Psi; \Theta; \Delta \vdash \tau_1 : \star$

(3) $\Psi; \Theta; \Delta \vdash \tau_2 : \star$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 : \star \Rightarrow \Phi' \text{ and } \Theta; \Delta \vDash \Phi'}$$

By IH on (2) and (3)

(4) $\Psi; \Theta; \Delta \vdash \tau_1 : \star \Rightarrow \Phi_1$

(5) $\Psi; \Theta; \Delta \vdash \tau_2 : \star \Rightarrow \Phi_2$

(6) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

By AK-Tensor on (4) and (5)

(7) $\Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

The Goal follows by (6) and (7), with $\Phi' = \Phi_1 \wedge \Phi_2$

▸ **Case 5:** *K-With.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1 \& \tau_2 : \star$

(2) $\Psi; \Theta; \Delta \vdash \tau_1 : \star$

(3) $\Psi; \Theta; \Delta \vdash \tau_2 : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \tau_1 \& \tau_2 : \star \Rightarrow \Phi' \text{ and } \Theta; \Delta \vDash \Phi'}$

By IH on (2) and (3)

(4) $\Psi; \Theta; \Delta \vdash \tau_1 : \star \Rightarrow \Phi_1$

(5) $\Psi; \Theta; \Delta \vdash \tau_2 : \star \Rightarrow \Phi_2$

(6) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

By K-With on (4) and (5)

(7) $\Psi; \Theta; \Delta \vdash \tau_1 \& \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

The Goal follows by (6) and (7), with $\Phi' = \Phi_1 \wedge \Phi_2$

▸ **Case 6:** *K-Sum.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau_1 \oplus \tau_2 : \star$

(2) $\Psi; \Theta; \Delta \vdash \tau_1 : \star$

(3) $\Psi; \Theta; \Delta \vdash \tau_2 : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \tau_1 \oplus \tau_2 : \star \Rightarrow \Phi' \text{ and } \Theta; \Delta \vDash \Phi'}$

By IH on (2) and (3)

(4) $\Psi; \Theta; \Delta \vdash \tau_1 : \star \Rightarrow \Phi_1$

(5) $\Psi; \Theta; \Delta \vdash \tau_2 : \star \Rightarrow \Phi_2$

(6) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

By AK-Sum on (4) and (5)

(7) $\Psi; \Theta; \Delta \vdash \tau_1 \oplus \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

The Goal follows by (6) and (7), with $\Phi' = \Phi_1 \wedge \Phi_2$

▸ **Case 7:** *K-Bang.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash !\tau : \star$ (2) $\Psi; \Theta; \Delta \vdash \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash !\tau : \star \Rightarrow \Phi' \text{ and } \Theta; \Delta \vDash \Phi'}$

By IH on (2)

(3) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi'$

(4) $\Theta; \Delta \vDash \Phi'$

By AK-Bang on (3)

(4) $\Psi; \Theta; \Delta \vdash !\tau : \star \Rightarrow \Phi'$

▸ **Case 8:** *K-IForall.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \forall i : S.\tau : \star$

(2) $\Psi; \Theta, i : S; \Delta \vdash \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \forall i : S.\tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (2)

(3) $\Psi; \Theta, i : S; \Delta \vdash \tau : \star \Rightarrow \Phi$

(4) $\Theta, i : S; \Delta \vDash \Phi$

Equivalently to (4)

(5) $\Theta; \Delta \vDash \forall i : S.\Phi$

By AK-IForall on (3)

(6) $\Psi; \Theta; \Delta \vdash \forall i : S.\tau : \star \Rightarrow \forall i : S.\Phi$

▸ **Case 9:** *K-IExists.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \exists i : S.\tau : \star$

(2) $\Psi; \Theta, i : S; \Delta \vdash \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \exists i : S.\tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (2)

(3) $\Psi; \Theta, i : S; \Delta \vdash \tau : \star \Rightarrow \Phi$

(4) $\Theta, i : S; \Delta \vDash \Phi$

Equivalently to (4)

(5) $\Theta; \Delta \vDash \forall i : S.\Phi$

By AK-IExists on (3)

(6) $\Psi; \Theta; \Delta \vdash \exists i : S.\tau : \star \Rightarrow \forall i : S.\Phi$

- **Case 10:** *K-List.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash L^I \tau : \star$

    (2) $\Theta; \Delta \vdash I : \mathbb{N}$

    (3) $\Psi; \Theta; \Delta \vdash \tau : \star$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash L^I \tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

  By IH on (3)

  (4) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_1$

  (5) $\Theta; \Delta \vDash \Phi_1$

  By Theorem A.25 on (2)

  (6) $\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi_2$

  (7) $\Theta; \Delta \vDash \Phi_2$

  By AK-List on (4) and (6)

  (8) $\Psi; \Theta; \Delta \vdash L^I \tau : \star \Rightarrow \Phi_1 \wedge \Phi_2$

  The goal follows from (5), (7), (8) with $\Phi = \Phi_1 \wedge \Phi_2$

- **Case 11:** *K-Conj.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash \Phi \& \tau : \star$

    (2) $\Theta; \Delta \vdash \Phi \; \texttt{wf}$

    (3) $\Psi; \Theta; \Delta \vdash \tau : \star$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash \Phi \& \tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

  By Theorem A.26 on (2)

  (4) $\Theta; \Delta \vdash \Phi \; \texttt{wf} \Rightarrow \Phi_1$

  (5) $\Theta; \Delta \vDash \Phi_!$

  By IH on (3)

(6) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_2$

(7) $\Theta; \Delta \vDash \Phi_2$

By AK-Conj on (4) and (6)

(8) $\Psi; \Theta; \Delta \vdash \Phi \& \tau : \star \Rightarrow \Phi_1 \land \Phi_2$

The Goal follows by (5), (7), (8)

▸ **Case 12:** *K-Impl.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \Phi \implies \tau : \star$

(2) $\Theta; \Delta \vdash \Phi$ `wf`

(3) $\Psi; \Theta; \Delta \vdash \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \Phi \implies \tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By Theorem A.26 on (2)

(4) $\Theta; \Delta \vdash \Phi\ \texttt{wf} \Rightarrow \Phi_1$

(5) $\Theta; \Delta \vDash \Phi_!$

By IH on (3)

(6) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_2$

(7) $\Theta; \Delta \vDash \Phi_2$

By AK-Impl on (4) and (6)

(8) $\Psi; \Theta; \Delta \vdash \Phi \implies \tau : \star \Rightarrow \Phi_1 \land \Phi_2$

The Goal follows by (5), (7), (8)

▸ **Case 13:** *K-Monad.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \mathbb{M}(I, \vec{p})\tau : \star$

(2) $\Theta; \Delta \vdash I : \mathbb{N}$

(3) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+}$

(4) $\Psi; \Theta; \Delta \vdash \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \mathbb{M}(I, \vec{p})\tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By Theorem A.25 on (2) and (3)

(5) $\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi_1$

(6) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \Rightarrow \Phi_2$

(7) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

By IH on (4)

(8) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_3$

(9) $\Theta; \Delta \vDash \Phi_3$

By AK-Monad on (5), (6), (8)

(10) $\Psi; \Theta; \Delta \vdash \mathbb{M}(I, \vec{p})\tau : \star \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3$

Goal follows by (7), (9), (10)

▸ **Case 14:** *K-Pot.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash [I|\vec{p}]\tau : \star$

(2) $\Theta; \Delta \vdash I : \mathbb{N}$

(3) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+}$

(4) $\Psi; \Theta; \Delta \vdash \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash [I|\vec{p}]\tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By Theorem A.25 on (2) and (3)

(5) $\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi_1$

(6) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \Rightarrow \Phi_2$

(7) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

By IH on (4)

(8) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_3$

(9) $\Theta; \Delta \vDash \Phi_3$

By AK-Pot on (5), (6), (8)

(10) $\Psi; \Theta; \Delta \vdash [I|\vec{p}]\tau : \star \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3$

Goal follows by (7), (9), (10)

▸ **Case 15:** *K-ConstPot.*

- **Given:**

  (1) $\Psi; \Theta; \Delta \vdash [I]\, \tau : \star$

  (2) $\Theta; \Delta \vdash I : \mathbb{R}^+$

  (3) $\Psi; \Theta; \Delta \vdash \tau : \star$

- **Goal:**

  $\boxed{\Psi; \Theta; \Delta \vdash [I]\, \tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By Theorem A.25 on (2)

(4) $\Theta; \Delta \vdash I : \mathbb{R}^+ \Rightarrow \Phi_1$

(5) $\Theta; \Delta \vDash \Phi_1$

By IH on (3)

(6) $\Psi; \Theta; \Delta \vdash \tau : \star \Rightarrow \Phi_2$

(7) $\Theta; \Delta \vDash \Phi_2$

Applying AK-ConstPot to (4) and (6)

(8) $\Psi; \Theta; \Delta \vdash [I]\, \tau : \star \Rightarrow \Phi_1 \wedge \Phi_2$

Goal follows by (5), (7), (8)

- **Case 16:** *K-FamLam.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash \lambda i : S.\tau : S \to K$

    (2) $\Psi; \Theta, i : S; \Delta \vdash \tau : K$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash \lambda i : S.\tau : S \to K \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

  By IH on (2)

  (3) $\Psi; \Theta, i : S; \Delta \vdash \tau : K \Rightarrow \Phi$

  (4) $\Theta, i : S; \Delta \vDash \Phi$

  By AK-FamLam on (3)

  (5) $\Psi; \Theta; \Delta \vdash \lambda i : S.\tau : S \to K \Rightarrow \forall i : S.\Phi$

  Equivalently to (4)

  (6) $\Theta; \Delta \vDash \forall i : S.\Phi$

- **Case 17:** *K-FamApp.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \tau\ I : K$

(2) $\Psi; \Theta; \Delta \vdash \tau : S \to K$

(3) $\Theta; \Delta \vdash I : S$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \tau\ I : K \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi; \Theta; \Delta \vdash \tau : S \to K \Rightarrow \Phi_1$

(5) $\Theta; \Delta \vDash \Phi_1$

By Theorem A.25 on (3)

(6) $\Theta; \Delta \vdash I : S \Rightarrow \Phi_2$

(7) $\Theta; \Delta \vDash \Phi_2$

By AK-FamApp on (4) and (6)

(8) $\Psi; \Theta; \Delta \vdash \tau\ I : K \Rightarrow \Phi_1 \wedge \Phi_2$

Goal is done by (5), (7), and (8)

□

Proof of Theorem 8.14

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \tau : K$

(2) $\tau\ \texttt{nf}$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \tau <:_{\texttt{nf}} \tau : K \Rightarrow \Phi \text{ with } \Theta; \Delta \vDash \Phi}$

By induction on (1) and inversion on (2)

▸ **Case 1:** *K-Var.*

Immediate

▸ **Case 2:** *K-Unit.*

Immediate

▸ **Case 3:** *K-FamApp.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \tau\ I : K$

(2) $\tau\ I\ \texttt{ne}$

(3) $\Psi;\Theta;\Delta \vdash \tau : S \to K$

(4) $\Theta;\Delta \vdash I : S$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \tau\ I <:_{\texttt{nf}} \tau\ I : K \Rightarrow \Phi \text{ with } \Theta;\Delta \vDash \Phi}$

By inversion on (2)

(5) $\tau\ \texttt{ne}$

By IH on (3)

(6) $\Psi;\Theta;\Delta \vdash_p \tau <:_{\texttt{nf}} \tau : S \to K \Rightarrow \Phi$

(7) $\Theta;\Delta \vDash \Phi$

By AK-FamApp on (6)

(8) $\Psi;\Theta;\Delta \vdash \tau\ I <:_{\texttt{nf}} \tau\ I : K \Rightarrow \Phi \wedge (I = I)$

We re-establish the presupposition for (8) by applying Theorem 8.2 to $\Theta;\Delta \vdash I : S$ from (1)

(9) $\Psi;\Theta;\Delta \vdash_p \tau\ I <:_{\texttt{nf}} \tau\ I : K \Rightarrow \Phi \wedge (I = I)$

By (7), (9), and the fact that $\Theta;\Delta \vDash I = I$

□

Proof of Theorem 8.15

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \tau : K$

(2) $\tau\ \texttt{nf}$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \tau <:_{\texttt{nf}} \tau : K \Rightarrow \Phi \text{ with } \Theta;\Delta \vDash \Phi}$

By induction on (1), followed by inversion on (2)

▸ **Case 1:** *K-Var.*

Immediate by Theorem 8.14

- **Case 2:** *K-Unit.*

  Immediate by Theorem 8.14

- **Case 3:** *K-Arr.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \tau_1 \multimap \tau_2 : \star$

    (2) τ_1 and τ_2 $\texttt{nf}$

    (3) $\Psi; \Theta; \Delta \vdash \tau_1 : \star$

    (4) $\Psi; \Theta; \Delta \vdash \tau_2 : \star$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1 \multimap \tau_2 <:_{\texttt{nf}} \tau_1 \multimap \tau_2 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

  By IH on (3) and (4)

  (5) $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_1 : \star \Rightarrow \Phi_1$

  (6) $\Psi; \Theta; \Delta \vdash_p \tau_2 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi_2$

  (7) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

  By AS-Arr on (5) and (6)

  (8) $\Psi; \Theta; \Delta \vdash_p \tau_1 \multimap \tau_2 <:_{\texttt{nf}} \tau_1 \multimap \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

  Goal is immediate from (7), (8)

- **Case 4:** *K-Tensor.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \tau_1 \otimes \tau_2 : \star$

    (2) τ_1 and τ_2 $\texttt{nf}$

    (3) $\Psi; \Theta; \Delta \vdash \tau_1 : \star$

    (4) $\Psi; \Theta; \Delta \vdash \tau_2 : \star$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1 \otimes \tau_2 <:_{\texttt{nf}} \tau_1 \otimes \tau_2 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

  By IH on (3) and (4)

  (5) $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_1 : \star \Rightarrow \Phi_1$

(6) $\Psi; \Theta; \Delta \vdash_p \tau_2 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi_2$

(7) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

By AS-Tensor on (5) and (6)

(8) $\Psi; \Theta; \Delta \vdash_p \tau_1 \otimes \tau_2 <:_{\mathtt{nf}} \tau_1 \otimes \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

Goal is immediate from (7), (8)

▸ **Case 5:** *K-With.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \tau_1 \& \tau_2 : \star$

(2) τ_1 and τ_2 $\mathtt{nf}$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 : \star$

(4) $\Psi; \Theta; \Delta \vdash \tau_2 : \star$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1 \& \tau_2 <:_{\mathtt{nf}} \tau_1 \& \tau_2 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$$

By IH on (3) and (4)

(5) $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_1 : \star \Rightarrow \Phi_1$

(6) $\Psi; \Theta; \Delta \vdash_p \tau_2 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi_2$

(7) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

By AS-With on (5) and (6)

(8) $\Psi; \Theta; \Delta \vdash_p \tau_1 \& \tau_2 <:_{\mathtt{nf}} \tau_1 \& \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

Goal is immediate from (7), (8)

▸ **Case 6:** *K-Sum.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \tau_1 \oplus \tau_2 : \star$

(2) τ_1 and τ_2 $\mathtt{nf}$

(3) $\Psi; \Theta; \Delta \vdash \tau_1 : \star$

(4) $\Psi; \Theta; \Delta \vdash \tau_2 : \star$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1 \oplus \tau_2 <:_{\mathtt{nf}} \tau_1 \oplus \tau_2 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$$

By IH on (3) and (4)

(5) $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_1 : \star \Rightarrow \Phi_1$

(6) $\Psi; \Theta; \Delta \vdash_p \tau_2 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi_2$

(7) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

By AS-Sum on (5) and (6)

(8) $\Psi; \Theta; \Delta \vdash_p \tau_1 \oplus \tau_2 <:_{\texttt{nf}} \tau_1 \oplus \tau_2 : \star \Rightarrow \Phi_1 \wedge \Phi_2$

Goal is immediate from (7), (8)

▸ **Case 7:** *K-Bang.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p !\tau : \star$

(2) $\tau\ \texttt{nf}$

(3) $\Psi; \Theta; \Delta \vdash \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p !\tau <:_{\texttt{nf}} !\tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (3)

(4) $\Psi; \Theta; \Delta \vdash_p \tau <:_{\texttt{nf}} \tau : \star \Rightarrow \Phi$

(5) $\Theta; \Delta \vDash \Phi$

By AS-Bang on (4)

(6) $\Psi; \Theta; \Delta \vdash_p !\tau <:_{\texttt{nf}} !\tau : \star \Rightarrow \Phi$

▸ **Case 8:** *K-IForall.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \forall i : S.\tau : \star$

(2) $\tau\ \texttt{nf}$

(3) $\Psi; \Theta, i : S; \Delta \vdash \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \forall i : S.\tau <:_{\texttt{nf}} \forall i : S.\tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (3)

(4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau <:_{\texttt{nf}} \tau : \star \Rightarrow \Phi$

(5) $\Theta, i : S; \Delta \vDash \Phi$

By AS-IForall on (4)

(6) $\Psi; \Theta; \Delta \vdash_p \forall i : S.\tau <: \forall i : S.\tau : \star \Rightarrow \forall i : S.\Phi$

Equivalently to (5)

(7) $\Theta; \Delta \vDash \forall i : S.\Phi$

▸ **Case 9:** *K-IExists.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \exists i : S.\tau : \star$

(2) $\tau\ \texttt{nf}$

(3) $\Psi; \Theta, i : S; \Delta \vdash \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \exists i : S.\tau <:_{\texttt{nf}} \exists i : S.\tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (3)

(4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau <:_{\texttt{nf}} \tau : \star \Rightarrow \Phi$

(5) $\Theta, i : S; \Delta \vDash \Phi$

By AS-IExists on (4)

(6) $\Psi; \Theta; \Delta \vdash_p \exists i : S.\tau <: \exists i : S.\tau : \star \Rightarrow \forall i : S.\Phi$

Equivalently to (5)

(7) $\Theta; \Delta \vDash \forall i : S.\Phi$

▸ **Case 10:** *K-TForall.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash \forall \alpha : K.\tau : \star$

(2) $\tau\ \texttt{nf}$

(3) $\Psi, \alpha : K; \Theta; \Delta \vdash \tau : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \forall \alpha : K.\tau <:_{\texttt{nf}} \forall \alpha : K.\tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (3)

(4) $\Psi, \alpha : K; \Theta; \Delta \vdash_p \tau <:_{\texttt{nf}} \tau : \star \Rightarrow \Phi$

(5) $\Theta; \Delta \vDash \Phi$

By AS-TForall on (4)

(6) $\Psi; \Theta; \Delta \vdash \forall \alpha : K.\tau <:_{\texttt{nf}} \forall \alpha : K.\tau : \star \Rightarrow \Phi$

▸ **Case 11:** *K-List.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p L^I \tau : \star$

(2) $\tau\ \texttt{nf}$

(3) $\Psi; \Theta; \Delta \vdash \tau : \star$

(4) $\Theta; \Delta \vdash I : \mathbb{N}$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash L^I \tau <:_{\texttt{nf}}: L^I \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash_p \tau <:_{\texttt{nf}} \tau : \star \Rightarrow \Phi$

(6) $\Theta; \Delta \vDash \Phi$

By AS-List on (5)

(7) $\Psi; \Theta; \Delta \vdash L^I \tau <:_{\texttt{nf}} L^I \tau : \star \Rightarrow \Phi \wedge (I = I)$

By **??** applied to (4), we may re-establish the presupposition for (7)

(8) $\Psi; \Theta; \Delta \vdash_p L^I \tau <:_{\texttt{nf}} L^I \tau : \star \Rightarrow \Phi \wedge (I = I)$

The goal is immediate from (6), (8), and $\Theta; \Delta \vDash I = I$

▸ **Case 12:** *K-Conj.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \Phi' \& \tau : \star$

(2) $\tau\ \texttt{nf}$

(3) $\Psi; \Theta; \Delta \vdash \tau : \star$

(4) $\Theta; \Delta \vdash \Phi'\ \texttt{wf}$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash \Phi' \& \tau <:_{\texttt{nf}} \Phi' \& \tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash_p \tau <:_{\texttt{nf}} \tau : \star \Rightarrow \Phi$

(6) $\Theta; \Delta \vDash \Phi$

By AS-Conj on (5)

(7) $\Psi; \Theta; \Delta \vdash \Phi' \& \tau <:_{\texttt{nf}} \Phi' \& \tau : \star \Rightarrow \Phi \wedge (\Phi' \to \Phi')$

By Theorem 8.12 on (4), we may re-establish the presupposition for (7)

(8) $\Psi;\Theta;\Delta \vdash \Phi'\&\tau <:_{\texttt{nf}} \Phi'\&\tau : \star \Rightarrow \Phi \wedge (\Phi' \to \Phi')$

The goal follows from (6), (8), and $\Theta;\Delta \vDash \Phi' \to \Phi'$

▸ **Case 12:** *K-Impl.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \Phi' \implies \tau : \star$

(2) $\tau\ \texttt{nf}$

(3) $\Psi;\Theta;\Delta,\Phi' \vdash \tau : \star$

(4) $\Theta;\Delta \vdash \Phi'\ \texttt{wf}$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash \Phi' \implies \tau <:_{\texttt{nf}} \Phi' \implies \tau : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$$

By IH on (3)

(5) $\Psi;\Theta;\Delta,\Phi' \vdash_p \tau <:_{\texttt{nf}} \tau : \star \Rightarrow \Phi$

(6) $\Theta;\Delta,\Phi' \vDash \Phi$

By AS-Impl on (5)

(7) $\Psi;\Theta;\Delta \vdash \Phi' \implies \tau <:_{\texttt{nf}} \Phi' \implies \tau : \star \Rightarrow (\Phi' \to \Phi) \wedge (\Phi' \to \Phi')$

By Theorem 8.12 on (4), we may re-establish the presupposition for (7)

(8) $\Psi;\Theta;\Delta \vdash_p \Phi' \implies \tau <:_{\texttt{nf}} \Phi' \implies \tau : \star \Rightarrow (\Phi' \to \Phi) \wedge (\Phi' \to \Phi')$

The goal follows from (6), (8), and $\Theta;\Delta \vDash \Phi' \to \Phi'$

▸ **Case 13:** *K-Monad.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash \mathbb{M}(I,\vec{p})\tau : \star$

(2) $\tau\ \texttt{nf}$

(3) $\Psi;\Theta;\Delta \vdash \tau : \star$

(4) $\Theta;\Delta \vdash I : \mathbb{N}$

(5) $\Theta;\Delta \vdash \vec{p} : \vec{\mathbb{R}^+}$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash \mathbb{M}(I,\vec{p})\tau <:_{\texttt{nf}} \mathbb{M}(I,\vec{p})\tau : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$$

By IH on (3)

(6) $\Psi; \Theta; \Delta \vdash_p \tau <:_{\texttt{nf}} \tau : \star \Rightarrow \Phi$

(7) $\Theta; \Delta \vDash \Phi$

By AS-Monad on (6), followed by **??** on (4) and (5) to establish the presuppositions

(8) $\Psi; \Theta; \Delta \vdash_p \mathbb{M}(I, \vec{p})\tau <:_{\texttt{nf}} \mathbb{M}(I, \vec{p})\tau : \star \Rightarrow \Phi \wedge (I = I) \wedge (\vec{p} \leq \vec{p})$

Goal is immediate from (7) and (8)

▸ **Case 14:** *K-Pot.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash [I|\vec{p}]\tau : \star$

(2) $\tau\ \texttt{nf}$

(3) $\Psi; \Theta; \Delta \vdash \tau : \star$

(4) $\Theta; \Delta \vdash I : \mathbb{N}$

(5) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+}$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash [I|\vec{p}]\tau <:_{\texttt{nf}} [I|\vec{p}]\tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (3)

(6) $\Psi; \Theta; \Delta \vdash_p \tau <:_{\texttt{nf}} \tau : \star \Rightarrow \Phi$

(7) $\Theta; \Delta \vDash \Phi$

By AS-Pot on (6), followed by **??** on (4) and (5) to establish the presuppositions

(8) $\Psi; \Theta; \Delta \vdash_p [I|\vec{p}]\tau <:_{\texttt{nf}} [I|\vec{p}]\tau : \star \Rightarrow \Phi \wedge (I = I) \wedge (\vec{p} \leq \vec{p})$

Goal is immediate from (7) and (8)

▸ **Case 15:** *K-ConstPot.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p [I]\ \tau : \star$

(2) $\tau\ \texttt{nf}$

(3) $\Psi; \Theta; \Delta \vdash \tau : \star$

(4) $\Theta; \Delta \vdash I : \mathbb{R}^+$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash [I]\tau <:_{\texttt{nf}} [I]\tau : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (3)

(5) $\Psi; \Theta; \Delta \vdash_p \tau <:_{\texttt{nf}} \tau : \star \Rightarrow \Phi$

(6) $\Theta; \Delta \vDash \Phi$

By AS-ConstPot on (5) followed by Theorem 8.10 on (4) to re-establish the presupposition

(7) $\Psi; \Theta; \Delta \vdash [I]\,\tau <:_{\texttt{nf}} [I]\,\tau : \star \Rightarrow \Phi \wedge (I \leq I)$

Goal is immediate by (7), (6), and $\Theta; \Delta \vDash I \leq I$

- **Case 16:** *K-FamLam.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \lambda i : S.\tau : S \to K$

    (2) $\tau\ \texttt{nf}$

    (3) $\Psi; \Theta, i : S; \Delta \vdash \tau : K$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash \lambda i : S.\tau <:_{\texttt{nf}} \tau : S \to K \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

  By IH on (3)

  (4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau <:_{\texttt{nf}} \tau : K \Rightarrow \Phi$

  (5) $\Theta, i : S; \Delta \vDash \Phi$

  By AS-FamLam on (4)

  (6) $\Psi; \Theta; \Delta \vdash_p \lambda i : S.\tau <:_{\texttt{nf}} \lambda i : S\tau : S \to K \Rightarrow \forall i : S.\Phi$

  Equivalently to (5)

  (7) $\Theta; \Delta \vDash \forall i : S.\Phi$

- **Case 17:** *K-FamApp.*

  Immediate by Theorem 8.14

□

Proof of Theorem 8.17

- **Given:**

  (1) $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_2 : K \Rightarrow \Phi_1$

  (2) $\Psi; \Theta; \Delta \vdash_p \tau_2 <:_{\texttt{nf}} \tau_3 : K \Rightarrow \Phi_2$

  (3) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_3 : K \Rightarrow \Phi \text{ such that } \Theta;\Delta \vDash \Phi}$$

By strong induction on the sum of the sizes of (1) and (2). We note that for a given choice of final rule for (1), the final rule for (2) must be the same, by inspection of the rules generating $<:_{\texttt{nf}}$. For this reason, we present the proof as a case analysis over the rules for $<:_{\texttt{nf}}$.

▸ **Case 1:** *AS-Unit.*

Immediate.

▸ **Case 2:** *AS-Var.*

Immediate.

▸ **Case 3:** *AS-Arr.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \tau_1 \multimap \tau_1' <:_{\texttt{nf}} \tau_2 \multimap \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Psi;\Theta;\Delta \vdash_p \tau_2 \multimap \tau_2' <:_{\texttt{nf}} \tau_3 \multimap \tau_3' : \star \Rightarrow \Phi_3 \wedge \Phi_4$

(3) $\Theta;\Delta \vdash \bigwedge_{i=1}^{4} \Phi_i$

(4) $\Psi;\Theta;\Delta \vdash \tau_2 <:_{\texttt{nf}} \tau_1 : \star \Rightarrow \Phi_1$

(5) $\Psi;\Theta;\Delta \vdash \tau_1' <:_{\texttt{nf}} \tau_2' : \star \Rightarrow \Phi_2$

(6) $\Psi;\Theta;\Delta \vdash \tau_3 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi_3$

(7) $\Psi;\Theta;\Delta \vdash \tau_2' <:_{\texttt{nf}} \tau_3' : \star \Rightarrow \Phi_4$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash_p \tau_1 \multimap \tau_1' <:_{\texttt{nf}} \tau_3 \multimap \tau_3' : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$$

By IH on (4) and (6)

(8) $\Psi;\Theta;\Delta \vdash_p \tau_3 <:_{\texttt{nf}} \tau_1 : \star \Rightarrow \Phi_1'$

(9) $\Theta;\Delta \vDash \Phi_1'$

By IH on (5) and (7)

10 $\Psi;\Theta;\Delta \vdash_p \tau_1' <:_{\texttt{nf}} \tau_3' : \star \Rightarrow \Phi_2'$

(11) $\Theta;\Delta \vDash \Phi_2'$

By AS-Arr on (8) and (10)

(12) $\Psi;\Theta;\Delta \vdash_p \tau_1 \multimap \tau_1' <:_{\texttt{nf}} \tau_3 \multimap \tau_3' : \star \Rightarrow \Phi_1' \wedge \Phi_2'$

The Goal follows by (9), (11), and (12)

▸ **Case 4:** *AS-Tensor.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \tau_1 \otimes \tau_1' <:_{\mathtt{nf}} \tau_2 \otimes \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Psi;\Theta;\Delta \vdash_p \tau_2 \otimes \tau_2' <:_{\mathtt{nf}} \tau_3 \otimes \tau_3' : \star \Rightarrow \Phi_3 \wedge \Phi_4$

(3) $\Theta;\Delta \vdash \bigwedge_{i=1}^{4} \Phi_i$

(4) $\Psi;\Theta;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi_1$

(5) $\Psi;\Theta;\Delta \vdash \tau_1' <:_{\mathtt{nf}} \tau_2' : \star \Rightarrow \Phi_2$

(6) $\Psi;\Theta;\Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi_3$

(7) $\Psi;\Theta;\Delta \vdash \tau_2' <:_{\mathtt{nf}} \tau_3' : \star \Rightarrow \Phi_4$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash_p \tau_1 \otimes \tau_1' <:_{\mathtt{nf}} \tau_3 \otimes \tau_3' : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$$

By IH on (4) and (6)

(8) $\Psi;\Theta;\Delta \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi_1'$

(9) $\Theta;\Delta \vDash \Phi_1'$

By IH on (5) and (7)

10 $\Psi;\Theta;\Delta \vdash_p \tau_1' <:_{\mathtt{nf}} \tau_3' : \star \Rightarrow \Phi_2'$

(11) $\Theta;\Delta \vDash \Phi_2'$

By AS-Tensor on (8) and (10)

(12) $\Psi;\Theta;\Delta \vdash_p \tau_1 \otimes \tau_1' <:_{\mathtt{nf}} \tau_3 \otimes \tau_3' : \star \Rightarrow \Phi_1' \wedge \Phi_2'$

The Goal follows by (9), (11), and (12)

▸ **Case 5:** *AS-With.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \tau_1 \& \tau_1' <:_{\mathtt{nf}} \tau_2 \& \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Psi;\Theta;\Delta \vdash_p \tau_2 \& \tau_2' <:_{\mathtt{nf}} \tau_3 \& \tau_3' : \star \Rightarrow \Phi_3 \wedge \Phi_4$

(3) $\Theta;\Delta \vdash \bigwedge_{i=1}^{4} \Phi_i$

(4) $\Psi;\Theta;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi_1$

(5) $\Psi;\Theta;\Delta \vdash \tau_1' <:_{\mathtt{nf}} \tau_2' : \star \Rightarrow \Phi_2$

(6) $\Psi;\Theta;\Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi_3$

(7) $\Psi;\Theta;\Delta \vdash \tau_2' <:_{\mathtt{nf}} \tau_3' : \star \Rightarrow \Phi_4$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \tau_1 \& \tau_1' <:_{\mathtt{nf}} \tau_3 \& \tau_3' : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

By IH on (4) and (6)

(8) $\Psi;\Theta;\Delta \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi_1'$

(9) $\Theta;\Delta \vDash \Phi_1'$

By IH on (5) and (7)

10 $\Psi;\Theta;\Delta \vdash_p \tau_1' <:_{\mathtt{nf}} \tau_3' : \star \Rightarrow \Phi_2'$

(11) $\Theta;\Delta \vDash \Phi_2'$

By AS-With on (8) and (10)

(12) $\Psi;\Theta;\Delta \vdash_p \tau_1 \& \tau_1' <:_{\mathtt{nf}} \tau_3 \& \tau_3' : \star \Rightarrow \Phi_1' \wedge \Phi_2'$

The Goal follows by (9), (11), and (12)

▸ **Case 6:** *AS-Sum.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \tau_1 \oplus \tau_1' <:_{\mathtt{nf}} \tau_2 \oplus \tau_2' : \star \Rightarrow \Phi_1 \wedge \Phi_2$

(2) $\Psi;\Theta;\Delta \vdash_p \tau_2 \oplus \tau_2' <:_{\mathtt{nf}} \tau_3 \oplus \tau_3' : \star \Rightarrow \Phi_3 \wedge \Phi_4$

(3) $\Theta;\Delta \vdash \bigwedge_{i=1}^{4} \Phi_i$

(4) $\Psi;\Theta;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi_1$

(5) $\Psi;\Theta;\Delta \vdash \tau_1' <:_{\mathtt{nf}} \tau_2' : \star \Rightarrow \Phi_2$

(6) $\Psi;\Theta;\Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi_3$

(7) $\Psi;\Theta;\Delta \vdash \tau_2' <:_{\mathtt{nf}} \tau_3' : \star \Rightarrow \Phi_4$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \tau_1 \oplus \tau_1' <:_{\mathtt{nf}} \tau_3 \oplus \tau_3' : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

By IH on (4) and (6)

(8) $\Psi;\Theta;\Delta \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi_1'$

(9) $\Theta;\Delta \vDash \Phi_1'$

By IH on (5) and (7)

10 $\Psi;\Theta;\Delta \vdash_p \tau_1' <:_{\mathtt{nf}} \tau_3' : \star \Rightarrow \Phi_2'$

(11) $\Theta;\Delta \vDash \Phi_2'$

By AS-Sum on (8) and (10)

(12) $\Psi;\Theta;\Delta \vdash_p \tau_1 \oplus \tau_1' <:_{\mathtt{nf}} \tau_3 \oplus \tau_3' : \star \Rightarrow \Phi_1' \wedge \Phi_2'$

The Goal follows by (9), (11), and (12)

▸ **Case 7:** *AS-Bang.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p !\tau_1 <:_{\mathtt{nf}} !\tau_2 : \star \Rightarrow \Phi_1$

(2) $\Psi;\Theta;\Delta \vdash_p !\tau_2 <:_{\mathtt{nf}} !\tau_3 : \star \Rightarrow \Phi_2$

(3) $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$

(4) $\Psi;\Theta;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi_1$

(5) $\Psi;\Theta;\Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p !\tau_1 <:_{\mathtt{nf}} !\tau_3 : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

By IH on (4) and (5)

(6) $\Psi;\Theta;\Delta \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow Phi$

(7) $\Theta;\Delta \vDash \Phi$

By AS-Bang on (6)

(8) $\Psi;\Theta;\Delta \vdash_p !\tau_1 <:_{\mathtt{nf}} !\tau_3 : \star \Rightarrow Phi$

▸ **Case 8:** *AS-IForall.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \forall i : S.\tau_1 <:_{\mathtt{nf}} \forall i : S.\tau_2 : \star \Rightarrow \forall i : S.\Phi_1$

(2) $\Psi;\Theta;\Delta \vdash_p \forall i : S.\tau_2 <:_{\mathtt{nf}} \forall i : S.\tau_3 : \star \Rightarrow \forall i : S.\Phi_2$

(3) $\Theta;\Delta \vDash \forall i : S.\Phi_1 \wedge \forall i : S.\Phi_2$

(4) $\Psi;\Theta, i : S;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi_1$

(5) $\Psi;\Theta, i : S;\Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi_3$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \forall i : S.\tau_1 <:_{\mathtt{nf}} \forall i : S.\tau_3 : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

Equivalently to (3), $\Theta, i : S;\Delta \vDash \Phi_1$ and $\Theta, i : S;\Delta \vDash \Phi_2$, and so by IH on (4),(5)

(6) $\Psi;\Theta, i : S;\Delta \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi$

(7) $\Theta, i : S : \Delta \vDash \Phi$

By AS-IForall on (6)

(8) $\Psi;\Theta;\Delta \vdash_p \forall i : S.\tau_1 <:_{\mathtt{nf}} \forall i : S.\tau_3 : \star \Rightarrow \forall i : S.\Phi$

Equivalently to (7)

(9) $\Theta; \Delta \vDash \forall i : S.\Phi$

The goal follows by (8) and (9)

- **Case 9:** *AS-IExists.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \exists i : S.\tau_1 <:_{\mathtt{nf}} \exists i : S.\tau_2 : \star \Rightarrow \forall i : S.\Phi_1$

    (2) $\Psi; \Theta; \Delta \vdash_p \exists i : S.\tau_2 <:_{\mathtt{nf}} \exists i : S.\tau_3 : \star \Rightarrow \forall i : S.\Phi_2$

    (3) $\Theta; \Delta \vDash \forall i : S.\Phi_1 \wedge \forall i : S.\Phi_2$

    (4) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi_1$

    (5) $\Psi; \Theta, i : S; \Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi_3$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \exists i : S.\tau_1 <:_{\mathtt{nf}} \exists i : S.\tau_3 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

  Equivalently to (3), $\Theta, i : S; \Delta \vDash \Phi_1$ and $\Theta, i : S; \Delta \vDash \Phi_2$, and so by IH on (4),(5)

  (6) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi$

  (7) $\Theta, i : S : \Delta \vDash \Phi$

  By AS-IExists on (6)

  (8) $\Psi; \Theta; \Delta \vdash_p \exists i : S.\tau_1 <:_{\mathtt{nf}} \exists i : S.\tau_3 : \star \Rightarrow \forall i : S.\Phi$

  Equivalently to (7)

  (9) $\Theta; \Delta \vDash \forall i : S.\Phi$

  The goal follows by (8) and (9)

- **Case 10:** *AS-TForall.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta \vdash_p \forall \alpha : K.\tau_1 <:_{\mathtt{nf}} \forall \alpha : K.\tau_2 : \star \Rightarrow \Phi_1$

    (2) $\Psi; \Theta; \Delta \vdash_p \forall \alpha : K.\tau_2 <:_{\mathtt{nf}} \forall \alpha : K.\tau_3 : \star \Rightarrow \Phi_2$

    (3) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2$

    (4) $\Psi, \alpha : K; \Theta; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi_1$

    (5) $\Psi, \alpha : K; \Theta; \Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi_2$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p \forall \alpha : K.\tau_1 <:_{\mathtt{nf}} \forall \alpha : K.\tau_3 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

  By IH on (4),(5)

(6) $\Psi, \alpha : K; \Theta; \Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_3 : \star \Rightarrow \Phi$

(7) $\Theta; \Delta \vDash \Phi$

By AS-TForall on (6)

(8) $\Psi; \Theta; \Delta \vdash_p \forall\alpha : K.\tau_1 <:_{\texttt{nf}} \forall\alpha : K.\tau_3 : \star \Rightarrow \Phi$

▸ **Case 11:** *AS-List.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p L^I\tau_1 <:_{\texttt{nf}} L^J\tau_2 : \star \Rightarrow \Phi_1 \wedge (I = J)$

(2) $\Psi; \Theta; \Delta \vdash_p L^J\tau_2 <:_{\texttt{nf}} L^K\tau_3 : \star \Rightarrow \Phi_2 \wedge (J = K)$

(3) $\Theta; \Delta \vDash \Phi_1 \wedge \Phi_2 \wedge (I = J) \wedge (J = K)$

(4) $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi_1$

(5) $\Psi; \Theta; \Delta \vdash_p \tau_2 <:_{\texttt{nf}} \tau_3 : \star \Rightarrow \Phi_2$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash_p L^I\tau_1 <:_{\texttt{nf}} L^K\tau_3 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$$

By IH on (4), (5)

(6) $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_3 : \star \Rightarrow \Phi$

(7) $\Theta; \Delta \vDash \Phi$

By AS-List on (7)

(8) $\Psi; \Theta; \Delta \vdash_p \tau_1^I <:_{\texttt{nf}} \tau_3^K : \star \Rightarrow \Phi \wedge (I = K)$

By (3)

(9) $\Theta; \Delta \vDash I = K$

The goal follows by (7), (8), (9)

▸ **Case 12:** *AS-Impl.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \Phi_1 \implies \tau_1 <:_{\texttt{nf}} \Phi_2 \implies \tau_2 : \star \Rightarrow (\Phi_2 \to \Phi_1') \wedge (\Phi_2 \to \Phi_1)$

(2) $\Psi; \Theta; \Delta \vdash_p \Phi_2 \implies \tau_2 <:_{\texttt{nf}} \Phi_3 \implies \tau_3 : \star \Rightarrow (\Phi_3 \to \Phi_2') \wedge (\Phi_3 \to \Phi_2)$

(3) $\Theta; \Delta \vDash \Phi_1' \wedge \Phi_2' \wedge (Phi_3 \to \Phi_2) \wedge (Phi_2 \to \Phi_1)$

(4) $\Psi; \Theta; \Delta, \Phi_2 \vdash_p \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi_1'$

(5) $\Psi; \Theta; \Delta, \Phi_3 \vdash_p \tau_2 <:_{\texttt{nf}} \tau_3 : \star \Rightarrow \Phi_2'$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash \Phi_1 \implies \tau_1 <:_{\mathtt{nf}} \Phi_3 \implies \tau_3 : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

FIXME FIXME

By Theorem A.31 on (4)

(6) $\Psi;\Theta;\Delta,\Phi_3 \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi_1''$

(7) $\Theta;\Delta \vDash \Phi_1''$

By IH on (5), (6)

(8) $\Psi;\Theta;\Delta,\Phi_3 \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi$

(9) $\Theta;\Delta \vDash \Phi$

By AS-Impl on (8)

(10) $\Psi;\Theta;\Delta \vdash_p \Phi_1 \implies \tau_1 <:_{\mathtt{nf}} \Phi_3 \implies \tau_3 : \star \Rightarrow \Phi \wedge (\Phi_3 \to \Phi_1)$

By (3)

(11) $\Theta;\Delta \vDash \Phi_3 \to \Phi_1$

The goal follows by (7), (8), (9)

▸ **Case 13:** *AS-Conj.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \Phi_1 \& \tau_1 <:_{\mathtt{nf}} \Phi_2 \& \tau_2 : \star \Rightarrow \Phi_1' \wedge (\Phi_1 \to \Phi_2)$

(2) $\Psi;\Theta;\Delta \vdash_p \Phi_2 \& \tau_2 <:_{\mathtt{nf}} \Phi_3 \& \tau_3 : \star \Rightarrow \Phi_2' \wedge (\Phi_2 \to \Phi_3)$

(3) $\Theta;\Delta \vDash \Phi_1' \wedge \Phi_2' \wedge (Phi_1 \to \Phi_3) \wedge (Phi_1 \to \Phi_2)$

(4) $\Psi;\Theta;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi_1'$

(5) $\Psi;\Theta;\Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi_2'$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash \Phi_1 \& \tau_1 <:_{\mathtt{nf}} \Phi_3 \& \tau_3 : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

By IH on (4), (5)

(6) $\Psi;\Theta;\Delta \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_3 : \star \Rightarrow \Phi$

(7) $\Theta;\Delta \vDash \Phi$

By AS-Conj on (6)

(8) $\Psi;\Theta;\Delta \vdash_p \Phi_1 \& \tau_1 <:_{\mathtt{nf}} \Phi_3 \& \tau_3 : \star \Rightarrow \Phi \wedge (\Phi_1 \to \Phi_3)$

By (3)

(9) $\Theta;\Delta \vDash \Phi_1 \to \Phi_3$

The goal follows by (7), (8), (9)

▸ **Case 14:** *AS-Monad.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \mathbb{M}(I,\vec{q})\tau_1 <:_{\texttt{nf}} \mathbb{M}(J,\vec{p})\tau_2 : \star \Rightarrow (I = J) \wedge (\vec{q} \leq \vec{p}) \wedge \Phi_1$

(2) $\Psi;\Theta;\Delta \vdash_p \mathbb{M}(J,\vec{p})\tau_2 <:_{\texttt{nf}} \mathbb{M}(K,\vec{l})\tau_3 : \star \Rightarrow (J = K) \wedge (\vec{p} \leq \vec{l}) \wedge \Phi_2$

(3) $\Theta;\Delta \vDash (I = J) \wedge (J = K) \wedge (\vec{q} \leq \vec{p}) \wedge (\vec{p} \leq \vec{l}) \wedge \Phi_1 \wedge \Phi_2$

(4) $\Psi;\Theta;\Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi_1$

(5) $\Psi;\Theta;\Delta \vdash \tau_2 <:_{\texttt{nf}} \tau_3 : \star \Rightarrow \Phi_2$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash_p \mathbb{M}(I,\vec{q})\tau_1 <:_{\texttt{nf}} \mathbb{M}(K,\vec{l})\tau_3 : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$$

By IH on (4), (5)

(7) $\Psi;\Theta;\Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_3 : \star \Rightarrow \Phi$

(8) $\Theta;\Delta \vDash \Phi$

By (3)

(9) $\Theta;\Delta \vDash (I = K) \wedge (\vec{q} \leq \vec{l})$

By AS-Monad on (7)

(10) $\Psi;\Theta;\Delta \vdash_p \mathbb{M}(I,\vec{q})\tau_1 <:_{\texttt{nf}} \mathbb{M}(K,\vec{p})\tau_3 : \star \Rightarrow (I = K) \wedge (\vec{q} \leq \vec{l}) \wedge \Phi$

Goal follows by (8), (9), (10)

▸ **Case 15:** *AS-Pot.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p [I|\vec{q}]\tau_1 <:_{\texttt{nf}} [J|\vec{p}]\tau_2 : \star \Rightarrow (I = J) \wedge (\vec{q} \geq \vec{p}) \wedge \Phi_1$

(2) $\Psi;\Theta;\Delta \vdash_p [J|\vec{p}]\tau_2 <:_{\texttt{nf}} [K|\vec{l}]\tau_3 : \star \Rightarrow (J = K) \wedge (\vec{p} \geq \vec{l}) \wedge \Phi_2$

(3) $\Theta;\Delta \vDash (I = J) \wedge (J = K) \wedge (\vec{q} \geq \vec{p}) \wedge (\vec{p} \geq \vec{l}) \wedge \Phi_1 \wedge \Phi_2$

(4) $\Psi;\Theta;\Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi_1$

(5) $\Psi;\Theta;\Delta \vdash \tau_2 <:_{\texttt{nf}} \tau_3 : \star \Rightarrow \Phi_2$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash_p [I|\vec{q}]\tau_1 <:_{\texttt{nf}} [K|\vec{l}]\tau_3 : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$$

By IH on (4), (5)

(7) $\Psi;\Theta;\Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_3 : \star \Rightarrow \Phi$

(8) $\Theta;\Delta \vDash \Phi$

By (3)

(9) $\Theta; \Delta \vDash (I = K) \wedge (\vec{q} \geq \vec{l})$

By AS-Pot on (7)

(10) $\Psi; \Theta; \Delta \vdash_p [I|\vec{q}]\tau_1 <:_{\texttt{nf}} [K|\vec{p}]\tau_3 : \star \Rightarrow (I = K) \wedge (\vec{q} \geq \vec{l}) \wedge \Phi$

Goal follows by (8), (9), (10)

▸ **Case 16:** *AS-ConstPot.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p [I]\tau_1 <:_{\texttt{nf}} [J]\tau_2 : \star \Rightarrow (I \leq J) \wedge \Phi_1$

(2) $\Psi; \Theta; \Delta \vdash_p [J]\tau_2 <:_{\texttt{nf}} [K]\tau_3 : \star \Rightarrow (J \leq K) \wedge \Phi_2$

(3) $\Theta; \Delta \vDash (I \leq J) \wedge (J \leq K) \wedge \Phi_1 \wedge \Phi_2$

(4) $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi_1$

(5) $\Psi; \Theta; \Delta \vdash \tau_2 <:_{\texttt{nf}} \tau_3 : \star \Rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p [I]\tau_1 <:_{\texttt{nf}} [K]\tau_3 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (4), (5)

(7) $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_3 : \star \Rightarrow \Phi$

(8) $\Theta; \Delta \vDash \Phi$

By (3)

(9) $\Theta; \Delta \vDash (I \leq K)$

By AS-ConstPot on (7)

(10) $\Psi; \Theta; \Delta \vdash_p [I]\tau_1 <:_{\texttt{nf}} [K]\tau_3 : \star \Rightarrow (I \leq K) \wedge \Phi$

Goal follows by (8), (9), (10)

▸ **Case 17:** *AS-FamLam.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \lambda i : S.\tau_1 <:_{\texttt{nf}} \lambda i : S.\tau_2 : S \to K \Rightarrow \forall i : S.\Phi_1$

(2) $\Psi; \Theta; \Delta \vdash_p \lambda i : S.\tau_2 <:_{\texttt{nf}} \lambda i : S.\tau_3 : S \to K \Rightarrow \forall i : S.\Phi_2$

(3) $\Theta; \Delta \vDash \forall i : S.\Phi_1 \wedge \forall i : S.\Phi_2$

(4) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : K \Rightarrow \Phi_1$

(5) $\Psi; \Theta, i : S; \Delta \vdash \tau_2 <:_{\texttt{nf}} \tau_3 : K \Rightarrow \Phi_2$

▸ **Goal:**

$$\boxed{Psi; \Theta; \Delta \vdash_p \lambda i : S.\tau_1 <:_{\mathtt{nf}} \lambda i : S.\tau_3 : S \to K \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$$

By IH on (4) and (5)

(6) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_3 : K \Rightarrow \Phi$

(7) $\Theta, i : S; \Delta \vDash \Phi$

By AS-FamLam on (6)

(8) $\Psi; \Theta; \Delta \vdash_p \lambda i : S.\tau_1 <:_{\mathtt{nf}} \lambda i : S.\tau_3 : S \to K \Rightarrow \forall i : S.\Phi$

Equivalently to (7)

(9) $\Theta; \Delta \vDash \forall i : S.\Phi$

The Goal follows from (8) and (9)

▸ **Case 18:** *AS-FamApp.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \tau_1\ I <:_{\mathtt{nf}} \tau_2\ J : K \Rightarrow (I = J) \wedge \Phi_1$

(2) $\Psi; \Theta; \Delta \vdash_p \tau_2\ J <:_{\mathtt{nf}} \tau_3\ L : K \Rightarrow (J = L) \wedge \Phi_2$

(3) $\Theta; \Delta \vDash (I = J) \wedge (J = L) \wedge \Phi_1 \wedge \Phi_2$

(4) $\Psi; \Theta; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : S \to K \Rightarrow \Phi_1$

(5) $\Psi; \Theta; \Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_3 : S \to K \Rightarrow \Phi_2$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1\ I <:_{\mathtt{nf}} \tau_3\ L : K \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$$

By IH on (4) and (5)

(5) $\Psi; \Theta; \Delta \vdash_p \tau_1 <:_{\mathtt{nf}} \tau_3 : S \to K \Rightarrow \Phi$

(6) $\Theta; \Delta \vDash \Phi$

By (3)

(7) $\Theta; \Delta \vDash (I = L)$

By AS-FamApp on (5)

(8) $\Psi; \Theta; \Delta \vdash_p \tau_1\ I <:_{\mathtt{nf}} \tau_3\ L : K \Rightarrow (I = L) \wedge \Phi$

The Goal follows by (6),(7),(8)

□

Proof of Theorem 8.19

▸ **Given:**

(1) $\Psi;\Theta,i:S;\Delta \vdash_p \tau_1 <:_{\texttt{nf}} \tau_2 : K \Rightarrow \Phi$

(2) $\Theta;\Delta \vDash \Phi$

(3) $\Theta \vdash \Delta\ \texttt{wf}$

(4) $\Theta;\Delta \vdash_p I : S \Rightarrow \Phi_1$ with $\Theta;\Delta \vDash \Phi_1$

(5) $\Theta;\Delta \vdash_p J : S \Rightarrow \Phi_2$ with $\Theta;\Delta \vDash \Phi_2$

(6) $\Theta;\Delta \vDash I = J$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash_p \tau_1[I/i] <:_{\texttt{nf}} \tau_2[J/i] : K \Rightarrow \Phi' \text{ for some } \Phi' \text{ with } \Theta;\Delta \vDash \Phi'}$$

In most cases, it suffices to show that $\Psi;\Theta;\Delta \vdash \tau_1[I/i] <:_{\texttt{nf}} \tau_2[J/i] : K \Rightarrow \Phi'$ for some solvable Φ' since Theorem 8.11 on (3), Theorem 7.1 along with the presuppositions for (1) give the presuppositions for the conclusion, using Theorem A.34. When this is not immediate, we manually reconstruct the presuppositions required.

▸ **Case 1:** *AS-Unit.*

Immediate.

▸ **Case 2:** *AS-Var.*

Immediate.

▸ **Case 3:** *AS-Arr.*

▸ **Given:**

(7) $\Psi;\Theta,i:S;\Delta \vdash_p \tau_1 \multimap \tau_2 <:_{\texttt{nf}} \tau_1' \multimap \tau_2' : \star \Rightarrow \Phi_1' \wedge \Phi_2'$

(8) $\Psi;\Theta,i:S;\Delta \vdash \tau_1' <:_{\texttt{nf}} \tau_1 : \star \Rightarrow \Phi_1'$

(9) $\Psi;\Theta,i:S;\Delta \vdash \tau_2 <:_{\texttt{nf}} \tau_2' : \star \Rightarrow \Phi_2'$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash (\tau_1 \multimap \tau_2)[I/i] <:_{\texttt{nf}} (\tau_1' \multimap \tau_2')[J/i] : \star \Rightarrow \Phi' \text{ for some } \Theta;\Delta \vDash \Phi'}$$

By IH on (8) and (9)

(10) $\Psi;\Theta;\Delta \vdash_p \tau_1'[J/i] <:_{\texttt{nf}} \tau_1[I/i] : \star \Rightarrow \Phi_1''$

(11) $\Theta;\Delta \vDash \Phi_1''$

(12) $\Psi;\Theta;\Delta \vdash \tau_2[I/i] <:_{\texttt{nf}} \tau_2'[J/i] : \star \Rightarrow \Phi_2''$

(13) $\Theta;\Delta \vDash \Phi_2''$

By AS-Arr on (10) and (12)

(14) $\Psi;\Theta;\Delta \vdash_p \tau_1[I/i] \multimap \tau_2[I/i] <:_{\mathtt{nf}} \tau_1'[J/i] \multimap \tau_2'[J/i] : \star \Rightarrow \Phi_1'' \wedge \Phi_2''$

Goal follows immediately from (14), with $\Phi' = \Phi_1'' \wedge \Phi_2''$

▸ **Case 4:** *AS-Tensor.*

▸ **Given:**

(7) $\Psi;\Theta, i : S;\Delta \vdash_p \tau_1 \otimes \tau_2 <:_{\mathtt{nf}} \tau_1' \otimes \tau_2' : \star \Rightarrow \Phi_1' \wedge \Phi_2'$

(8) $\Psi;\Theta, i : S;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_1' : \star \Rightarrow \Phi_1'$

(9) $\Psi;\Theta, i : S;\Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_2' : \star \Rightarrow \Phi_2'$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash (\tau_1 \otimes \tau_2)[I/i] <:_{\mathtt{nf}} (\tau_1' \otimes \tau_2')[J/i] : \star \Rightarrow \Phi' \text{ for some } \Theta;\Delta \vDash \Phi'}$

By IH on (8) and (9)

(10) $\Psi;\Theta;\Delta \vdash_p \tau_1[J/i] <:_{\mathtt{nf}} \tau_1'[I/i] : \star \Rightarrow \Phi_1''$

(11) $\Theta;\Delta \vDash \Phi_1''$

(12) $\Psi;\Theta;\Delta \vdash \tau_2[I/i] <:_{\mathtt{nf}} \tau_2'[J/i] : \star \Rightarrow \Phi_2''$

(13) $\Theta;\Delta \vDash \Phi_2''$

By AS-Tensor on (10) and (12)

(14) $\Psi;\Theta;\Delta \vdash_p \tau_1[I/i] \otimes \tau_2[I/i] <:_{\mathtt{nf}} \tau_1'[J/i] \otimes \tau_2'[J/i] : \star \Rightarrow \Phi_1'' \wedge \Phi_2''$

Goal follows immediately from (14), with $\Phi' = \Phi_1'' \wedge \Phi_2''$

▸ **Case 5:** *AS-With.*

▸ **Given:**

(7) $\Psi;\Theta, i : S;\Delta \vdash_p \tau_1 \& \tau_2 <:_{\mathtt{nf}} \tau_1' \& \tau_2' : \star \Rightarrow \Phi_1' \wedge \Phi_2'$

(8) $\Psi;\Theta, i : S;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_1' : \star \Rightarrow \Phi_1'$

(9) $\Psi;\Theta, i : S;\Delta \vdash \tau_2 <:_{\mathtt{nf}} \tau_2' : \star \Rightarrow \Phi_2'$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash (\tau_1 \& \tau_2)[I/i] <:_{\mathtt{nf}} (\tau_1' \& \tau_2')[J/i] : \star \Rightarrow \Phi' \text{ for some } \Theta;\Delta \vDash \Phi'}$

By IH on (8) and (9)

(10) $\Psi;\Theta;\Delta \vdash_p \tau_1[J/i] <:_{\mathtt{nf}} \tau_1'[I/i] : \star \Rightarrow \Phi_1''$

(11) $\Theta;\Delta \vDash \Phi_1''$

(12) $\Psi;\Theta;\Delta \vdash \tau_2[I/i] <:_{\mathtt{nf}} \tau_2'[J/i] : \star \Rightarrow \Phi_2''$

(13) $\Theta; \Delta \vDash \Phi_2''$

By AS-With on (10) and (12)

(14) $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] \& \tau_2[I/i] <:_{\texttt{nf}} \tau_1'[J/i] \& \tau_2'[J/i] : \star \Rightarrow \Phi_1'' \wedge \Phi_2''$

Goal follows immediately from (14), with $\Phi' = \Phi_1'' \wedge \Phi_2''$

▸ **Case 6:** *AS-Sum.*

▸ **Given:**

(7) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 \oplus \tau_2 <:_{\texttt{nf}} \tau_1' \oplus \tau_2' : \star \Rightarrow \Phi_1' \wedge \Phi_2'$

(8) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_1' : \star \Rightarrow \Phi_1'$

(9) $\Psi; \Theta, i : S; \Delta \vdash \tau_2 <:_{\texttt{nf}} \tau_2' : \star \Rightarrow \Phi_2'$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash (\tau_1 \oplus \tau_2)[I/i] <:_{\texttt{nf}} (\tau_1' \oplus \tau_2')[J/i] : \star \Rightarrow \Phi' \text{ for some } \Theta; \Delta \vDash \Phi'}$

By IH on (8) and (9)

(10) $\Psi; \Theta; \Delta \vdash_p \tau_1[J/i] <:_{\texttt{nf}} \tau_1'[I/i] : \star \Rightarrow \Phi_1''$

(11) $\Theta; \Delta \vDash \Phi_1''$

(12) $\Psi; \Theta; \Delta \vdash \tau_2[I/i] <:_{\texttt{nf}} \tau_2'[J/i] : \star \Rightarrow \Phi_2''$

(13) $\Theta; \Delta \vDash \Phi_2''$

By AS-Sum on (10) and (12)

(14) $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] \oplus \tau_2[I/i] <:_{\texttt{nf}} \tau_1'[J/i] \oplus \tau_2'[J/i] : \star \Rightarrow \Phi_1'' \wedge \Phi_2''$

Goal follows immediately from (14), with $\Phi' = \Phi_1'' \wedge \Phi_2''$

▸ **Case 7:** *AS-Bang.*

▸ **Given:**

(7) $\Psi; \Theta, i : S; \Delta \vdash_p !\tau_1 <:_{\texttt{nf}} !\tau_2 : \star \Rightarrow \Phi$

(8) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash (!\tau_1)[I/i] <:_{\texttt{nf}} (!\tau_2)[J/i] : \star \Rightarrow \Phi' \text{ for some } \Theta; \Delta \vDash \Phi'}$

By IH on (8)

(9) $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] <:_{\texttt{nf}} \tau_2[J/i] : \star \Rightarrow \Phi'$

(10) $\Theta; \Delta \vDash \Phi'$

By AS-Bang on (9)

(9) $\Psi; \Theta; \Delta \vdash_p !\tau_1[I/i] <:_{\texttt{nf}} !\tau_2[J/i] : \star \Rightarrow \Phi'$

Goal follows immediately from (9)

▸ **Case 8:** *AS-IForall.*

▸ **Given:**

(7) $\Psi; \Theta, i : S; \Delta \vdash_p \forall j : S'.\tau_1 <:_{\texttt{nf}} \forall j : S'.\tau_2 : \star \Rightarrow \forall j : S'.\Phi$

(8) $\Psi; \Theta, i : S, j : S'; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash (\forall j : S'.\tau_1)[I/i] <:_{\texttt{nf}} (\forall j : S'.\tau_2)[J/i] : \star \Rightarrow \Phi' \text{ for some } \Theta; \Delta \vDash \Phi'}$

By (2)

(9) $\Theta, i : S, j : S; \Delta \vDash \Phi$

IH on (8)

(10) $\Psi; \Theta j : S'; \Delta \vdash \tau_1[I/i] <:_{\texttt{nf}} \tau_2[J//i] : \star \Rightarrow \Phi'$

(11) $\Theta, j : S'; \Delta \vDash \Phi'$

Equivalently to (11)

(12) $\Theta; \Delta \vDash \forall j : S'.\Phi'$

By AS-IForall on (10)

(13) $Psi; \Theta; \Delta \vdash \forall j : S'.\tau_1[I/i] <:_{\texttt{nf}} \forall j : S'.\tau_2[J/i] : \star \Rightarrow \forall j : S'.\Phi'$

Goal follows by (12) and (13)

▸ **Case 8:** *AS-IExists.*

▸ **Given:**

(7) $\Psi; \Theta, i : S; \Delta \vdash_p \exists j : S'.\tau_1 <:_{\texttt{nf}} \exists j : S'.\tau_2 : \star \Rightarrow \forall j : S'.\Phi$

(8) $\Psi; \Theta, i : S, j : S'; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash (\exists j : S'.\tau_1)[I/i] <:_{\texttt{nf}} (\exists j : S'.\tau_2)[J/i] : \star \Rightarrow \Phi' \text{ for some } \Theta; \Delta \vDash \Phi'}$

By (2)

(9) $\Theta, i : S, j : S; \Delta \vDash \Phi$

IH on (8)

(10) $\Psi; \Theta j : S'; \Delta \vdash \tau_1[I/i] <:_{\texttt{nf}} \tau_2[J//i] : \star \Rightarrow \Phi'$

(11) $\Theta, j : S'; \Delta \vDash \Phi'$

Equivalently to (11)

(12) $\Theta; \Delta \vDash \forall j : S'.\Phi'$

By AS-IExists on (10)

(13) $Psi; \Theta; \Delta \vdash \exists j : S'.\tau_1[I/i] <:_{\texttt{nf}} \exists j : S'.\tau_2[J/i] : \star \Rightarrow \forall j : S'.\Phi'$

Goal follows by (12) and (13)

▸ **Case 9:** *AS-TForall.*

▸ **Given:**

(7) $\Psi; \Theta, i : S; \Delta \vdash_p \forall \alpha : K.\tau_1 <:_{\texttt{nf}} \forall \alpha : K.\tau_2 : \star \Rightarrow \Phi$

(8) $\Psi, \alpha : K; \Theta, i : S; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash (\forall \alpha : K.\tau_1)[I/i] <:_{\texttt{nf}} (\forall \alpha : K.\tau_2)[J/i] : \star \Rightarrow \Phi' \text{ for some } \Theta; \Delta \vDash \Phi'}$

By IH on (8)

(9) $\Psi, \alpha : K; \Theta; \Delta \vdash_p \tau_1[I/i] <:_{\texttt{nf}} \tau_2[I/i] : \star \Rightarrow \Phi'$

(10) $\Theta; \Delta \vDash \Phi'$

By AS-TForall on (9)

(11) $\Psi; \Theta; \Delta \vdash_p \forall \alpha : K.\tau_1[I/i] <:_{\texttt{nf}} \forall \alpha : K.\tau_2[I/i] : \star \Rightarrow \Phi'$

Goal follows immediately from (11)

▸ **Case 10:** *AS-List.*

▸ **Given:**

(7) $\Psi; \Theta, i : S; \Delta \vdash_p L^M \tau_1 <:_{\texttt{nf}} L^N \tau_2 : \star \Rightarrow M = N \wedge \Phi'$

(8) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi'$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash (L^M \tau_1)[I/i] <:_{\texttt{nf}} (L^N \tau_2)[J/i] : \star \Rightarrow \Phi' \text{ for some } \Theta; \Delta \vDash \Phi'}$

By (2)

(9) $\Theta, i : S; \Delta \vDash M = N$

From (9) and (4)

(10) $\Theta; \Delta \vDash M[I/i] = N[J/i]$

By IH on (8)

(11) $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] <:_{\texttt{nf}} \tau_2[J/i] : \star \Rightarrow \Phi''$

(12) $\Theta; \Delta \vDash \Phi''$

By the presupposition for (7) and two inversions

(13) $\Theta, i : S; \Delta \vdash M : \mathbb{N} \Rightarrow \Phi_1$ with $\Theta, i : S; \Delta \vDash \Phi_1$

(14) $\Theta, i : S; \Delta \vdash N : \mathbb{N} \Rightarrow \Phi_2$ with $\Theta, i : S; \Delta \vDash \Phi_2$

By **??** and **??** on (13) (14) (4) (5)

(15) $\Theta; \Delta \vdash M[I/i] : \mathbb{N} \Rightarrow \Phi_1'$ with $\Theta; \Delta \vDash \Phi_1'$

(16) $\Theta; \Delta \vdash N[J/i] : \mathbb{N} \Rightarrow \Phi_2'$ with $\Theta; \Delta \vDash \Phi_1'$

By AS-List on (10) and (11)

(17) $\Psi; \Theta; \Delta \vdash L^{M[I/i]}\tau_1[I/i] <:_{\texttt{nf}} L^{N[J/i]}\tau_2[J/i] : \star \Rightarrow (M[I/i] = N[J/i]) \wedge \Phi''$

By (15), (16), (17)

(18) $\Psi; \Theta; \Delta \vdash_p L^{M[I/i]}\tau_1[I/i] <:_{\texttt{nf}} L^{N[J/i]}\tau_2[J/i] : \star \Rightarrow (M[I/i] = N[J/i]) \wedge \Phi''$

Goal follows from (18), taking $\Phi' = (M[I/i] = N[J/i]) \wedge \Phi''$

▸ **Case 11:** *AS-Conj.*

▸ **Given:**

(7) $\Psi; \Theta, i : S; \Delta \vdash_p \Phi_1 \& \tau_1 <:_{\texttt{nf}} \Phi_2 \& \tau_2 : \star \Rightarrow \Phi \wedge (Phi_1 \rightarrow \Phi_2)$

(8) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash (\Phi_1 \& \tau_1)[I/i] <:_{\texttt{nf}} (\Phi_2 \& \tau_2)[J/i] : \star \Rightarrow \Phi' \text{ for some } \Theta; \Delta \vDash \Phi'}$$

By IH on (8)

(9) $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] <:_{\texttt{nf}} \tau_2[J/i] : \star \Rightarrow \Phi'$

(10) $\Theta; \Delta \vDash \Phi'$

Inverting the presupposition that $\Phi_1 \& \tau_1$ and $\Phi_2 \& \tau_2$ are well-formed types from (1), we have

(11) $\Theta, i : S; \Delta \vdash \Phi_1 \texttt{ wf} \Rightarrow \Phi_1'$ with $\Theta, i : S; \Delta \vDash \Phi_1'$

(12) $\Theta, i : S; \Delta \vdash \Phi_2 \texttt{ wf} \Rightarrow \Phi_2'$ with $\Theta, i : S; \Delta \vDash \Phi_2'$

By Theorem A.33

(13) $\Theta; \Delta \vDash \Phi_1[I/i] \texttt{ wf} \Rightarrow \Phi_1''$ with $\Theta; \Delta \vDash \Phi_1''$

(14) $\Theta; \Delta \vDash \Phi_2[J/i] \texttt{ wf} \Rightarrow \Phi_2''$ with $\Theta; \Delta \vDash \Phi_2''$

By AS-Conj on (9)

(15) $\Psi;\Theta;\Delta \vdash \Phi_1[I/i]\&\tau_1[I/i] <:_{\texttt{nf}} \Phi_2[J/i]\&\tau_2[I/i] \Rightarrow \Phi' \wedge (\Phi_1[I/i] \to \Phi_2[J/i])$

By (11) and (12), the presuppositions for (15) hold

(16) $\Psi;\Theta;\Delta \vdash_p \Phi_1[I/i]\&\tau_1[I/i] <:_{\texttt{nf}} \Phi_2[J/i]\&\tau_2[I/i] \Rightarrow \Phi' \wedge (\Phi_1[I/i] \to \Phi_2[J/i])$

By (2)

(17) $\Theta, i : S;\Delta \vDash \Phi_1 \to \Phi_2$

By (17) and (6)

(18) $\Theta;\Delta \vDash \Phi_1[I/i] \to \Phi_2[J/i]$

The result follows by (16) and (18), with $\Phi' = \Phi' \wedge (\Phi_1[I/i] \to \Phi_2[J/i])$

▸ **Case 12:** *AS-Impl.*

▸ **Given:**

(7) $\Psi;\Theta, i : S;\Delta \vdash_p \Phi_1 \implies \tau_1 <:_{\texttt{nf}} \Phi_2 \implies \tau_2 : \star \Rightarrow (\Phi_2 \to \Phi) \wedge (Phi_2 \to \Phi_1)$

(8) $\Psi;\Theta, i : S;\Delta, \Phi_2 \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash (\Phi_1 \implies \tau_1)[I/i] <:_{\texttt{nf}} (\Phi_2\&\tau_2)[J/i] : \star \Rightarrow \Phi' \text{ for some } \Theta;\Delta \vDash \Phi'}$

By IH on (8)

(9) $\Psi;\Theta;\Delta, \Phi_2 \vdash_p \tau_1[I/i] <:_{\texttt{nf}} \tau_2[J/i] : \star \Rightarrow \Phi'$

(10) $\Theta;\Delta, \Phi_2 \vDash \Phi'$

Inverting the presupposition that $\Phi_1\&\tau_1$ and $\Phi_2\&\tau_2$ are well-formed types from (1), we have

(11) $\Theta, i : S;\Delta \vdash \Phi_1\ \texttt{wf} \Rightarrow \Phi'_1$ with $\Theta, i : S;\Delta \vDash \Phi'_1$

(12) $\Theta, i : S;\Delta \vdash \Phi_2\ \texttt{wf} \Rightarrow \Phi'_2$ with $\Theta, i : S;\Delta \vDash \Phi'_2$

By Theorem A.33

(13) $\Theta;\Delta \vDash \Phi_1[I/i]\ \texttt{wf} \Rightarrow \Phi''_1$ with $\Theta;\Delta \vDash \Phi''_1$

(14) $\Theta;\Delta \vDash \Phi_2[J/i]\ \texttt{wf} \Rightarrow \Phi''_2$ with $\Theta;\Delta \vDash \Phi''_2$

By AS-Impl on (9)

(15) $\Psi;\Theta;\Delta \vdash \Phi_1[I/i] \implies \tau_1[I/i] <:_{\texttt{nf}} \Phi_2[J/i] \implies \tau_2[I/i] \Rightarrow (\Phi_2[J/i] \to \Phi') \wedge (\Phi_2[J/i] \to \Phi_1[I/i])$

By (11) and (12), the presuppositions for (15) hold

(16) $\Psi;\Theta;\Delta \vdash_p \Phi_1[I/i] \implies \tau_1[I/i] <:_{\mathtt{nf}} \Phi_2[J/i] \implies \tau_2[I/i] \Rightarrow (\Phi_2[J/i] \rightarrow \Phi') \wedge (\Phi_2[J/i] \rightarrow \Phi_1[I/i])$

By (2)

(17) $\Theta, i : S; \Delta \vDash \Phi_2 \rightarrow \Phi_1$

By (17) and (6)

(18) $\Theta; \Delta \vDash \Phi_2[J/i] \rightarrow \Phi_1[I/i]$

The result follows by (16) and (18), with $\Phi' = (\Phi_2[J/i] \rightarrow \Phi') \wedge (\Phi_2[J/i] \rightarrow \Phi_1[I/i])$

- **Case 13:** *AS-Monad.*

  - **Given:**

    (7) $\Psi;\Theta, i : S;\Delta \vdash_p \mathbb{M}(M, \vec{q})\tau_1 <:_{\mathtt{nf}} \mathbb{M}(N, \vec{p})\tau_2 : \star \Rightarrow (M = N) \wedge (\vec{q} \leq \vec{p}) \wedge \Phi$

    (8) $\Psi;\Theta, i : S;\Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : \star \Rightarrow \Phi$

  - **Goal:**

    $\boxed{\Psi;\Theta;\Delta \vdash (\mathbb{M}(M, \vec{q})\tau_1)[I/i] <:_{\mathtt{nf}} (\mathbb{M}(N, \vec{p})\tau_2)[J/i] : \star \Rightarrow \Phi' \text{ and } \Theta;\Delta \vDash \Phi'}$

  From (2)

  (9) $\Theta, i : S; \Delta \vDash M = N$

  (10) $\Theta, i : S; \Delta \vDash \vec{q} \leq \vec{p}$

  By IH on (8)

  (11) $\Psi;\Theta;\Delta \vdash_p \tau_1[I/i] <:_{\mathtt{nf}} \tau_2[J/i] : \star \Rightarrow \Phi'$

  (12) $\Theta;\Delta \vDash \Phi'$

  By AS-Monad on (11) and Theorem A.7 on the presuppositions of (1) for $M, N, \vec{q}, \vec{p}$

  (13) $\Psi;\Theta;\Delta \vdash_p \mathbb{M}(M[I/i], \vec{q}[I/i])\ (\tau_1[I/i]) <:_{\mathtt{nf}} \mathbb{M}(N[J/i], \vec{q}[J/i])\ (\tau_2[J/i]) : \star \Rightarrow (M[I/i] = N[J/i]) \wedge (\vec{q}[I/i] \leq \vec{p}[J/i]) \wedge \Phi'$

  By (6), (9), and (10)

  (14) $\Theta;\Delta \vDash M[I/i] = N[J/i]$

  (15) $\Theta;\Delta \vDash \vec{q}[I/i] \leq \vec{p}[J/i]$

  The Goal is immediate from (12), (13), (14), (15)

- **Case 13:** *AS-Pot.*

  - **Given:**

    (7) $\Psi;\Theta, i : S;\Delta \vdash_p [M|\vec{q}]\tau_1 <:_{\mathtt{nf}} [N|\vec{p}]\tau_2 : \star \Rightarrow (M = N) \wedge (\vec{q} \geq \vec{p}) \wedge \Phi$

(8) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta \vdash ([M|\vec{q}]\tau_1)[I/i] <:_{\texttt{nf}} ([N|\vec{p}]\tau_2)[J/i] : \star \Rightarrow \Phi' \text{ and } \Theta; \Delta \vDash \Phi'}$$

From (2)

(9) $\Theta, i : S; \Delta \vDash M = N$

(10) $\Theta, i : S; \Delta \vDash \vec{q} \geq \vec{p}$

By IH on (8)

(11) $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] <:_{\texttt{nf}} \tau_2[J/i] : \star \Rightarrow \Phi'$

(12) $\Theta; \Delta \vDash \Phi'$

By AS-Pot on (11) and Theorem A.7 on the presuppositions of (1) for $M, N, \vec{q}, \vec{p}$

(13) $\Psi; \Theta; \Delta \vdash_p [M[I/i]|\vec{q}[I/i]]\ (\tau_1[I/i]) <:_{\texttt{nf}} [N[J/i]|\vec{q}[J/i]]\ (\tau_2[J/i]) : \star \Rightarrow (M[I/i] = N[J/i]) \wedge (\vec{q}[I/i] \geq \vec{p}[J/i]) \wedge \Phi'$

By (6), (9), and (10)

(14) $\Theta; \Delta \vDash M[I/i] = N[J/i]$

(15) $\Theta; \Delta \vDash \vec{q}[I/i] \geq \vec{p}[J/i]$

The Goal is immediate from (12), (13), (14), (15)

▸ **Case 14:** *AS-ConstPot.*

▸ **Given:**

(7) $\Psi; \Theta, i : S; \Delta \vdash_p [M]\ \tau_1 <:_{\texttt{nf}} [N]\ ; \tau_2 : \star \Rightarrow \Phi \wedge (N \leq M)$

(8) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\texttt{nf}} \tau_2 : \star \Rightarrow \Phi$

▸ **Goal:**

$$\boxed{\Psi; \Theta; \Delta; \vdash ([M]\ \tau_1)[I/i] <:_{\texttt{nf}} ([N]\ \tau_2)[J/i] : \star \Rightarrow \Phi' \text{ and } \Theta; \Delta \vDash \Phi'}$$

By (2)

(9) $\Theta, i : S; \Delta \vDash N \leq M$

By IH on (8)

(10) $\Psi; \Theta; \Delta \vdash_p \tau_1[I/i] <:_{\texttt{nf}} \tau_2[J/i] : \star \Rightarrow \Phi'$

(11) $\Theta; \Delta \vDash \Phi'$

By AS-ConstPot and Theorem A.7 for M, N

(12) $\Psi; \Theta; \Delta \vdash_p [M[I/i]]\ (\tau_1[I/i]) <:_{\mathtt{nf}} [N[I/i]]\ (\tau_2[J/i]) : \star \Rightarrow \Phi' \wedge (N[J/i] \leq M[I/i])$

By (9) and (6)

(13) $\Theta; \Delta \vDash N[J/i] \leq M[I/i]$

Goal follows immediately from (11), (12), (13), with $\Phi' = \Phi' \wedge (N[J/i] \leq M[I/i])$

- **Case 15:** *AS-FamLam.*

  - **Given:**

    (7) $\Psi; \Theta, i : S; \Delta \vdash_p \lambda j : S'.\tau_1 <:_{\mathtt{nf}} \lambda j : S'.\tau_2 : S \to K \Rightarrow \forall j : S.\Phi$

    (8) $\Psi; \Theta, i : S, j : S'; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : K \Rightarrow \Phi$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p (\lambda j : S'.\tau_1)[I/i] <:_{\mathtt{nf}} (\lambda j : S'.\tau_2)[J/i] : S' \to K \Rightarrow \Phi' \text{ and } \Theta; \Delta \vDash \Phi'}$

  By IH on (8)

  (9) $\Psi; \Theta, j : S'; \Delta \vdash_p \tau_1[I/i] <:_{\mathtt{nf}} \tau_2[J/i] : K \Rightarrow \Phi'$

  (10) $\Theta, j : S'; \Delta \vDash \Phi'$

  By AT-FamLam on (9)

  (11) $\Psi; \Theta; \Delta \vdash_p \forall j : S'.\tau_1[I/i] <:_{\mathtt{nf}} \forall j : S'.\tau_2[J/i] : K \Rightarrow \forall j : S'.\Phi'$

  The goal is immediate by (10) and (11)

- **Case 15:** *AS-FamApp.*

  - **Given:**

    (7) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1\ M <:_{\mathtt{nf}} \tau_2\ N : K \Rightarrow \Phi \wedge (M = N)$

    (8) $\Psi; \Theta, i : S; \Delta \vdash \tau_1 <:_{\mathtt{nf}} \tau_2 : S' \to K \Rightarrow \Phi$

  - **Goal:**

    $\boxed{\Psi; \Theta; \Delta \vdash_p (\tau_1\ M)[I/i] <:_{\mathtt{nf}} (\tau_2[J/i])[J/i] : K \Rightarrow \Phi' \text{ and } \Theta; \Delta \vDash \Phi'}$

  By (2)

  (9) $\Theta, i : S; \Delta \vDash M = N$

  By (9) and (6)

  (10) $\Theta; \Delta \vDash M[I/i] = N[J/i]$

  IH on (8)

  (11) $\Psi; \Theta; \Delta \vdash \tau_1[I/i] <:_{\mathtt{nf}} \tau_2[J/i] : S' \to K \Rightarrow \Phi'$

(12) $\Theta; \Delta \vDash \Phi'$

By AS-FamApp on (11) and **??** on the proofs that $M, N : S'$ in (7)

(13) $\Psi; \Theta; \Delta \vdash_p (\tau_1[I/i])\ M[I/i] <:_{\mathtt{nf}} (\tau_2[J/i])\ N[J/i] : K \Rightarrow \Phi' \wedge (M[I/i] = N[J/i])$

The Goal follows by (10), (12), (13)

□

Proof of Theorem 8.21

(1) By induction on $\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : K$.

▸ **Case 1:** *S-Refl.*

Immediate by Theorem 8.16.

▸ **Case 2:** *S-Trans.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_3 : K$

(2) $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K$

(3) $\Psi; \Theta; \Delta \vdash_p \tau_2 <: \tau_3 : K$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_3 : K \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : K \Rightarrow \Phi_1$

(5) $\Theta; \Delta \vDash \Phi_1$

By IH on (3)

(6) $\Psi; \Theta; \Delta \vdash_p \tau_2 <: \tau_3 : K \Rightarrow \Phi_2$

(7) $\Theta; \Delta \vDash \Phi_2$

Goal follows by **??** on (4), (5), (6), (7)

▸ **Case 3:** *S-Arr.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \tau_1 \multimap \tau_2 <: \tau_1' \multimap \tau_2' : \star$

(2) $\Psi; \Theta; \Delta \vdash_p \tau_1' <: \tau_1 : \star$

(3) $\Psi; \Theta; \Delta \vdash_p \tau_2 <: \tau_2' : \star$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash_p \tau_1 \multimap \tau_2 <: \tau_1' \multimap \tau_2' : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$$

By IH on (2)

(4) $\Psi;\Theta;\Delta \vdash_p \tau_1' <: \tau_1 : \star \Rightarrow \Phi_1$

(5) $\Theta;\Delta \vDash \Phi_1$

By IH on (3)

(6) $\Psi;\Theta;\Delta \vdash \tau_2 <: \tau_2' : \star \Rightarrow \Phi_2$

(7) $\Theta;\Delta \vDash \Phi_2$

Inverting (4) and (6)

(8) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1') <:_{\texttt{nf}} \texttt{eval}(\tau_1) : \star \Rightarrow \Phi_1$

(9) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_2) <:_{\texttt{nf}} \texttt{eval}(\tau_2') : \star \Rightarrow \Phi_2$

By AS-Arr on (8) and (9)

(10) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) \multimap \texttt{eval}(\tau_2) <:_{\texttt{nf}} \texttt{eval}(\tau_1') \multimap \texttt{eval}(\tau_2') : \star \Rightarrow \Phi_1 \wedge \Phi_2$

Goal follows by AS-Normalize on (10), taking $\Phi = \Phi_1 \wedge \Phi_2$

▸ **Case 4:** *S-Tensor.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \tau_1 \otimes \tau_2 <: \tau_1' \otimes \tau_2' : \star$

(2) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_1' : \star$

(3) $\Psi;\Theta;\Delta \vdash_p \tau_2 <: \tau_2' : \star$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash_p \tau_1 \otimes \tau_2 <: \tau_1' \otimes \tau_2' : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$$

By IH on (2)

(4) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_1' : \star \Rightarrow \Phi_1$

(5) $\Theta;\Delta \vDash \Phi_1$

By IH on (3)

(6) $\Psi;\Theta;\Delta \vdash \tau_2 <: \tau_2' : \star \Rightarrow \Phi_2$

(7) $\Theta;\Delta \vDash \Phi_2$

Inverting (4) and (6)

(8) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_1') : \star \Rightarrow \Phi_1$

(9) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_2) <:_{\texttt{nf}} \texttt{eval}(\tau_2') : \star \Rightarrow \Phi_2$

By AS-Tensor on (8) and (9)

(10) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) \otimes \texttt{eval}(\tau_2) <:_{\texttt{nf}} \texttt{eval}(\tau_1') \otimes \texttt{eval}(\tau_2') : \star \Rightarrow \Phi_1 \wedge \Phi_2$

Goal follows by AS-Normalize on (10), taking $\Phi = \Phi_1 \wedge \Phi_2$

▸ **Case 5:** *S-With.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \tau_1 \& \tau_2 <: \tau_1' \& \tau_2' : \star$

(2) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_1' : \star$

(3) $\Psi;\Theta;\Delta \vdash_p \tau_2 <: \tau_2' : \star$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \tau_1 \& \tau_2 <: \tau_1' \& \tau_2' : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_1' : \star \Rightarrow \Phi_1$

(5) $\Theta;\Delta \vDash \Phi_1$

By IH on (3)

(6) $\Psi;\Theta;\Delta \vdash \tau_2 <: \tau_2' : \star \Rightarrow \Phi_2$

(7) $\Theta;\Delta \vDash \Phi_2$

Inverting (4) and (6)

(8) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_1') : \star \Rightarrow \Phi_1$

(9) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_2) <:_{\texttt{nf}} \texttt{eval}(\tau_2') : \star \Rightarrow \Phi_2$

By AS-With on (8) and (9)

(10) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) \& \texttt{eval}(\tau_2) <:_{\texttt{nf}} \texttt{eval}(\tau_1') \& \texttt{eval}(\tau_2') : \star \Rightarrow \Phi_1 \wedge \Phi_2$

Goal follows by AS-Normalize on (10), taking $\Phi = \Phi_1 \wedge \Phi_2$

▸ **Case 6:** *S-Sum.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \tau_1 \oplus \tau_2 <: \tau_1' \oplus \tau_2' : \star$

(2) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_1' : \star$

(3) $\Psi;\Theta;\Delta \vdash_p \tau_2 <: \tau_2' : \star$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \tau_1 \oplus \tau_2 <: \tau_1' \oplus \tau_2' : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_1' : \star \Rightarrow \Phi_1$

(5) $\Theta;\Delta \vDash \Phi_1$

By IH on (3)

(6) $\Psi;\Theta;\Delta \vdash \tau_2 <: \tau_2' : \star \Rightarrow \Phi_2$

(7) $\Theta;\Delta \vDash \Phi_2$

Inverting (4) and (6)

(8) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_1') : \star \Rightarrow \Phi_1$

(9) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_2) <:_{\texttt{nf}} \texttt{eval}(\tau_2') : \star \Rightarrow \Phi_2$

By AS-Sum on (8) and (9)

(10) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) \oplus \texttt{eval}(\tau_2) <:_{\texttt{nf}} \texttt{eval}(\tau_1') \oplus \texttt{eval}(\tau_2') : \star \Rightarrow \Phi_1 \wedge \Phi_2$

Goal follows by AS-Normalize on (10), taking $\Phi = \Phi_1 \wedge \Phi_2$

▸ **Case 7:** *S-Bang.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p !\tau_1 <: !\tau_2 : \star$

(2) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_2 : \star$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p !\tau_1 <: !\tau_2 : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

(5) $\Theta;\Delta \vDash \Phi$

Inverting (5)

(6) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

By AS-Bang on (6)

(7) $\Psi;\Theta;\Delta \vdash_p !\texttt{eval}(\tau_1) <:_{\texttt{nf}} !\texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

Goal follows by AS-Normalize on (7)

▸ **Case 8:** *S-IForall.*

- **Given:**

  (1) $\Psi; \Theta; \Delta \vdash_p \forall i : S.\tau_1 <: \forall i : S.\tau_2 : \star$

  (2) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 <: \tau_2 : \star$

- **Goal:**

  $\boxed{\Psi; \Theta; \Delta \vdash_p \forall i : S.\tau_1 <: \forall i : S.\tau_2 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

(5) $\Theta, i : S; \Delta \vDash \Phi$

Inverting (5)

(6) $\Psi; \Theta, i : S; \Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

By AS-IForall on (6)

(7) $\Psi; \Theta; \Delta \vdash_p \forall i : S.\texttt{eval}(\tau_1) <:_{\texttt{nf}} \forall i : S.\texttt{eval}(\tau_2) : \star \Rightarrow \forall i : S.\Phi$

Goal follows by AS-Normalize on (7)

- **Case 9:** *S-IExists.*

- **Given:**

  (1) $\Psi; \Theta; \Delta \vdash_p \exists i : S.\tau_1 <: \exists i : S.\tau_2 : \star$

  (2) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 <: \tau_2 : \star$

- **Goal:**

  $\boxed{\Psi; \Theta; \Delta \vdash_p \exists i : S.\tau_1 <: \exists i : S.\tau_2 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

(5) $\Theta, i : S; \Delta \vDash \Phi$

Inverting (5)

(6) $\Psi; \Theta, i : S; \Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

By AS-IExists on (6)

(7) $\Psi; \Theta; \Delta \vdash_p \exists i : S.\texttt{eval}(\tau_1) <:_{\texttt{nf}} \exists i : S.\texttt{eval}(\tau_2) : \star \Rightarrow \forall i : S.\Phi$

Goal follows by AS-Normalize on (7)

- **Case 10:** *S-TForall.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \forall\alpha : K.\tau_1 <: \forall\alpha : K.\tau_2 : \star$

(2) $\Psi, \alpha : K; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : \star$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \forall\alpha : K.\tau_1 <: \forall\alpha : K.\tau_2 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi, \alpha : K; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

(5) $\Theta; \Delta \vDash \Phi$

Inverting (5)

(6) $\Psi, \alpha : K; \Theta; \Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

By AS-IForall on (6)

(7) $\Psi; \Theta; \Delta \vdash_p \forall\alpha : K.\texttt{eval}(\tau_1) <:_{\texttt{nf}} \forall\alpha : K.\texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

Goal follows by AS-Normalize on (7)

▸ **Case 11:** *S-List.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p L^I\tau_1 <: L^J\tau_2 : \star$

(2) $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : \star$

(3) $\Theta; \Delta \vDash I = J$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p L^I\tau_1 <: L^J\tau_2 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (2)

(5) $\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

(6) $\Theta; \Delta \vDash \Phi$

By (3) and (6)

(7) $\Theta; \Delta \vDash \Phi \wedge (I = J)$

By inversion on (5)

(8) $\Psi; \Theta; \Delta \vdash \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

By AS-List on (8)

(9) $\Psi; \Theta; \Delta \vdash L^I\texttt{eval}(\tau_1) <:_{\texttt{nf}} L^J\texttt{eval}(\tau_2) : \star \Rightarrow \Phi \wedge (I = J)$

Goal follows by AS-Normalize on (9)

▸ **Case 12:** *S-Impl.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \Phi_1 \implies \tau_1 <: \Phi_2 \implies \tau_2 : \star$

(2) $\Psi;\Theta;\Delta,\Phi_2 \vdash_p \tau_1 <: \tau_2 : \star$

(3) $\Theta;\Delta \vDash \Phi_2 \rightarrow \Phi_1$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \Phi_1 \implies \tau_1 <: \Phi_2 \implies \tau_2 : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi;\Theta;\Delta,\Phi_2 \vdash \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

(5) $\Theta;\Delta,\Phi_2 \vDash \Phi$

By (3) and (5)

(6) $\Theta;\Delta \vDash (\Phi_2 \rightarrow \Phi) \wedge (\Phi_2 \rightarrow \Phi_1)$

Inverting (4)

(7) $\Psi;\Theta;\Delta,\Phi_2 \vdash \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

By AS-Impl on (7)

(8) $\Psi;\Theta;\Delta \vdash \Phi_1 \implies \texttt{eval}(\tau_1) <:_{\texttt{nf}} \Phi_2 \implies \texttt{eval}(\tau_2) : \star \Rightarrow (\Phi_2 \rightarrow \Phi) \wedge (\Phi_2 \rightarrow \Phi_1)$

(Goal follows by AS-Normalize on (8))

▸ **Case 13:** *S-Conj.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p \Phi_1 \& \tau_1 <: \Phi_2 \& \tau_2 : \star$

(2) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_2 : \star$

(3) $\Theta;\Delta \vDash \Phi_1 \rightarrow \Phi_2$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p \Phi_1 \& \tau_1 <: \Phi_2 \& \tau_2 : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

(5) $\Theta;\Delta \vDash \Phi$

By (3) and (5)

(6) $\Theta; \Delta \vDash \Phi \wedge (\Phi_1 \to \Phi_2)$

Inverting (4)

(7) $\Psi; \Theta; \Delta \vdash \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

By AS-Conj on (7)

(8) $\Psi; \Theta; \Delta \vdash \Phi_1 \& \texttt{eval}(\tau_1) <:_{\texttt{nf}} \Phi_2 \& \texttt{eval}(\tau_2) : \star \Rightarrow \Phi \wedge (\Phi_1 \to \Phi_2)$

(Goal follows by AS-Normalize on (8))

▸ **Case 14:** *S-Monad.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p \mathbb{M}(I, \vec{q})\tau_1 <: \mathbb{M}(J, \vec{p})\tau_2 : \star$

(2) $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : \star$

(3) $\Theta; \Delta \vDash (I = J) \wedge (\vec{q} \leq \vec{p})$

▸ **Goal:**

$\boxed{\Psi; \Theta; \Delta \vdash_p \mathbb{M}(I, \vec{q})\tau_1 <: \mathbb{M}(J, \vec{p})\tau_2 : \star \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

(5) $\Theta; \Delta \vDash \Phi$

From (3) and (5)

(6) $\Theta; \Delta \vDash \Phi \wedge (I = J) \wedge (\vec{q} \leq \vec{p})$

By inversion on (4)

(7) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

By AS-Monad on (7)

(8) $\Psi; \Theta; \Delta \vdash_p \mathbb{M}(I, \vec{q})\texttt{eval}(\tau_1) <:_{\texttt{nf}} \mathbb{M}(J, \vec{p})\texttt{eval}(\tau_2) : \star \Rightarrow \Phi \wedge (I = J) \wedge (\vec{q} \leq \vec{p})$

Goal follows by AS-Normalize on (8)

▸ **Case 15:** *S-Pot.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta \vdash_p [I|\vec{q}]\, \tau_1 <: [J|\vec{p}]\, \tau_2 : \star$

(2) $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : \star$

(3) $\Theta; \Delta \vDash (I = J) \wedge (\vec{q} \geq \vec{p})$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p [I|\vec{q}]\,\tau_1 <: [J|\vec{p}]\,\tau_2 : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

(5) $\Theta;\Delta \vDash \Phi$

From (3) and (5)

(6) $\Theta;\Delta \vDash \Phi \wedge (I = J) \wedge (\vec{q} \geq \vec{p})$

By inversion on (4)

(7) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

By AS-Pot on (7)

(8) $\Psi;\Theta;\Delta \vdash_p [I|\vec{q}]\texttt{eval}(\tau_1) <:_{\texttt{nf}} [J|\vec{p}]\texttt{eval}(\tau_2) : \star \Rightarrow \Phi \wedge (I = J) \wedge (\vec{q} \geq \vec{p})$

Goal follows by AS-Normalize on (8)

▸ **Case 16:** *S-ConstPot.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p [I]\,\tau_1 <: [J]\,\tau_2 : \star$

(2) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_2 : \star$

(3) $\Theta;\Delta \vDash (I \geq J)$

▸ **Goal:**

$\boxed{\Psi;\Theta;\Delta \vdash_p [I]\,\tau_1 <: [J]\,\tau_2 : \star \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi;\Theta;\Delta \vdash_p \tau_1 <: \tau_2 : \star \Rightarrow \Phi$

(5) $\Theta;\Delta \vDash \Phi$

From (3) and (5)

(6) $\Theta;\Delta \vDash \Phi \wedge (I \geq J)$

By inversion on (4)

(7) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : \star \Rightarrow \Phi$

By AS-ConstPot on (7)

(8) $\Psi;\Theta;\Delta \vdash_p [I]\texttt{eval}(\tau_1) <:_{\texttt{nf}} [J]\texttt{eval}(\tau_2) : \star \Rightarrow \Phi \wedge (I \geq J)$

Goal follows by AS-Normalize on (8)

▸ **Case 17:** *S-FamLam.*

- **Given:**

  (1) $\Psi; \Theta; \Delta \vdash_p \lambda i : S.\tau_1 <: \lambda i : S.\tau_2 : S \to K$

  (2) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 <: \tau_2 : \star$

- **Goal:**

  $\boxed{\Psi; \Theta; \Delta \vdash_p \lambda i : S.\tau_1 <: \lambda i : S.\tau_2 : S \to K \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (2)

(3) $\Psi; \Theta, i : S; \Delta \vdash_p \tau_1 <: \tau_2 : K \Rightarrow \Phi$

(4) $\Theta, i : S; \Delta \vDash \Phi$

Equivalently to (4)

(5) $Theta; Delta \vDash \forall i : S.\Phi$

By inversion on (3)

(6) $\Psi; \Theta, i : S; \Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : K \Rightarrow \Phi$

By AS-FamLam on (6)

(7) $\Psi; \Theta; \Delta \vdash_p \lambda i : S.\texttt{eval}(\tau_1) <:_{\texttt{nf}} \lambda i : S.\texttt{eval}(\tau_2) : S \to K \Rightarrow \forall i : S.\Phi$

Goal follows by AS-Noramlize on (7)

- **Case 18:** *S-FamApp.*

- **Given:**

  (1) $\Psi; \Theta; \Delta \vdash_p \tau_1\ I <: \tau_2\ J : K$

  (2) $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : S \to K$

  (3) $\Theta; \Delta \vDash I = J$

- **Goal:**

  $\boxed{\Psi; \Theta; \Delta \vdash_p \tau_1\ I <: \tau_2\ J : K \Rightarrow \Phi \text{ and } \Theta; \Delta \vDash \Phi}$

By IH on (2)

(4) $\Psi; \Theta; \Delta \vdash_p \tau_1 <: \tau_2 : S \to K \Rightarrow \Phi$

(5) $\Theta; \Delta \vDash \Phi$

Inverting (4)

(6) $\Psi; \Theta; \Delta \vdash_p \texttt{eval}(\tau_1) <:_{\texttt{nf}} \texttt{eval}(\tau_2) : S \to K \Rightarrow \Phi$

By Theorem 8.20 on (3) and (6)

(7) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau_1\ I) <:_{\texttt{nf}} \texttt{eval}(\tau_2\ J) : K \Rightarrow \Phi'$

(8) $\Theta;\Delta \vDash \Phi'$

Goal follows from AS-Normalize on (7)

▸ **Case 19:** *S-Fam-Beta1.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta \vdash_p (\lambda i : S.\tau)\ J <: \tau[J/i] : K$

▸ **Goal:**

$$\boxed{\Psi;\Theta;\Delta \vdash (\lambda i : S.\tau)\ J <: \tau[J/i] : K \Rightarrow \Phi \text{ and } \Theta;\Delta \vDash \Phi}$$

By the presupposition for (1)

(2) $\Psi;\Theta;\Delta \vdash_p \tau[J/i] : K$

By Theorem 7.3 on (2)

(3) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau[J/i]) : K$

(4) $\texttt{eval}(\tau[J/i])\ \texttt{nf}$

By Theorem 8.15 on (3) and (4)

(5) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}(\tau[J/i]) <:_{\texttt{nf}} \texttt{eval}(\tau[J/i]) : K \Rightarrow \Phi$

(6) $\Theta;\Delta \vDash \Phi$

By Theorem 7.4

(7) $\texttt{eval}(\tau[J/i]) = \texttt{eval}(\tau)[J/i]$

By definition of `eval`

(8) $\texttt{eval}(\tau)[J/i] = \texttt{eval}((\lambda i : S.\tau)\ J)$

Combining (7) and (8) in (5)

(9) $\Psi;\Theta;\Delta \vdash_p \texttt{eval}((\lambda i : S.\tau)\ J) <:_{\texttt{nf}} \texttt{eval}(\tau[J/i]) : K \Rightarrow \Phi$

Goal follows by AS-Normalize on (9)

▸ **Case 20:** *S-Fam-Beta2.*

Identical to case 19

□

Proof. By a mutual induction on the premises of both cases. We will write this as a case analysis over the AT-* rules. We will use Theorem 5.6 liberally, sometimes silently.

(AT-Var-1) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash x \uparrow \tau \Rightarrow \top, \Gamma \setminus \{x:\tau\}$ from $x:\tau \in \Gamma$. By Theorem 5.6, there is some τ' so that $\Psi;\Theta;\Delta \vdash \tau' <: \tau : \star$, and $x:\tau' \in \Gamma'$. By AT-Var-1, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash x \uparrow \tau' \Rightarrow \top, \Gamma' \setminus \{x:\tau'\}$. By Theorem 8.21, $\Psi;\Theta;\Delta \vdash \tau' <: \tau : \star \Rightarrow \Phi$ with $\Theta;\Delta \vDash \Phi$, and so by AT-Sub, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash x \downarrow \tau \Rightarrow \top \wedge \Phi, \Gamma' \setminus \{x:\tau'\}$. Finally, $\Psi;\Theta;\Delta \vdash \Gamma' \setminus \{x:\tau'\} \sqsubseteq \Gamma' \setminus \Gamma$ and $\Psi;\Theta;\Delta \vdash \Gamma' \setminus \{x:\tau'\} \sqsubseteq \Gamma \setminus \{x:\tau\}$ since $x:\tau \in \Gamma$, and $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, which proves (1). For (2), one use of AT-Anno gives $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash (x:\tau) \uparrow \tau \Rightarrow \top \wedge \Phi, \Gamma' \setminus \{x:\tau'\}$. But, $|(x:\tau)| = x$, and so we are done.

(AT-Var-2) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash x \uparrow \tau \Rightarrow \top, \Gamma$ from $x:\tau \in \Omega$, with $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$ with $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$. By Theorem 5.6, there is some τ' so that $x:\tau' \in \Omega'$ and $\Phi;\Theta;\Delta \vdash \tau' <: \tau : \star$. By Theorem 8.21, $\Phi;\Theta;\Delta \vdash \tau' <: \tau : \star \Rightarrow \Phi$ with $\Theta;\Delta \vDash \Phi$. By AT-Var-2, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash x \uparrow \tau' \Rightarrow \top, \Gamma'$. By AT-Sub, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash x \downarrow \tau \Rightarrow \top \wedge \Phi, \Gamma'$, which proves (2). For (1), we apply AT-Anno and note that $|(x:\tau)| = x$.

(AT-Unit) Immediate.

(AT-Base) Immediate.

(AT-Absurd) Immediate.

(AT-Nil) Immediate.

(AT-Cons) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 :: e_2 \downarrow L^I\tau \Rightarrow (I \geq 1) \wedge \Phi_1 \wedge \Phi_2, \Gamma_2$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \downarrow \tau \Rightarrow \Phi_1, \Gamma_1$ and $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_2 \downarrow L^{I-1}\tau \Rightarrow \Phi_2, \Gamma_2$ with $\Theta;\Delta \vDash (I \geq 1) \wedge \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vDash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e_1', Φ_1', Γ_1' such that $|e_1'| = |e_1|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1' \downarrow \tau \Rightarrow \Phi_1', \Gamma_1'$. By IH, there are e_2', Φ_2', Γ_2' such that $|e_2'| = |e_2|$, $\Theta;\Delta \vDash \Phi_2'$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_1' \setminus \Gamma_1$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$, and $\Psi;\Theta;\Delta;\Omega';\Gamma_1' \vdash e_2' \downarrow L^{I-1}\tau \Rightarrow \Phi_2', \Gamma_2'$. By AT-Cons, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1' :: e_2' \downarrow L^I\tau \Rightarrow (I \geq 1) \wedge \Phi_1' \wedge \Phi_2', \Gamma_2'$. Since $\Theta;\Delta \vDash I \geq 1$, we have that $\Theta;\Delta \vDash (I \geq 1) \wedge \Phi_1' \wedge \Phi_2'$. Further, $|e_1' :: e_2'| = |e_1'| :: |e_2'| = |e_1| :: |e_2| = |e_1 :: e_2|$. Finally, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$ by the second IH, and $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma' \setminus \Gamma$ by $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_1' \setminus \Gamma_1$ and $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$ using Theorem 5.6, which completes (1). For (2), AT-Anno gives $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash (e_1' :: e_2' : L^I) \uparrow L^I\tau \Rightarrow (I \geq 1) \wedge \Phi_1' \wedge \Phi_2', \Gamma_2'$, as required.

(AT-Match) Suppose

$$\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \mathtt{match}(e, e_1, h.t.e_2) \downarrow \tau' \Rightarrow \Phi_1 \wedge (I = 0 \to \Phi_2) \wedge (I \geq 1 \to \Phi_3), \Gamma_2 \cap (\Gamma_3 \setminus \{h,t\}$$

from

$$\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow L^I\tau \Rightarrow \Phi_1, \Gamma_1$$

,

$$\Psi;\Theta;\Delta, I = 0;\Omega;\Gamma_1 \vdash e_1 \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2$$

$$\Psi;\Theta;\Delta, I \geq 1;\Omega;\Gamma_1, h : \tau, t : L^{I-1}\tau \vdash e_2 \downarrow \tau' \Rightarrow \Phi_3, \Gamma_3$$

with

$$\Theta;\Delta \vDash \Phi_1 \wedge (I = 0 \to \Phi_2) \wedge (I \geq 1 \to \Phi_3)$$

$$\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$$

$$\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$$

By IH, there are e', Φ_1', Γ_1' with $|e'| = |e|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \uparrow L^I\tau \Rightarrow \Phi_1', \Gamma_1'$. By IH, there are e_1', Φ_2', Γ_2' with $|e_1'| = |e_1|$, $\Theta;\Delta, I = 0 \vDash \Phi_2'$, $\Psi;\Theta;\Delta, I = 0 \vdash \Gamma_2' \sqsubseteq \Gamma_1' \setminus \Gamma_1$, $\Psi;\Theta;\Delta, I = 0 \vdash \Gamma_2' \sqsubseteq \Gamma_2$, and $\Psi;\Theta;\Delta;\Omega';\Gamma_1' \vdash e_1' \downarrow \tau' \Rightarrow \Phi_2', \Gamma_2'$. Since $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, we have that $\Psi;\Theta;\Delta \vdash \Gamma_1', h : \tau, t : L^{I-1}\tau \sqsubseteq \Gamma_1, h : \tau, t : L^{I-1}\tau$, and so by IH, there are e_2', Φ_3', Γ_3' such that $|e_2'| = |e_2|$, $\Theta;\Delta, I \geq 1 \vDash \Phi_3'$, $\Psi;\Theta;\Delta, I \geq 1 \vdash \Gamma_3' \sqsubseteq (\Gamma_1', h : \tau, t : L^{I-1}\tau) \setminus (\Gamma_1, h : \tau, t : L^{I-1}\tau)$, $\Psi;\Theta;\Delta, I \geq 1 \vdash \Gamma_3' \sqsubseteq \Gamma_3$, and $\Psi;\Theta;\Delta;\Omega';\Gamma_1', h : \tau, t : L^{I-1}\tau \vdash e_2' \downarrow \tau' \Rightarrow \Phi_3', \Gamma_3'$. By AT-Match, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{match}(e', e1', h.t.e_2') \downarrow \tau' \Rightarrow \Phi_1' \wedge (I = 0 \to \Phi_2') \wedge (I \geq 1 \to \Phi_3'), \Gamma_2' \cap (\Gamma_3' \setminus \{h, t\})$ Firstly, we note that $|\texttt{match}(e', e1', h.t.e_2')| = \texttt{match}(|e'|, |e_1'|, h.t.|e_2'|) = \texttt{match}(|e|, |e_1|, h.t.|e_2|) = \texttt{match}(e, e_1, h.t.e_2)$. Then, since $\Theta;\Delta \vDash I = 0 \to \Phi_2'$ and $\Theta;\Delta \vDash I \geq 1 \to \Phi_3'$, we have $\Theta;\Delta \vDash \Phi_1' \wedge (I = 0 \to \Phi_2') \wedge (I \geq 1 \to \Phi_3')$. Then, $\Phi;\Theta;\Delta \vdash \Gamma_2' \cap (\Gamma_3' \setminus \{h, t\}) \sqsubseteq \Gamma' \setminus \Gamma$ because of $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_1' \setminus \Gamma_1$, and $\Psi;\Theta;\Delta, I \geq 1 \vdash \Gamma_3' \sqsubseteq \Gamma_1' \setminus \Gamma_1$, making heavy use of Theorem 5.6. Finally, we have $\Phi;\Theta;\Delta \vdash \Gamma_2' \cap (\Gamma_3' \setminus \{h, t\}) \sqsubseteq \Gamma_2 \cap (\Gamma_3 \setminus \{h, t\}$ since $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$ and $\Psi;\Theta;\Delta \vdash \Gamma_3' \sqsubseteq \Gamma_3$. We may strengthen away the assumptions in Δ because of the presuppositions of well-formedness all contexts involved. This completes (1). For (2), we apply AT-Anno to get $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash (\texttt{match}(e', e1', h.t.e_2') : \tau') \uparrow \tau' \Rightarrow \Phi_1' \wedge (I = 0 \to \Phi_2') \wedge (I \geq 1 \to \Phi_3'), \Gamma_2' \cap (\Gamma_3' \setminus \{h, t\})$, and we are done.

(AT-ExistI) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{pack}[I](e) \downarrow \exists i : S.\tau \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma''$ from $\Theta;\Delta \vdash I : S \Rightarrow \Phi_1$ and $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau[I/i] \Rightarrow \Phi_2, \Gamma''$, with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ_2', Γ''' such that $|e'| = |e|$, $\Theta;\Delta \vDash \Phi_2'$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, and $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \downarrow \tau[I/i] \Rightarrow \Phi_2', \Gamma'''$. By AT-ExistI, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{pack}[I](e') \downarrow \exists i : S.\tau[I/i] \Rightarrow \Phi_1 \wedge \Phi_2', \Gamma'''$. Since $|\texttt{pack}[I](e')| = \texttt{pack}[I](|e'|) = \texttt{pack}[I](|e|) = |\texttt{pack}[I](e)|$ and $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2'$, this completes (1). For (2), one application of AT-Anno suffices.

(AT-ExistE) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{unpack}\ (i,x) = e_1\ \texttt{in}\ e_2 \downarrow \tau' \Rightarrow \Phi_1 \wedge (\forall i : S.\Phi_2), \Gamma_2 \setminus \{x : \tau\}$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \uparrow \exists i : S.\tau \Rightarrow \Phi_1, \Gamma_1$ and $\Psi;\Theta,i : S;\Delta;\Omega;\Gamma_1, x : \tau \vdash e_2 \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2$ with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e_1', Φ_1', Γ_1' such that $|e_1'| = |e_1|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1' \uparrow \exists i : S.\tau \Rightarrow \Phi_1', \Gamma_1'$. Since $\Theta;\Delta \vDash \Phi_1 \wedge (\forall i : S.\Phi_2)$, we have $\Theta, i : S;\Delta \vDash \Phi_2$. We also have $\Psi;\Theta;\Delta \vdash \Gamma_1', x : \tau \sqsubseteq \Gamma_1, x : \tau$. From these two facts we have by IH that there are e_2', Φ_2', Γ_2' such that $|e_2'| = |e_2|$, $\Theta, i : S;\Delta \vDash \Phi_2'$, $\Psi;\Theta, i : S;\Delta \vdash \Gamma_2' \sqsubseteq (\Gamma_1', x : \tau) \setminus (\Gamma_1, x : \tau)$, $\Psi;\Theta, i : S;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$, and that $\Psi;\Theta, i : S;\Delta;\Omega';\Gamma_1', x : \tau \vdash e_2' \downarrow \tau' \Rightarrow \Phi_2', \Gamma_2'$. By AT-ExistE, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{unpack}\ (i,x) = e_1'\ \texttt{in}\ e_2' \downarrow \tau' \Rightarrow \Phi_1' \wedge (\forall i : S.\Phi_2'), \Gamma_2'$. Since $\Theta;\Delta \vDash \Phi_1'$ and $\Theta, i : S;\Delta \vDash \Phi_2'$, we have $\Theta;\Delta \vDash \Phi_1' \wedge (\forall i : S.\Phi_2')$. Next, we note that $|\texttt{unpack}\ (i,x) = e_1'\ \texttt{in}\ e_2'| = \texttt{unpack}\ (i,x) = |e_1'|\ \texttt{in}\ |e_2'| = \texttt{unpack}\ (i,x) = |e_1|\ \texttt{in}\ |e_2| = |\texttt{unpack}\ (i,x) = e_1\ \texttt{in}\ e_2|$. $\Psi;\Theta, i : S;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$ is immediate from the second IH, and the fact that $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma' \setminus \Gamma$ follows from Theorem 5.6. This completes (1), and (2) follows immediately from AT-Anno.

(AT-Lam) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \lambda x.e \downarrow \tau_1 \multimap \tau_2 \Rightarrow \Phi, \Gamma'' \setminus \{x : \tau_1\}$ from $\Psi;\Theta;\Delta;\Omega;\Gamma, x : \tau_1 \vdash e \downarrow \tau_2, \Rightarrow \Phi, \Gamma''$, with $\Theta;\Delta \vDash \Phi$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. Since $\Psi;\Theta;\Delta \vdash \Gamma', x : \tau_1 \sqsubseteq \Gamma, x : \tau_1$, by IH there are Φ', e', Γ''' so that $\Theta;\Delta \vDash \Phi'$, $|e'| = |e|$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma \setminus \Gamma'$, and $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and

$$\Psi;\Theta;\Delta;\Omega';\Gamma', x : \tau_1 \vdash e' : \tau_2 \Rightarrow \Phi', \Gamma'''$$

. By AT-Lam, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \lambda x.e' : \tau_1 \multimap \tau_2 \Rightarrow \Phi', \Gamma''' \setminus \{x : \tau_1\}$. Then, $|\lambda x.e'| = \lambda x.|e'| = \lambda x.|e| = |\lambda x.e|$. Finally, $\Psi;\Theta;\Delta \vdash \Gamma''' \setminus \{x : \tau_1\} \sqsubseteq \Gamma' \setminus \Gamma$ and $\Psi;\Theta;\Delta \vdash \Gamma''' \setminus \{x : \tau_1\} \sqsubseteq \Gamma'' \setminus \{x : \tau_1\}$, which proves (1). For (2), one use of AT-Anno suffices.

(AT-App) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1\, e_2 \uparrow \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \uparrow \tau_1 \multimap \tau_2 \Rightarrow \Phi_1, \Gamma_1$ and $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_2 \downarrow \tau_1 \Rightarrow \Phi_2, \Gamma_2$, with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$, and $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$. By IH, there are e_1', Φ_1', and Γ_1' such that $|e_1'| = |e_1|$, $\Theta;\Delta \vDash \Phi_1'$,

$\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1' \uparrow \tau_1 \multimap \tau_2 \Rightarrow \Phi_1', \Gamma_1'$. Since $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, we have by IH that there are e_2', Φ_2', Γ_2' such that $|e_2'| = |e_2|$, $\Theta;\Delta \vDash \Phi_2'$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_1' \setminus \Gamma_1$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$, and $\Psi;\Theta;\Delta;\Omega';\Gamma_1' \vdash e_2' \downarrow \tau_1 \Rightarrow \Phi_2', \Gamma_2'$. By AT-App, we have $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1'\ e_2' \uparrow \tau_2 \Rightarrow \Phi_1' \wedge \Phi_2', \Gamma_2'$. This completes (2). For (1), we invoke Theorem 8.16 to get that $\Psi;\Theta;\Delta \vdash \tau_2 <: \tau_2 : \star \Rightarrow \Phi'$ with $\Theta;\Delta \vDash \Phi'$. By AT-Sub, we have $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1'\ e_2' \uparrow \tau_2 \Rightarrow \Phi_1' \wedge \Phi_2' \wedge \Phi', \Gamma_2'$, completing (1).

(AT-TensorI) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \langle\!\langle e_1, e_2 \rangle\!\rangle \downarrow \tau_1 \otimes \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \downarrow \tau_1 \Rightarrow \Phi_1, \Gamma_1$ and $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_2 \downarrow \tau_2 \Rightarrow \Phi_2, \Gamma_2$, with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are Φ_1', e_1', and Γ_1' such that $\Theta;\Delta \vDash \Phi_1'$, $|e_1'| = |e_1|$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$,

$$\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1' : \tau_1 \Rightarrow \Phi_1', \Gamma_1'$$

. Since $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, we also have by IH that there are Φ_2', e_2', Γ_2' such that $\Theta;\Delta \vDash \Phi_2'$, $|e_2'| = |e_2|$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_1' \setminus \Gamma_1$, and $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$ such that

$$\Psi;\Theta;\Delta;\Omega';\Gamma_1' \vdash e_2' : \tau_2 \Rightarrow \Phi_2', \Gamma_2'$$

. By AT-TensorI, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \langle\!\langle e_1', e_2' \rangle\!\rangle : \tau_1 \otimes \tau_2 \Rightarrow (\Phi_1' \wedge \Phi_2'), \Gamma_2'$. We have that $\Theta;\Delta \vDash \Phi_1' \wedge \Phi_2'$ and $|\langle\!\langle e_1', e_2' \rangle\!\rangle| = \langle\!\langle |e_1'|, |e_2'| \rangle\!\rangle = \langle\!\langle |e_1|, |e_2| \rangle\!\rangle = |\langle\!\langle e_1, e_2 \rangle\!\rangle|$. Finally, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma' \setminus \Gamma$ since $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_1' \setminus \Gamma_1$ and $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$ by Theorem 5.6, completing (1). For (2), a single use of AT-Anno gives $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash (\langle\!\langle e_1', e_2' \rangle\!\rangle : \tau_1 \otimes \tau_2) : \tau_1 \otimes \tau_2 \uparrow (\Phi_1' \wedge \Phi_2'), \Gamma_2'$, as required.

(AT-TensorE) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{let}\ \langle\!\langle x, y \rangle\!\rangle = e_1\ \texttt{in}\ e_2 \downarrow \tau' \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2 \setminus \{x, y\}$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \uparrow \tau_1 \otimes \tau_2 \Rightarrow \Phi_1, \Gamma_1$ and $\Psi;\Theta;\Delta;\Omega;\Gamma_1, x : \tau_1, y : \tau_2 \vdash e_2 \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2$, with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$. By IH, there are e_1', Φ_1', Γ_1' such that $|e_1'| = |e_1|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1' \uparrow \tau_1 \otimes \tau_2 \Rightarrow \Phi_1', \Gamma_1'$. Then, since $\Psi;\Theta;\Delta \vdash \Gamma_1', x : \tau_1, y : \tau_2 \sqsubseteq \Gamma_1, x : \tau_1, y : \tau_2$, we have by IH that there are e_2', Φ_2', Γ_2' such that $|e_2'| = |e_2|$, $\Theta;\Delta \vDash \Phi_2'$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq (\Gamma_1', x : \tau_1, y : \tau_2) \setminus (\Gamma_1, x : \tau_1, y : \tau_2)$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$, and $\Psi;\Theta;\Delta;\Omega';\Gamma_1', x : \tau_1, y : \tau_2 \vdash e_2' \downarrow \tau_2 \Rightarrow \Phi_2', \Gamma_2'$. By AT-TensorE, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{let}\ \langle\!\langle x, y \rangle\!\rangle = e_1'\ \texttt{in}\ e_2' \downarrow \tau' \Rightarrow \Phi_1' \wedge \Phi_2', \Gamma_2' \setminus \{x, y\}$. Of course, $|\texttt{let}\ \langle\!\langle x, y \rangle\!\rangle = e_1'\ \texttt{in}\ e_2'| =$

$|\texttt{let}\ \langle\!\langle x,y\rangle\!\rangle = e_1\ \texttt{in}\ e_2|$ and $\Theta;\Delta \vDash \Phi_1' \wedge \Phi_2'$ as usual, and $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$ is immediate by IH. The fact that $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma' \setminus \Gamma$ follows from considering the weakening judgments from both IHs, and liberally applying Theorem 5.6. This completes (1). For (2), we simply apply AT-Anno, and are done.

(AT-WithI) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash (e_1,e_2) \downarrow \tau_1 \& \tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_1 \sqcap \Gamma_2$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \downarrow \tau_1 \Rightarrow \Phi_1, \Gamma_1$ and $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_2 \downarrow \tau_2 \Rightarrow \Phi_2, \Gamma_2$ with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$, and $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$. By IH, there are e_1', Φ_1', Γ_1' such that $|e_1'| = |e_1|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1' \downarrow \tau_1 \Rightarrow \Phi_1', \Gamma_1'$. Again by IH, there are e_2', Φ_2', Γ_2' such that $|e_2'| = |e_2|$, $\Theta;\Delta \vDash \Phi_2'$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_2' \downarrow \tau_2 \Rightarrow \Phi_2', \Gamma_2'$. Then, by AT-WithI, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash (e_1',e_2') \downarrow \tau_1 \& \tau_2 \Rightarrow \Phi_1' \wedge \Phi_2', \Gamma_1' \sqcap \Gamma_2'$. Then, by Theorem 5.6, we have that $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqcap \Gamma_2' \sqsubseteq \Gamma_1 \sqcap \Gamma_2$, and that $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqcap \Gamma_2' \sqsubseteq \Gamma' \setminus \Gamma$. This completes (1), and (2) follows by AT-Anno.

(AT-Fst) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{fst}(e) \uparrow \tau_1 \Rightarrow \Phi, \Gamma''$ by way of $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \tau_1 \& \tau_2 \Rightarrow \Phi, \Gamma''$ with $\Theta;\Delta \vDash \Phi$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, we have e', Φ', Γ''' such that $|e'| = |e|$, $\Theta;\Delta \vDash \Phi'$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \uparrow \tau_1 \& \tau_2 \Rightarrow \Phi', \Gamma'''$. By AT-Fst, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{fst}(e') \uparrow \tau_1 \Rightarrow \Phi', \Gamma'''$, which completes (2), since $|\texttt{fst}(e')| = \texttt{fst}(|e'|) = \texttt{fst}(|e|) = |\texttt{fst}(e)|$. For (1), by Theorem 8.16, there is Φ'' such that $\Psi;\Theta;\Delta \vdash \tau_1 <: \tau_1 : \star \Rightarrow \Phi''$ with $\Theta;\Delta \vDash \Phi''$. By AT-Sub, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{fst}(e') \downarrow \tau_1 \Rightarrow \Phi', \Gamma'''$, completing (1).

(AT-Snd) Identical to AT-Fst.

(AT-Inl) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{inl}(e) \downarrow \tau_1 \oplus \tau_2 \Rightarrow \Phi, \Gamma''$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau_1 \Rightarrow \Phi, \Gamma''$, and $\Theta;\Delta \vDash \Phi$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ', Γ''' such that $|e'| = |e|$, $\Theta;\Delta \vDash \Phi'$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \downarrow \tau_1 \Rightarrow \Phi', \Gamma'''$. By AT-Inl, we have $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{inl}(e') \downarrow \tau_1 \oplus \tau_2 \Rightarrow \Phi', \Phi'''$, which completes (1), and (2) is done by AT-Anno.

(AT-Inr) Identical to AT-Inl.

(AT-Case) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{case}(e, x.e_1, y.e_2) \downarrow \tau \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3, (\Gamma_2 \setminus \{x : \tau_1\}) \sqcap (\Gamma_3 \setminus \{y : \tau_2\})$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \tau_1 \oplus \tau_2 \Rightarrow \Phi_1, \Gamma_1$, $\Psi;\Theta;\Delta;\Omega;\Gamma_1, x : \tau_1 \vdash e_1 \downarrow \tau \Rightarrow \Phi_2, \Gamma_2$, and $\Psi;\Theta;\Delta;\Omega;\Gamma_1, y : \tau_2 \vdash e_2 \downarrow \tau \Rightarrow \Phi_3, \Gamma_3$, and also that $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2 \wedge \Phi_3$,

$\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, we have e', Φ_1', Γ_1' such that $|e'| = |e|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \uparrow \tau_1 \oplus \tau_2 \Rightarrow \Phi_1', \Gamma_1'$. Then, since $\Psi;\Theta;\Delta \vdash \Gamma_1', x : \tau_1 \sqsubseteq \Gamma_1, x : \tau_1$, we have by IH e_1'. Φ_2'. Γ_2' such that $|e_1'| = |e_1|$, $\Theta;\Delta \vDash \Phi_2'$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_1' \setminus \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega';\Gamma', x : \tau_1 \vdash e_1' \downarrow \tau \Rightarrow \Phi_2', \Gamma_2'$. Similarly, since $\Psi;\Theta;\Delta \vdash \Gamma_1', y : \tau_2 \sqsubseteq \Gamma_1, y : \tau_2$, we have by IH e_2'. Φ_3'. Γ_3' such that $|e_2'| = |e_2|$, $\Theta;\Delta \vDash \Phi_3'$, $\Psi;\Theta;\Delta \vdash \Gamma_3' \sqsubseteq \Gamma_3$, $\Psi;\Theta;\Delta \vdash \Gamma_3' \sqsubseteq \Gamma_1' \setminus \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega';\Gamma', x : \tau_2 \vdash e_2' \downarrow \tau \Rightarrow \Phi_3', \Gamma_3'$. Then, by AT-Case, we have $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{case}(e', x.e_1', y.e_2') \downarrow \tau \Rightarrow \Phi_1' \wedge \Phi_2' \wedge \Phi_3', (\Gamma_2' \setminus \{x : \tau_1\}) \cap (\Gamma_3' \setminus \{y : \tau_2\})$. Of course, $|\texttt{case}(e', x.e_1', y.e_2')| = |\texttt{case}(e, x.e_1, y.e_2)|$ since $|e'| = |e|$, $|e_1'| = |e_1|$, and $|e_2'| = |e_2|$. Similarly, $\Theta;\Delta \vDash \Phi_1' \wedge \Phi_2' \wedge \Phi_3'$. It remains to show that $\Psi;\Theta;\Delta \vdash (\Gamma_2' \setminus \{x : \tau_1\}) \cap (\Gamma_3' \setminus \{y : \tau_2\}) \sqsubseteq (\Gamma_2 \setminus \{x : \tau_1\}) \cap (\Gamma_3 \setminus \{y : \tau_2\})$ and $\Psi;\Theta;\Delta \vdash (\Gamma_2' \setminus \{x : \tau_1\}) \cap (\Gamma_3' \setminus \{y : \tau_2\}) \sqsubseteq \Gamma' \setminus \Gamma$, but both follow from the six weakening judgments and a few applications of Theorem 5.6. This completes (1), and (2) follows immediately by AT-Anno.

(AT-Sub) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma''$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \tau' \Rightarrow \Phi_1, \Gamma''$ and $\Psi;\Theta;\Delta \vdash \tau' <: \tau : \star \Rightarrow \Phi_2$, and $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e',Φ_1', and Γ''' such that $|e| = |e'|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vDash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vDash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega'\Gamma' \vdash e' \uparrow \tau' \Rightarrow \Phi_1', \Gamma'''$. by AT-Sub, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \downarrow \tau \Rightarrow \Phi_1' \wedge \Phi_2, \Gamma'''$, which proves (1). For (2), we again use AT-Anno.

(AT-ExpI) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash !e \downarrow !\tau \Rightarrow \Phi, \Gamma$ from $\Psi;\Theta;\Delta;\Omega;\cdot \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma''$, and $\Theta;\Delta \vDash \Phi$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH (not weakening the empty affine environment) there are e', Φ', Γ''' such that $|e'| = |e|$, $\Theta;\Delta \vDash \Phi'$, and $\Psi;\Theta;\Delta;\Omega';\cdot \vdash e' \downarrow \tau \Rightarrow \Phi', \Gamma'''$. By AT-ExpI, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash !e' \downarrow \tau \Rightarrow \Phi', \Gamma'$. This completes (1), since $|!e'| = |!e|$, $\Theta;\Delta \vDash \Phi'$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma'$ and $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma' \setminus \Gamma$ as consequences of Theorem 5.6.

(AT-ExpE) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{let } !x = e_1 \texttt{ in } e_2 \downarrow \tau' \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \uparrow !\tau \Rightarrow \Phi_1, \Gamma_1$ and $\Psi;\Theta;\Delta;\Omega, x : \tau;\Gamma_1 \vdash e_2 \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2$, with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e_1', Φ_1', Γ_1' such that $|e_1'| = |e_1|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma'$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1' \uparrow !\tau \Rightarrow \Phi_1', \Gamma_1'$. Since $\Psi;\Theta;\Delta \vdash \Omega', x : \tau \sqsubseteq \Omega, x : \tau$, we have by IH we have e_2', Φ_2', Γ_2' such that $|e_2'| = |e_2|$,

$\Theta;\Delta \vDash \Phi'_2$, $\Psi;\Theta;\Delta \vdash \Gamma'_2 \sqsubseteq \Gamma'_1 \setminus \Gamma_1$, $\Psi;\Theta;\Delta \vdash \Gamma'_2 \sqsubseteq \Gamma'_2$, and $\Psi;\Theta;\Delta;\Omega',x:\tau;\Gamma'_1 \vdash e'_2 \downarrow \tau' \Rightarrow \Phi'_2,\Gamma'_2$. By AT-ExpE, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{let } !x = e'_1 \texttt{ in } e'_2 \downarrow \tau' \Rightarrow \Phi'_1 \wedge \Phi'_2,\Gamma'_2$. Applying Theorem 5.6 to the inductive hypotheses gives us that $\Psi;\Theta;\Delta \vdash \Gamma'_2 \sqsubseteq \Gamma' \setminus \Gamma$, completing (1). For (2), one application of AT-Anno suffices.

(AT-TAbs) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \Lambda\alpha.e \downarrow \forall\alpha : K.\tau \Rightarrow \Phi,\Gamma''$ from $\Psi,\alpha : K;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi,\Gamma''$, and $\Theta;\Delta \vDash \Phi$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, we have e', Φ', Γ''' such that $|e'| = |e|$, $\Theta;\Delta \vdash \Phi'$, $\Psi,\alpha : K;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$ $\Psi,\alpha : K;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi,\alpha : K;\Theta;\Delta;\Omega';\Gamma' \vdash e' \downarrow \tau \Rightarrow \Phi',\Gamma'''$. By AT-TAbs, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \downarrow \forall\alpha : K.\tau \Rightarrow \Phi',\Gamma'''$. By Theorem 4, we have that $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$ and $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, which completes (1). For (2), one use of AT-Anno suffices.

(AT-TApp) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e[\tau'] \uparrow \tau[\tau'/\alpha] \Rightarrow \Phi_1 \wedge \Phi_2,\Gamma''$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \forall\alpha : K.\tau \Rightarrow \Phi_1,\Gamma''$ and $\Psi;\Theta;\Delta \vdash \tau' : K \Rightarrow \Phi_2$, with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ'_1, Γ''' such that $|e'| = |e'|$, $\Theta;\Delta \vDash \Phi'_1$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \uparrow \forall\alpha : K.\tau \Rightarrow \Phi'_1,\Gamma'''$. By AT-TApp, $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e'[\tau'] \uparrow \tau[\tau'/\alpha] \Rightarrow \Phi'_1 \wedge \Phi_2,\Gamma'''$. Since $|e'[\tau']| = |e'|[\tau'] = |e[\tau']|$ and $\Theta;\Delta \vDash \Phi'_1 \wedge \Phi_2$, we are done with (1). For (2), a single use of AT-Anno completes the proof.

(AT-IAbs) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \Lambda i.e \downarrow \forall i : S.\tau \Rightarrow \forall i : S.\Phi,\Gamma''$ from $\Psi;\Theta,i : S;\Delta;\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi,\Gamma''$, with $\Theta;\Delta \vDash \Phi$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By Theorem A.14, $\Psi;\Theta,i : S;\Delta \vdash \Gamma' \sqsubseteq \Gamma$ and $\Psi;\Theta,i : S;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ', Γ''' such that $|e'| = |e|$, $\Theta,i : S;\Delta \vDash \Phi'$, $\Psi;\Theta,i : S;\Delta \vdash \Gamma'' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta,i : S;\Delta \vdash \Gamma'' \sqsubseteq \Gamma''$, and $\Psi;\Theta,i : S;\Delta;\Omega';\Gamma' \vdash e' : \tau \Rightarrow \Phi',\Gamma'''$. By AT-IAbs, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \Lambda i.e' : \tau \Rightarrow \forall i : S.\Phi',\Gamma'''$. By Theorem 4, $\Psi;\Theta;\Delta \vdash \Gamma'' \sqsubseteq \Gamma' \setminus \Gamma$ and $\Psi;\Theta;\Delta \vdash \Gamma'' \sqsubseteq \Gamma''$. Finally, the fact that $\Theta;\Delta \vDash \forall i : S.\Phi'$ completes the proof of (1). For (2), AT-Anno suffices.

(AT-IApp) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e[I] \uparrow \tau[I/i] \Rightarrow \Phi_1 \wedge \Phi_2,\Gamma''$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \forall i : S.\tau \Rightarrow \Phi_1,\Gamma''$ and $\Theta;\Delta \vdash I : S \Rightarrow \Phi_2$, with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ'_1, Γ''' such that $|e| = |e'|$, $\Theta;\Delta \vDash \Phi'_1$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e \uparrow \forall i : S.\tau \Rightarrow \Phi'_1,\Gamma'''$. By AT-IApp, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e[I] \uparrow \tau[I/i] \Rightarrow \Phi'_1 \wedge \Phi_2,\Gamma'''$. Since $\Theta;\Delta \vDash \Phi'_1 \wedge \Phi_2$ and $|e[I]| = |e|[I] = |e'|[I] = |e'[I]|$, this completes (2). For (1), we have by Theorem 8.16

some Φ_3 such that $\Psi; \Theta; \Delta \vdash \tau[I/i] <: \tau[I/i] : \star \Rightarrow \Phi_3$ and $\Theta; \Delta \vDash \Phi_3$. By AT-Sub, we have that $\Psi; \Theta; \Delta; \Omega'; \Gamma' \vdash e[I] \downarrow \tau[I/i] \Rightarrow \Phi_1' \wedge \Phi_2 \wedge \Phi_3, \Gamma'''$, as required for (1).

(AT-Fix) Suppose $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{fix}\ x.e \downarrow \tau \Rightarrow \Phi, \Gamma$ by way of $\Psi; \Theta; \Delta; \Omega, x : \tau; \cdot \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma''$, with $\Theta; \Delta \vDash \Phi$, $\Psi; \Theta; \Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi; \Theta; \Delta \vdash \Omega' \sqsubseteq \Omega$. Then, $\Psi; \Theta; \Delta \vdash \Omega', x : \tau \sqsubseteq \Omega, x : \tau$. By IH, there are e', Φ', Γ'' such that $|e| = |e'|$, $\Theta; \Delta \vDash \Phi'$, and $\Psi; \Theta; \Delta; \Omega', x : \tau; \cdot \vdash e' \downarrow \tau \Rightarrow \Phi', \Gamma'''$. By AT-Fix, $\Psi; \Theta; \Delta; \Omega'; \Gamma' \vdash \texttt{fix}\ x.e' \downarrow \tau \Rightarrow \Phi', \Gamma'$. Of course, $|\texttt{fix}\ x.e'| = \texttt{fix}\ x.|e'| = \texttt{fix}\ x.|e| = |\texttt{fix}\ x.e|$. Further, $\Psi; \Theta; \Delta \vdash \Gamma' \sqsubseteq \Gamma' \setminus \Gamma$ by Theorem 5.6, and $\Psi; \Theta; \Delta \vdash \Gamma' \sqsubseteq \Gamma$ by one of the premises. This completes (1), and (2) follows by AT-Anno.

(AT-Ret) Suppose $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{ret}\ e \downarrow \mathbb{M}\,\phi(I, \vec{p})\,\tau \Rightarrow \Phi, \Gamma''$ from $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma''$, with $\Theta; \Delta \vdash \Phi$, $\Psi; \Theta; \Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi; \Theta; \Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ', Γ''' such that $|e'| = |e|$, $\Theta; \Delta \vDash \Phi'$, $\Psi; \Theta; \Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi; \Theta; \Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi; \Theta; \Delta; \Omega'; \Gamma' \vdash e' \downarrow \tau \Rightarrow \Phi', \Gamma'''$. By AT-Ret, $\Psi; \Theta; \Delta; \Omega'; \Gamma' \vdash \texttt{ret}\ e' \downarrow \mathbb{M}\,\phi(I, \vec{p})\,\tau \Rightarrow \Phi', \Gamma'''$, completing (1). (2) follows by AT-Anno.

(AT-Bind) Suppose $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{bind}\ x = e_1\ \texttt{in}\ e_2 \downarrow \mathbb{M}\,\phi(I, \vec{q})\,\tau_2 \Rightarrow (\vec{q} \geq \vec{p}) \wedge (I = J) \wedge \Phi_1 \wedge \Phi_2, \Gamma_2 \setminus \{x : \tau_1\}$ from $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 \uparrow \mathbb{M}\,\phi(J, \vec{p})\,\tau_1 \Rightarrow \Phi_1, \Gamma_1$ and $\Psi; \Theta; \Delta; \Omega; \Gamma_1, x : \tau_1 \vdash e_2 \downarrow \mathbb{M}\,\phi(I, \vec{q} - \vec{p})\,\tau_2 \Rightarrow \Phi_2, \Gamma_2$ with $\Theta; \Delta \vDash (\vec{q} \geq \vec{p}) \wedge (I = J) \wedge \Phi_1 \wedge \Phi_2$, $\Psi; \Theta; \Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi; \Theta; \Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e_1', Φ_1', Γ_1' such that $|e_1'| = |e_1|$, $\Theta; \Delta \vDash \Phi_1'$, $\Psi; \Theta; \Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi; \Theta; \Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, and $\Psi; \Theta; \Delta; \Omega'; \Gamma' \vdash e_1' \uparrow \mathbb{M}\,\phi(J, \vec{p})\,\tau_1 \Rightarrow \Phi_1,' \Gamma_1'$. We note that $\Psi; \Theta; \Delta \vdash \Gamma_1', x : \tau_1 \sqsubseteq \Gamma_1, x : \tau_1$, and so by IH, there are e_2', Φ_2', Γ_2' such that $|e_2'| = |e_2|$, $\Theta; \Delta \vDash \Phi_2'$, $\Psi; \Theta; \Delta \vdash \Gamma_2' \sqsubseteq \Gamma_1' \setminus \Gamma_1$, $\Psi; \Theta; \Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$, and $\Psi; \Theta; \Delta; \Omega'; \Gamma_1', x : \tau_1 \vdash e_2' \downarrow \mathbb{M}\,\phi(I, \vec{q} - \vec{p})\,\tau_2 \Rightarrow \Phi_2', \Gamma_2'$. By AT-Bind, $\Psi; \Theta; \Delta; \Omega'; \Gamma' \vdash \texttt{bind}\ x = e_1'\ \texttt{in}\ e_2' \downarrow \mathbb{M}\,\phi(I, \vec{q})\,\tau_2 \Rightarrow (\vec{q} \geq \vec{p}) \wedge (I = J) \wedge \Phi_1' \wedge \Phi_2', \Gamma_2' \setminus \{x : \tau_1\}$. Of course, $|\texttt{bind}\ x = e_1\ \texttt{in}\ e_2| = |\texttt{bind}\ x = e_1'\ \texttt{in}\ e_2'|$. Since $\Theta; \Delta \vDash (\vec{q} \geq \vec{p}) \wedge (I = J)$, we also have that $\Theta; \Delta \vDash (\vec{q} \geq \vec{p}) \wedge (I = J) \wedge \Phi_1' \wedge \Phi_2'$. We have $\Psi; \Theta; \Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$ by IH, and $\Psi; \Theta; \Delta \vdash \Gamma_2' \setminus \{x : \tau_1\} \sqsubseteq \Gamma' \setminus \Gamma$ follows by applying Theorem 5.6 to the rest of the weakening premises. This completes (1), and (2) follows by AT-Anno.

(AT-Tick) Immediate.

(AT-Release) Suppose $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{release}\ x = e_1\ \texttt{in}\ e_2 \downarrow \mathbb{M}\,\phi(I, \vec{p})\,\tau_2 \Rightarrow (I = J) \wedge \Phi_1 \wedge \Phi_2, \Gamma_2 \setminus \{x\}$ from $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 \uparrow [J|\vec{q}]\tau_1 \Rightarrow \Phi_1, \Gamma_1$ and $\Psi; \Theta; \Delta; \Omega; \Gamma_1, x : \tau \vdash e_2 \downarrow \mathbb{M}\,\phi(I, \vec{p} +$

$\vec{q})\,\tau_2 \Rightarrow \Phi_2, \Gamma_2$, with $\Theta;\Delta \vDash (I = J) \wedge \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e_1', Φ_1', Γ_1' such that $|e_1'| = |e_1|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vDash \Gamma_1' \sqsubseteq \Gamma' \smallsetminus \Gamma$, $\Psi;\Theta;\Delta \vDash \Gamma_1' \sqsubseteq \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1' \uparrow [J|\vec{q}]\tau_1 \Rightarrow \Phi_1', \Gamma_1'$. Since $\Psi;\Theta;\Delta \vDash \Gamma_1', x : \tau \sqsubseteq \Gamma_1, x : \tau$, we have by IH that there are e_2', Φ_2', Γ_2' such that $|e_2'| = |e_2|$, $\Theta;\Delta \vDash \Phi_2'$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_1' \smallsetminus \Gamma_1$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$, and $\Psi;\Theta;\Delta;\Omega';\Gamma_1', x : \tau \vdash e_2' \downarrow \mathbb{M}\,\phi(I, \vec{p} + \vec{q})\,\tau_2 \Rightarrow \Phi_2', \Gamma_2'$. By AT-Release, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{release}\ x = e_1'\ \texttt{in}\ e_2' \downarrow \mathbb{M}\,\phi(I, \vec{p})\,\tau_2 \Rightarrow (I = J) \wedge \Phi_1' \wedge \Phi_2', \Gamma_2 \smallsetminus \{x\}$. Of course, $|\texttt{release}\ x = e_1'\ \texttt{in}\ e_2'| = |\texttt{release}\ x = e_1\ \texttt{in}\ e_2|$. Since $\Theta;\Delta \vDash I = J$, we have $\Theta;\Delta \vDash (I = J) \wedge \Phi_1' \wedge \Phi_2'$. $\Psi;\Theta;\Delta \vdash \Gamma_2' \smallsetminus \{x\} \sqsubseteq \Gamma_2\{x\}$ is implied by $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$, and $\Psi;\Theta;\Delta \vdash \Gamma_2' \smallsetminus \{x\} \sqsubseteq \Gamma' \smallsetminus \Gamma$ follows from Theorem 5.6. This completes (1), and (2) follows from AT-Anno.

(AT-Store) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{store}[K|\vec{w}](e) \downarrow \mathbb{M}\,\phi(I, \vec{q})\,([J|\vec{p}]\,\tau) \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge (\vec{p} \le \vec{w} \le \vec{q}) \wedge (I = J = K), \Gamma''$ by way of $\Theta;\Delta \vdash K : \mathbb{N} \Rightarrow \Phi_1$, $\Theta;\Delta \vdash \vec{w} : \vec{\mathbb{R}^+} \Rightarrow \Phi_2$, $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi_3, \Gamma''$, with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge (\vec{p} \le \vec{w} \le \vec{q}) \wedge (I = J = K)$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Γ''', Φ_3' such that $|e'| = |e|$, $\Theta;\Delta \vDash \Phi_3'$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \smallsetminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \downarrow \tau \Rightarrow \Phi_3', \Gamma'''$. By AT-Store, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{store}[K|\vec{w}](e') \downarrow \mathbb{M}\,\phi(I, \vec{q})\,([J|\vec{p}]\,\tau) \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \Phi_3' \wedge (\vec{p} \le \vec{w} \le \vec{q}) \wedge (I = J = K), \Gamma'''$, which completes (1). For (2), AT-Anno suffices.

(AT-StoreConst) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{store}[J](e) \downarrow \mathbb{M}\,\phi(K, \vec{p})\,([I]\,\tau) \Rightarrow (\texttt{const}(I) \le \texttt{const}(J) \le \vec{p}) \wedge \Phi_1 \wedge \Phi_2, \Gamma''$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi_1, \Gamma''$ and $\Theta;\Delta \vdash J : \mathbb{R} \Rightarrow \Phi_2$, with $\Theta;\Delta \vDash (\texttt{const}(I) \le \texttt{const}(J) \le \vec{p}) \wedge \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ_1', Γ''' such that $|e'| = |e|$, $\Theta;\Delta \vDash \Phi_1'$, $\Phi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \smallsetminus \Gamma$, $\Phi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Phi;\Theta;\Delta;\Omega';\Theta' \vdash e' \downarrow \tau \Rightarrow \Phi_1', \Gamma'''$. By AT-StoreConst, $\Psi;\Theta;\Delta;\Omega';\Gamma \vdash \texttt{store}[J](e') \downarrow \mathbb{M}\,\phi(K, \vec{p})\,([I]\,\tau) \Rightarrow (\texttt{const}(I) \le \texttt{const}(J) \le \vec{p}) \wedge \Phi_1' \wedge \Phi_2, \Gamma''$, which completes (1). For (2), we use AT-Anno.

(AT-ReleaseConst) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{release}\ x = e_1\ \texttt{in}\ e_2 \downarrow \mathbb{M}\,\phi(I, \vec{p})\,\tau_2 \Rightarrow \Phi_1 \wedge \Phi_2, \Gamma_2 \smallsetminus \{x\}$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \uparrow [J]\tau_1 \Rightarrow \Phi_1, \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega;\Gamma_1, x : \tau_1 \vdash e_2 \downarrow \mathbb{M}\,\phi(I, \vec{p} + \texttt{const}(J))\,\tau_2 \Rightarrow \Phi_2, \Gamma_2$, with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e_1', Γ_1', Φ_1' such that $|e_1'| = |e_1|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \smallsetminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega;\Gamma' \vdash e_1' \uparrow [J]\tau_1 \Rightarrow \Phi_1', \Gamma_1'$. Since $\Psi;\Theta;\Delta \vdash \Gamma_1', x : \tau_1 \sqsubseteq \Gamma_1, x : \tau_1$, we have by IH that there are e_2', Φ_2', Γ_2' such that $|e_2'| = |e_2|$, $\Theta;\Delta \vDash \Phi_2'$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq$

$(\Gamma_1', x:\tau_1) \setminus (\Gamma_1, x:\tau_1)$, $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$, and $\Psi;\Theta;\Delta;\Omega';\Gamma_1', x:\tau_1 \vdash e_2' \downarrow \mathbb{M}\,\phi(I, \vec{p} + \texttt{const}(J))\,\tau_2 \Rightarrow \Phi_2', \Gamma_2'$. By AT-ReleaseCont, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{release}\ x = e_1'\ \texttt{in}\ e_2' \downarrow \mathbb{M}\,\phi(I,\vec{p})\,\tau_2 \Rightarrow \Phi_1' \wedge \Phi_2', \Gamma_2' \setminus \{x\}$. Of course, $|\texttt{release}\ x = e_1'\ \texttt{in}\ e_2'| = \texttt{release}\ x = |e_1'|\ \texttt{in}\ |e_2'| = \texttt{release}\ x = |e_1|\ \texttt{in}\ |e_2| = |\texttt{release}\ x = e_1\ \texttt{in}\ e_2|$. Also, $\Theta;\Delta \vDash \Phi_1' \wedge \Phi_2'$. $\Psi;\Theta;\Delta \vdash \Gamma_2' \sqsubseteq \Gamma_2$ holds by IH. It remains to show that $\Psi;\Theta;\Delta \vdash \Gamma_2' \setminus \{x\} \sqsubseteq \Gamma' \setminus \Gamma$. This follows by considering the remaining weakening judgments from the IHs with Theorem 5.6. This completes (1). For (2), as usual, we apply AT-Anno.

(AT-Shift) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{shift}(e) \downarrow \mathbb{M}\,\phi(I,\vec{q})\,\tau \Rightarrow (I \geq 1) \wedge \Phi, \Gamma''$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \mathbb{M}\,\phi(I-1, \lhd\,\vec{q})\,\tau \Rightarrow \Phi, \Gamma''$, with $\Theta;\Delta \vDash (I \geq 1) \wedge \Phi$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ', Γ''' such that $|e'| = |e|$, $\Theta;\Delta \vDash \Phi'$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \downarrow \mathbb{M}\,\phi(I-1, \lhd\,\vec{q})\,\tau \Rightarrow \Phi', \Gamma'''$. By AT-Shift, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{shift}(e') \downarrow \mathbb{M}\,\phi(I,\vec{q})\,\tau \Rightarrow (I \geq 1) \wedge \Phi', \Gamma'''$, which completes (1). For (2), we apply AT-Anno.

(AT-CImpI) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \Lambda.e \downarrow (\Phi' \Rightarrow \tau) \Rightarrow (\Phi' \to \Phi), \Gamma''$ from $\Psi;\Theta;\Delta,\Phi';\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma''$, with $\Theta;\Delta \vDash \Phi$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. Using **??**, we have by IH that there are e', Φ'', Γ''', $|e'| = |e|$, $\Theta;\Delta,\Phi' \vDash \Phi''$, $\Psi;\Theta;\Delta,\Phi' \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta,\Phi' \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta,\Phi';\Omega';\Gamma' \vdash e \downarrow \tau \Rightarrow \Phi'', \Gamma'''$. By AT-CImpI, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \Lambda.e' \downarrow (\Phi' \Rightarrow \tau) \Rightarrow (\Phi' \to \Phi''), \Gamma'''$. By definition, $|\Lambda.e'| = \Lambda.|e'| = \Lambda.|e| = |\Lambda.e|$. Next, since $\Theta;\Delta,\Phi' \vDash \Phi''$, we have $\Theta;\Delta \vDash \Phi' \to \Phi''$. The two weakenings again follow by Theorem 5.6. This completes (1), and (2) follows by AT-Anno.

(AT-CImpE) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e\{\} \uparrow \tau \Rightarrow \Phi \wedge \Phi', \Gamma''$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow (\Phi' \Rightarrow \tau) \Rightarrow \Phi, \Gamma''$, with $\Theta;\Delta \vDash \Phi \wedge \Phi'$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ''. Γ''' such that $|e'| = |e|$, $\Theta;\Delta \vDash \Phi''$. $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \uparrow (\Phi' \Rightarrow \tau) \Rightarrow \Phi'', \Gamma'''$. By AT-CImpE, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e'\{\} \uparrow \tau \Rightarrow \Phi'' \wedge \Phi', \Gamma'''$, which completes the proof of (2). For (1), we use Theorem 8.16 to get some Φ_1 with $\Theta;\Delta \vDash \Phi_1$ such that $\Psi;\Theta;\Delta \vdash \tau <: \tau : \star \Rightarrow \Phi_1$. Then, by AT-Sub, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e'\{\} \downarrow \tau \Rightarrow \Phi'' \wedge \Phi' \wedge \Phi_3, \Gamma'''$, as required.

(AT-CAndI) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash < e > \downarrow \Phi' \& \tau \Rightarrow \Phi \wedge \Phi', \Gamma''$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi, \Gamma''$, with $\Theta;\Delta \vDash \Phi \wedge \Phi'$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ'', Γ''' such that $|e'| = |e|$, $\Theta;\Delta \vDash \Phi''$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \downarrow \tau \Rightarrow \Phi'',\Gamma'''$. By AT-CAndI, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash < e' >\downarrow \Phi'\&\tau \Rightarrow \Phi'' \wedge \Phi',\Gamma'''$, which completes (1). For (2), one use of AT-Anno suffices.

(AT-CAndE) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{clet}\ x = e_1\ \texttt{in}\ e_2 \downarrow \tau' \Rightarrow \Phi_1 \wedge (\Phi \to \Phi_2),\Gamma_2 \setminus \{x : \tau\}$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1 \uparrow \Phi\&\tau \Rightarrow \Phi_1,\Gamma_1$ and $\Psi;\Theta;\Delta,\Phi;\Omega;\Gamma_1, x : \tau \vdash e_2 \downarrow \tau' \Rightarrow \Phi_2,\Gamma_2$, with $\Theta;\Delta \vDash \Phi_1 \wedge (\Phi \to \Phi_2)$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e_1', Φ_1', Γ_1' such that $|e_1'| = |e_1|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e_1' \uparrow \Phi\&\tau \Rightarrow \Phi_1',\Gamma_1'$. Since $\Psi;\Theta;\Delta \vdash \Gamma_1' \sqsubseteq \Gamma_1$, we also have by Theorem A.14 $\Psi;\Theta;\Delta,\Phi \vdash \Gamma_1', x : \tau \sqsubseteq \Gamma_1, x : \tau$, and so by IH, there are e_2', Φ_2', Γ_2' such that $|e_2'| = |e_2|$, $\Theta;\Delta,\Phi \vDash \Phi_2'$, $\Psi;\Theta;\Delta,\Phi \vdash \Gamma_2' \sqsubseteq \Gamma_1' \setminus \Gamma_1$, $\Psi;\Theta;\Delta,\Phi \vdash \Gamma_2' \sqsubseteq \Gamma_2$, and $\Psi;\Theta;\Delta,\Phi;\Omega';\Gamma_1', x : \tau \vdash e_2' \downarrow \tau' \Rightarrow \Phi_2',\Gamma_2'$. By AT-CAndE, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash \texttt{clet}\ x = e_1'\ \texttt{in}\ e_2' \downarrow \tau' \Rightarrow \Phi_1' \wedge (\Phi \to \Phi_2'),\Gamma_2' \setminus \{x : \tau\}$. As usual, we have that $|\texttt{clet}\ x = e_1'\ \texttt{in}\ e_2'| = |\texttt{clet}\ x = e_1\ \texttt{in}\ e_2|$. Combining the satisfactions from the premises, we have that $\Theta;\Delta \vDash \Phi_1' \wedge (\Phi \to \Phi_2')$. The fact that $\Psi;\Theta;\Delta,\Phi \vdash \Gamma_2' \sqsubseteq \Gamma_2$ is immediate from IH, and $\Psi;\Theta;\Delta,\Phi \vdash \Gamma_2' \sqsubseteq \Gamma' \setminus \Gamma$ follows as usual by considering the rest of the weakening judgments with Theorem 5.6. This completes (1). For (2), we apply AT-Anno.

(AT-Sub) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi_1 \wedge \Phi_2,\Gamma''$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \uparrow \tau' \Rightarrow \Phi_1,\Gamma''$ and $\Psi;\Theta;\Delta \vdash \tau' <: \tau : \star \Rightarrow \Phi_2$, with $\Theta;\Delta \vDash \Phi_1 \wedge \Phi_2$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ_1', Γ''' such that $|e'| = |e|$, $\Theta;\Delta \vDash \Phi_1'$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \uparrow \tau' \Rightarrow \Phi_1',\Gamma'''$. By AT-Sub, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \downarrow \tau \Rightarrow \Phi_1' \wedge \Phi_2,\Gamma'''$, which completes (1). For (2), one use of AT-Anno suffices.

(AT-Anno) Suppose $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash (e : \tau) \uparrow \tau \Rightarrow \Phi,\Gamma''$ from $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e \downarrow \tau \Rightarrow \Phi,\Gamma''$ with $\Theta;\Delta \vDash \Phi$, $\Psi;\Theta;\Delta \vdash \Gamma' \sqsubseteq \Gamma$, and $\Psi;\Theta;\Delta \vdash \Omega' \sqsubseteq \Omega$. By IH, there are e', Φ', Γ''' such that $|e| = |e'|$, $\Theta;\Delta \vDash \Phi'$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma' \setminus \Gamma$, $\Psi;\Theta;\Delta \vdash \Gamma''' \sqsubseteq \Gamma''$, and $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash e' \downarrow \tau \Rightarrow \Phi',\Gamma'''$. By AT-Anno, $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash (e' : \tau) \uparrow \tau \Rightarrow \Phi',\Gamma'''$, which completes (2) since $|(e' : \tau)| = |e'| = |e| = |(e : \tau)|$. For (1), we use Theorem 8.16 to get that $\Psi;\Theta;\Delta \vdash \tau <: \tau : \star \Rightarrow \Phi''$ with $\Theta;\Delta \vDash \Phi''$, and so by AT-Sub, we have that $\Psi;\Theta;\Delta;\Omega';\Gamma' \vdash (e' : \tau) \downarrow \tau \Rightarrow \Phi' \wedge \Phi'',\Gamma'''$, completing (1).

□

Proof of Theorem 8.23

By induction on the derivation of $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \tau$, we prove both claims simultaneously. In each case, we will only prove one of the two conclusions, based on the direction of the corresponding algorithmic rule. In cases where the rule is checking claim (2) follows immediately by AT-Anno. In cases where the rule is inferring, (1) follows by Theorem 8.16 and then AT-Sub. In all cases, the erasure property is immediate from the inductive hypotheses– we will elide this bit of the proof.

- **Case 1:** *T-Var-1.*
  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash x : \tau$

    (2) $x : \tau \in \Gamma$
  - **There exist** e', Φ', Γ' **such that:**
  - **Goal 1:**

    $\boxed{|e'| = x}$
  - **Goal 2:**

    $\boxed{\Psi;\Theta;\Delta \vDash \Phi'}$
  - **Goal 3:**

    $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e' \uparrow \tau \Rightarrow \Phi, \Gamma'}$

  Immediate
- **Case 2:** *T-Var-2.*

  Immediate
- **Case 3:** *T-Unit.*

  Immediate
- **Case 4:** *T-Base.*

  Immediate
- **Case 5:** *T-Absurd.*

  Immediate
- **Case 6:** *T-Nil.*
  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p \texttt{nil} : L^I \tau$

    (2) $\Theta;\Delta \vdash I : \mathbb{N}$

(3) $\Theta; \Delta \vDash I = 0$

- **There exist** e', Φ',Γ' **such that:**
- **Goal 1:**

$\boxed{|e'| = \texttt{nil}}$

- **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

- **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow L^I \tau \Rightarrow \Phi', \Gamma'}$

By Theorem 8.10, there is a Φ such that

(4) $\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi$

(5) $\Theta; \Delta \vDash \Phi$

Goals follow by AT-Nil, letting $e' = \texttt{nil}$, $\Phi' = \Phi \wedge (I = 0)$, and $\Gamma' = \Gamma$

- **Case 7:** *T-Cons.*
  - **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash_p e_1 :: e_2 : L^I \tau$

(2) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash_p e_1 : \tau$

(3) $\Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash_p e_2 : L^{I-1} \tau$

(4) $\Theta; \Delta \vDash I \geq 1$

  - **There exist** e', Φ',Γ' **such that:**
  - **Goal 1:**

$\boxed{|e'| = e_1 :: e_2}$

  - **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

  - **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash_p e' \downarrow L^I \tau \Rightarrow \Phi', \Gamma'}$

By IH, on (2) there are e_1', Φ_1, Γ_1' such that

(5) $|e_1'| = e_1$

(6) $\Theta; \Delta \vDash \Phi_1'$

(7) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1' \downarrow \tau \Rightarrow \Phi_1, \Gamma_1'$

By Theorem 8.22 on (7), there are e_1'', Φ_1', Γ_1'' such that

(8) $|e_1''| = |e_1'|$

(9) $\Theta; \Delta \vDash \Phi_1'$

(10) $\Psi; \Theta; \Delta \vdash \Gamma_1'' \sqsubseteq (\Gamma_1, \Gamma_2) \setminus \Gamma_1$

(11) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash e_1'' \downarrow \tau \Rightarrow \Phi_1', \Gamma_1''$

By IH, there are e_2', Φ_2, Γ_2' such that

(12) $|e_2'| = e_2$

(13) $\Theta; \Delta \vDash \Phi_2$

(14) $\Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash e_2' \downarrow L^{I-1}\tau \Rightarrow \Phi_2, \Gamma_2'$

Again by Theorem 8.22 on (14), we have e_2'', Φ_2', Γ_2'' such that

(15) $|e_2''| = |e_2'|$

(16) $\Theta; \Delta \vDash \Phi_2'$

(17) $\Psi; \Theta; \Delta; \Omega; \Gamma_1'' \vdash e_2'' \downarrow L^{I-1}\tau \Rightarrow \Phi_2', \Gamma_2''$

By AT-Cons on (11) and (17)

(18) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash e_1'' :: e_2'' \downarrow L^I\tau \Rightarrow \Phi_1' \wedge \Phi_2' \wedge (I \geq 1), \Gamma_2''$

Goals follow by $e' = e_1'' :: e_2''$, $\Phi' = \Phi_1' \wedge \Phi_2' \wedge (I \geq 1)$, and $\Gamma' = \Gamma_2''$

▸ **Case 8:** *T-Match.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{match}(e, e_1, h.t.e_2) : \tau'$

(2) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : L^I\tau$

(3) $\Psi; \Theta; \Delta, I = 0; \Omega; \Gamma_2 \vdash e_1 : \tau'$

(4) $\Psi; \Theta; \Delta, I \geq 1; \Omega; \Gamma_2, h : \tau, t : L^I\tau \vdash e_2 : \tau'$

▸ **There exist** e', Φ',Γ' **such that:**

▸ **Goal 1:**

$\boxed{|e'| = \texttt{match}(e, e_1, h.t.e_2)}$

▸ **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

▸ **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash_p e' \downarrow \tau' \Rightarrow \Phi', \Gamma'}$

By IH on (2)

(5) $|e'| = e$

(6) $\Theta; \Delta \vDash \Phi$

(7) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e' \uparrow L^I\tau \Rightarrow \Phi, \Gamma_1'$

By Theorem 8.22 on (7), there are e'', Φ', Γ_1'' such that

(8) $|e''| = |e'|$

(9) $\Theta; \Delta \vDash \Phi'$

(10) $\Psi; \Theta; \Delta \vdash \Gamma_1'' \sqsubseteq \Gamma_2$

(11) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash e'' \uparrow L^I\tau \Rightarrow \Phi', \Gamma_1''$

By IH on (3) there are e_1', Φ_1, Γ_2' such that

(12) $|e_1'| = e_1$

(13) $\Theta; \Delta, I = 0 \vDash \Phi_1$

(14) $\Psi; \Theta; \Delta, I = 0; \Omega; \Gamma_2 \vdash e_1' \downarrow \tau' \Rightarrow \Phi_1, \Gamma_2'$

By Theorem A.14 on (10)

(15) $\Psi; \Theta; \Delta, I = 0 \vdash \Gamma_1'' \sqsubseteq \Gamma_2$.

Then, by Theorem 8.22, there are e_1'', Φ_1', Γ_2'' such that

(16) $|e_1''| = |e_1'|$,

(17) $\Theta; \Delta, I = 0 \vDash \Phi_1'$

(18) $\Psi; \Theta; \Delta, I = 0; \Omega; \Gamma_1'' \vdash e_1'' \downarrow \tau' \Rightarrow \Phi_1', \Gamma_2''$

By IH again on (3) we have e_2', Φ_2, Γ_3' such that

(19) $|e_2'| = e_2$

(20) $\Theta; \Delta, I \geq 1 \vdash \Phi_2$

(21) $\Psi; \Theta; \Delta, I \geq 1; \Omega; \Gamma_2, h : \tau, t : L^I\tau \vdash e_2' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_3'$

By **??** and Theorem 5.6

(22) $\Psi; \Theta; \Delta, I \geq 1 \vdash \Gamma_1'', h : \tau, t : L^I\tau \sqsubseteq \Gamma_2, h : \tau, t : L^I\tau$

By Theorem 8.22 on (21) and (22), there are are e_2'', Φ_2', Γ_3'' such that

(23) $|e_2''| = |e_2'|$

(24) $\Theta; \Delta, I \geq 1 \vDash \Phi_2'$

(25) $\Psi; \Theta; \Delta, I \geq 1; \Omega; \Gamma_1'', h : \tau, t : L^I\tau \vdash e_2'' \downarrow \tau' \Rightarrow \Phi_2', \Gamma_3''$

Goals follow by AT-Match on (11), (18), (25)

▸ **Case 9:** *T-ExistI.*

- **Given:**

  (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p \texttt{pack}[I](e) : \exists i : S.\tau$

  (2) $\Theta; \Delta \vdash I : S$

  (3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e : \tau[I/i]$

- **There exist** e', Φ',Γ' **such that:**

- **Goal 1:**

  $\boxed{|e'| = \texttt{pack}[I](e)}$

- **Goal 2:**

  $\boxed{\Theta; \Delta \vDash \Phi'}$

- **Goal 3:**

  $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow \exists i : S.\tau \Rightarrow \Phi', \Gamma'}$

By Theorem 8.10 on (2) , there is a Φ_1 such that

(4) $\Theta; \Delta \vdash I : S \Rightarrow \Phi_1$

(5) $\Theta; \Delta \vDash \Phi_1$

By IH on (3), there are e', Φ_2, Γ' such that

(6) $|e'| = e$

(7) $\Theta; \Delta \vDash \Phi_2$

(8) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e' \downarrow \tau[I/i] \Rightarrow \Phi_2, \Gamma'$

Goals follow by AT-ExistI on (4) and (8)

- **Case 10:** *T-ExistE.*

  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{unpack}\ (i, x) = e_1\ \texttt{in}\ e_2 : \tau'$

    (2) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : \exists i : S.\tau$

    (3) $\Psi; \Theta, i : S; \Delta; \Omega; \Gamma_2, x : \tau \vdash e_2 : \tau'$

  - **There exist** e', Φ',Γ' **such that:**

  - **Goal 1:**

    $\boxed{|e'| = \texttt{unpack}\ (i, x) = e_1\ \texttt{in}\ e_2}$

  - **Goal 2:**

    $\boxed{\Theta; \Delta \vDash \Phi'}$

▸ **Goal 3:**

$$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash_p e' \downarrow \tau' \Rightarrow \Phi',\Gamma'}$$

By IH on (2), there are e_1', Φ_1, Γ_1' such that

(4) $|e_1'| = e_1$

(5) $\Theta;\Delta \vDash \Phi_1$

(6) $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_1' \uparrow \exists i:S.\tau \Rightarrow \Phi_1,\Gamma_1'$

By Theorem 8.22 there are e_1'', Φ_1', Γ_1'' such that

(7) $|e_1''| = |e_1'|$

(8) $\Theta;\Delta \vDash \Phi_1'$

(9) $\Psi;\Theta;\Delta \vdash \Gamma_1'' \sqsubseteq \Gamma_2$

(10) $\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash e_1'' \uparrow \exists i:S.\tau \Rightarrow \Phi_1',\Gamma_1''$

By IH on (3), there are e_2', Φ_2, Γ_2' such that

(11) $|e_2'| = e_2$

(12) $\Theta, i:S;\Delta \vDash \Phi_2$

(13) $\Psi;\Theta,i:S;\Delta;\Omega;\Gamma_2,x:\tau \vdash e_2' \downarrow \tau' \Rightarrow \Phi_2,\Gamma_2'$.

By Theorem 5.6 and Theorem 8.22 on (9) and (13) there are e_2'', Φ_2', Γ_2'' such that

(14) $|e_2''| = |e_2'|$

(15) $\Theta, i:S;\Delta \vDash \Phi_2'$

(16) $\Psi;\Theta,i:S;\Delta;\Omega;\Gamma_1'',x:\tau \vdash e_2'' \downarrow \tau' \Rightarrow \Phi_2',\Gamma_2''$

Goals follow by AT-ExistE on (10) and (16)

▸ **Case 11:** *T-Lam.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \lambda x.e : \tau_1 \multimap \tau_2$

(2) $\Psi;\Theta;\Delta;\Omega;\Gamma,x:\tau_1 \vdash e : \tau_2$

▸ **There exist** e', Φ',Γ' **such that:**

▸ **Goal 1:**

$$\boxed{|e'| = \lambda x.e}$$

▸ **Goal 2:**

$$\boxed{\Theta;\Delta \vDash \Phi'}$$

▸ **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow \tau' \Rightarrow \Phi', \Gamma'}$

By IH on (2) there are e', Φ, Γ' so that

(3) $|e'| = e$

(4) $\Theta; \Delta \vDash \Phi$

(5) $\Psi; \Theta; \Delta; \Omega; \Gamma, x : \tau_1 \vdash e' \downarrow \tau_2 \Rightarrow \Phi, \Gamma'$

Goals are immediate by AT-Lam

▸ **Case 12:** *T-App.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash e_1\ e_2 : \tau_2$

(2) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : \tau_1 \multimap \tau_2$

(3) $\Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash e_2 : \tau_1$

▸ **There exist** e', Φ',Γ' **such that:**

▸ **Goal 1:**

$\boxed{|e'| = e_1\ e_2}$

▸ **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

▸ **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash_p e' \uparrow \tau' \Rightarrow \Phi', \Gamma'}$

By IH on (2) there are e_1', Γ_1', Φ_1 such that

(4) $|e_1'| = e_1$

(5) $\Theta; \Delta \vDash \Phi_1$

(6) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1' \uparrow \tau_1 \multimap \tau_2 \Rightarrow \Phi_1, \Gamma_1'$

By Theorem 8.22 on (6), there are e_1'', Φ_1', Γ_1'' such that

(7) $|e_1''| = |e_1'|$

(8) $\Theta; \Delta \vDash \Phi_1'$

(9) $\Psi; \Theta; \Delta \vdash \Gamma_1'' \sqsubseteq \Gamma_2$

(10) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_1 \vdash e_1'' \uparrow \tau_1 \multimap \tau_2 \Rightarrow \Phi_1', \Gamma_1''$

By IH on (3), there are e_2', Γ_2', and Φ_2 such that

(11) $|e_2'| = e_2$

(12) $\Theta; \Delta \vDash \Phi_2$

(13) $\Psi;\Theta;\Delta;\Omega;\Gamma_2 \vdash e'_2 \downarrow \tau_1 \Rightarrow \Phi_2, \Gamma'_2$

By Theorem 8.22 on (9) and (13), there are e''_2, Γ''_2, and Φ'_2 such that

(14) $|e''_2| = |e'_2|$

(15) $\Theta;\Delta \vDash \Phi'_2$

(16) $\Psi;\Theta;\Delta;\Omega;\Gamma''_1 \vdash e''_2 \downarrow \tau_1 \Rightarrow \Phi'_2, \Gamma''_2$

Goals follow by AT-App on (10) and (16)

▸ **Case 13:** *T-TensorI.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash \langle\!\langle e_1, e_2 \rangle\!\rangle ; \tau_1 \otimes \tau_2$

(2) $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_1 : \tau_1$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma_2 \vdash e_2 : \tau_2$

▸ **There exist** e', Φ',Γ' **such that:**

▸ **Goal 1:**

$\boxed{|e'| = \langle\!\langle e_1, e_2 \rangle\!\rangle}$

▸ **Goal 2:**

$\boxed{\Theta;\Delta \vDash \Phi'}$

▸ **Goal 3:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash_p e' \downarrow \tau' \Rightarrow \Phi', \Gamma'}$

By IH on (2), there are e'_1, Φ_1, Γ'_1 such that

(4) $|e'_1| = e_1$

(5) $\Theta;\Delta \vDash \Phi_1$

(6) $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e'_1 \downarrow \tau_1 \Rightarrow \Phi_1, \Gamma'_1$

By Theorem 8.22 on (6), there are e''_1, Φ'_1, Γ''_1 such that

(7) $|e''_1| = |e'_1|$

(8) $\Theta;\Delta \vDash \Phi'_1$

(9) $\Psi;\Theta;\Delta \vdash \Gamma''_1 \sqsubseteq (\Gamma_1,\Gamma_2) \setminus \Gamma_1$

(10) $\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash e''_1 \downarrow \tau_2 \Rightarrow \Phi'_1, \Gamma''_1$.

By IH on (3), there are e'_2, Φ_2, Γ'_2 such that

(11) $|e'_2| = e_2$

(12) $\Theta;\Delta \vDash \Phi_2$

(13) $\Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash e_2' \downarrow \tau_2 \Rightarrow \Phi_2, \Gamma_2'$

But by Theorem 8.22 on (13) and (9), there are e_2'', Φ_2', Γ_2'' such that

(14) $|e_2''| = |e_2'|$

(15) $\Theta; \Delta \vDash \Phi_2'$

(16) $\Psi; \Theta; \Delta; \Omega; \Gamma_1'' \vdash e_2'' \downarrow \tau_2 \Rightarrow \Phi_2', \Gamma_2''$

Goals follow by AT-TensorI on (16)

- **Case 14:** *T-TensorE.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{let } \langle\!\langle x, y \rangle\!\rangle = e_1 \texttt{ in } e_2 : \tau'$

    (2) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : \tau_1 \otimes \tau_2$

    (3) $\Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau_1, y : \tau_2 \vdash e_2 : \tau'$
  - **There exist** e', Φ',Γ' **such that:**
  - **Goal 1:**

    $\boxed{|e'| = \texttt{let } \langle\!\langle x, y \rangle\!\rangle = e_1 \texttt{ in } e_2}$
  - **Goal 2:**

    $\boxed{\Theta; \Delta \vDash \Phi'}$
  - **Goal 3:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash_p e' \downarrow \tau' \Rightarrow \Phi', \Gamma'}$

  By IH on (2), there are e_1', Φ_1, Γ_1' such that

  (4) $|e_1'| = e_1$

  (5) $\Theta; \Delta \vDash \Phi_1$

  (6) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1' \uparrow \tau_1 \otimes \tau_2 \Rightarrow \Phi_1, \Gamma_1'$

  By Theorem 8.22 that there are e_1'', Φ_1', Γ_1'' such that

  (7) $|e_1''| = |e_1'|$

  (8) $\Theta; \Delta \vDash \Phi_1'$

  (9) $\Psi; \Theta; \Delta \vdash \Gamma_1'' \sqsubseteq (\Gamma_1, \Gamma_2 \setminus \Gamma_1)$

  (10) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash e_1'' \uparrow \tau_1 \otimes \tau_2 \Rightarrow \Phi_1', \Gamma_1''$.

  By IH on (3), there are e_2', Φ_2, Γ_2' such that

  (11) $|e_2'| = e_2$

  (12) $\Theta; \Delta \vDash \Phi_2$

(13) $\Psi;\Theta;\Delta;\Omega;\Gamma_2,x:\tau_1,y:\tau_2 \vdash e_2' \downarrow \tau' \Rightarrow \Phi_2,\Gamma_2'$

By Theorem 5.6 on (9) Theorem 8.22 on (13), there are e_2'', Φ_2', Γ_2'' such that

(14) $|e_2''| = |e_2'|$

(15) $\Theta;\Delta \vDash \Phi_2'$

(16) $\Psi;\Theta;\Delta,\Gamma_1'',x:\tau_1,y:\tau_2 \vdash e_2'' \downarrow \tau' \Rightarrow \Phi_2',\Gamma_2''$

Goals follow by AT-TensorE on (10) and (16)

- **Case 15:** *T-WithI.*
  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash (e_1,e_2):\tau_1\&\tau_2$

    (2) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1:\tau_1$

    (3) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_2:\tau_2.$

  - **There exist** e', Φ',Γ' **such that:**
  - **Goal 1:**

    $\boxed{|e'| = (e_1,e_2)}$

  - **Goal 2:**

    $\boxed{\Theta;\Delta \vDash \Phi'}$

  - **Goal 3:**

    $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e' \downarrow \tau_1\&\tau_2 \Rightarrow \Phi',\Gamma'}$

  By IH on (2), we have e_1', Φ_1, Γ_1 such that

  (4) $|e_1'| = e_1$

  (5) $\Theta;\Delta \vDash \Phi_1$

  (6) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_1':\tau_1 \Rightarrow \Phi_1,\Gamma_1.$

  By IH on (3), we have e_2', Φ_2, Γ_2 such that

  (4) $|e_2'| = e_2$

  (5) $\Theta;\Delta \vDash \Phi_2$

  (6) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e_2':\tau_2 \Rightarrow \Phi_2,\Gamma_2.$

  Goals follow by AT-WithI

- **Case 16:** *T-Fst.*
  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{fst}(e):\tau_1$

(2) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \tau_1 \& \tau_2$

- **There exist** e', Φ',Γ' **such that:**
- **Goal 1:**

  $\boxed{|e'| = \texttt{fst}(e)}$
- **Goal 2:**

  $\boxed{\Theta;\Delta \vDash \Phi'}$
- **Goal 3:**

  $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e' \uparrow \tau_1 \Rightarrow \Phi',\Gamma'}$

By IH on (2), there are e', Φ, Γ' such that

(3) $|e'| = e$

(4) $\Theta;\Delta \vDash \Phi$

(5) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e' \uparrow \tau_1 \& \tau_2 \Rightarrow \Phi,\Gamma'$

Goals follow by AT-Fst

- **Case 17:** *T-Snd.*

  Identical to case (16)

- **Case 18:** *T-Inl.*

- **Given:**

  (1) $\Psi;\Theta;\Delta;\Omega;\Gamma vdash \texttt{inl}(e) : \tau_1 \oplus \tau_2$

  (2) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \tau_1$
- **There exist** e', Φ',Γ' **such that:**
- **Goal 1:**

  $\boxed{|e'| = \texttt{inl}(e)}$
- **Goal 2:**

  $\boxed{\Theta;\Delta \vDash \Phi'}$
- **Goal 3:**

  $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e' \downarrow \tau_1 \oplus \tau_2 \Rightarrow \Phi',\Gamma'}$

By IH on (2), there are e', Φ, Γ' such that

(3) $|e'| = e$

(4) $\Theta;\Delta \vDash \Phi$

(5) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e' \downarrow \tau_1 \Rightarrow \Phi,\Gamma'$

Goals follow by AT-Inl

- **Case 19:** *T-Snd.*

  Identical to case (18)

- **Case 20:** *T-Case.*

  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash \texttt{case}(e, x.e_1, y.e_2) : \tau$

    (2) $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e : \tau_1 \oplus \tau_2$

    (3) $\Psi;\Theta;\Delta;\Omega;\Gamma_2, x : \tau_1 \vdash e_1 : \tau$

    (4) $\Psi;\Theta;\Delta;\Omega;\Gamma_2, y : \tau_2 \vdash e_2 : \tau$

  - **There exist** e', Φ',Γ' **such that:**

  - **Goal 1:**

    $\boxed{|e'| = \texttt{case}(e, x.e_1, y.e_2)}$

  - **Goal 2:**

    $\boxed{\Theta;\Delta \vDash \Phi'}$

  - **Goal 3:**

    $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash_p e' \downarrow \tau \Rightarrow \Phi',\Gamma'}$

  By IH on (2), there are e', Φ, Γ_1'

  (5) $|e'| = e$

  (6) $\Theta;\Delta \vDash \Phi$

  (7) $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e' \uparrow \tau_1 \oplus \tau_2 \Rightarrow \Phi,\Gamma_1'$

  By Theorem 8.22 there are e'', Φ', Γ_1'' such that

  (8) $|e''| = |e'|$

  (9) $\Theta;\Delta \vDash \Phi'$

  (10) $\Psi;\Theta;\Delta \vdash \Gamma_1'' \sqsubseteq (\Gamma_1,\Gamma_2) \smallsetminus \Gamma_1$

  (11) $\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash e'' \uparrow \tau_1 \oplus \tau_2 \Rightarrow \Phi',\Gamma_1''$.

  By IH on (3), there are e_1', Φ_1, Γ_2' such that

  (12) $|e_1'| = e_1$

  (13) $\Theta;\Delta \vDash \Phi_1$

  (14) $\Psi;\Theta;\Delta;\Omega;\Gamma_2, x : \tau_1 \vdash e_1' \downarrow \tau \Rightarrow \Phi_1,\Gamma_2'$

  By Theorem 5.6 and Theorem 8.22 on (10) and (14), there are e_1'', Φ_1', Γ_2'' such that

(15) $|e_1''| = |e_1'|$

(16) $\Theta; \Delta \vDash \Phi_1'$

(17) $\Psi; \Theta; \Delta; \Omega; \Gamma_1'', x : \tau_1 \vdash e_1'' \downarrow \tau \Rightarrow \Phi_1', \Gamma_2''$

By IH on (4), we have e_2', Φ_2, Γ_3' such that

(18) $|e_2'| = e_2$

(19) $\Theta; \Delta \vDash \Phi_2$

(20) $\Psi; \Theta; \Delta; \Omega; \Gamma_2, y : \tau_2 \vdash e_2' \downarrow \tau \Rightarrow \Phi_2, \Gamma_3'$

By Theorem 5.6 and Theorem 8.22 on (10) and (20), there are e_2'', Φ_2', Γ_3'' such that

(21) $|e_2''| = |e_2'|$

(22) $\Theta; \Delta \vDash \Phi_2'$

(23) $\Psi; \Theta; \Delta; \Omega; \Gamma_1'', y : \tau_2 \vdash e_2'' \downarrow \tau \Rightarrow \Phi_2', \Gamma_3''$

Goals follow by AT-Case on (11), (17), and (23)

▸ **Case 21:** *T-ExpI.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash !e : !\tau$

(2) $\Psi; \Theta; \Delta; \Omega; \cdot \vdash e : \tau$

▸ **There exist** e', Φ',Γ' **such that:**

▸ **Goal 1:**

$\boxed{|e'| = !e}$

▸ **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

▸ **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow \tau_1 \Rightarrow \Phi', \Gamma'}$

By IH on (2), there are e', Φ, Γ' such that

(3) $|e'| = e$

(4) $\Theta; \Delta \vDash \Phi$

(5) $\Psi; \Theta; \Delta; \Omega; \cdot \vdash e' \downarrow \tau \Rightarrow \Phi, \Gamma'$

Goals follow by AT-ExpI on (5)

▸ **Case 22:** *T-ExpE.*

▸ **Given:**

(1) $\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash \texttt{let}\ !x = e_1\ \texttt{in}\ e_2 : \tau'$

(2) $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_1 :!\tau$

(3) $\Psi;\Theta;\Delta;\Omega, x:\tau;\Gamma_2 \vdash e_2 : \tau'$.

▸ **There exist** e', Φ',Γ' **such that:**

▸ **Goal 1:**

$\boxed{|e'| = \texttt{let}\ !x = e_1\ \texttt{in}\ e_2}$

▸ **Goal 2:**

$\boxed{\Theta;\Delta \vDash \Phi'}$

▸ **Goal 3:**

$\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash_p e' \downarrow \tau' \Rightarrow \Phi', \Gamma'}$

By IH on (2), there are e_1', Φ_1, Γ_1' such that

(4) $|e_1'| = e_1$

(5) $\Theta;\Delta \vDash \Phi_1$

(6) $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_1' \uparrow !\tau \Rightarrow \Phi_1, \Gamma_1'$.

By Theorem 8.22, there are e_1'', Φ_1', Γ_1'' such thato

(7) $|e_1''| = |e_1'|$

(8) $\Theta;\Delta \vDash \Phi_1'$

(9) $\Psi;\Theta;\Delta \vdash \Gamma_1'' \sqsubseteq (\Gamma_1,\Gamma_2) \smallsetminus \Gamma_1$

(10) $\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash e_1'' \uparrow !\tau \Rightarrow \Phi_1', \Gamma_1''$

By IH on (3), there are e_2', Φ_2, Γ_2' such that

(11) $|e_2'| = e_2$

(12) $\Theta;\Delta \vDash \Phi_2$

(13) $\Psi;\Theta;\Delta;\Omega, x:\tau;\Gamma_2 \vdash e_2' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2'$

By **??** on (9) and (13), there are e_2'', Φ_2', Γ_2'' such that

(14) $|e_2''| = |e_2|$

(15) $\Theta;\Delta \vDash \Phi_2'$

(16) $\Psi;\Theta;\Delta;\Omega, x:\tau;\Gamma_1'' \vdash e_2'' \downarrow \tau' \Rightarrow \Phi_2', \Gamma_2''$

Goals follow by AT-ExpE on (10) and (16)

▸ **Case 23:** *T-TAbs.*

- **Given:**

  (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda\alpha.e : \forall\alpha : K.\tau$

  (2) $\Psi, \alpha : K; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau$

- **There exist** e', Φ',Γ' **such that:**

- **Goal 1:**

  $\boxed{|e'| = \Lambda\alpha.e}$

- **Goal 2:**

  $\boxed{\Theta; \Delta \vDash \Phi'}$

- **Goal 3:**

  $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow \forall\alpha : K.\tau \Rightarrow \Phi', \Gamma'}$

By IH on (2), there are e',Φ, Γ' such that

(3) $|e'| = e$

(4) $\Theta; \Delta \vDash \Phi$

(5) $\Psi, \alpha : K; \Theta; \Delta; \Omega; \Gamma \vdash e' \downarrow \tau \Rightarrow \Phi, \Gamma'$

Goals follow by AT-TAbs

- **Case 24:** *T-TApp.*

- **Given:**

  (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e[\tau'] : \tau[\tau'/\alpha]$

  (2) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \forall\alpha : K.\tau$

  (3) $\Psi; \Theta; \Delta \vdash \tau' : K.$

- **There exist** e', Φ',Γ' **such that:**

- **Goal 1:**

  $\boxed{|e'| = e\,[\tau]}$

- **Goal 2:**

  $\boxed{\Theta; \Delta \vDash \Phi'}$

- **Goal 3:**

  $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \uparrow \tau[\tau'/\alpha] \Rightarrow \Phi', \Gamma'}$

By IH on (2), there are e', Φ, Γ' such that

(4) $|e'| = e$

(5) $\Theta; \Delta \vDash \Phi$

(6) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e' \uparrow \forall \alpha : K.\tau \Rightarrow \Phi, \Gamma'$

By Theorem 8.13 on (3), there is some Φ' such that

(7) $\Theta; \Delta \vDash \Phi'$

(8) $\Psi; \Theta; \Delta \vdash \tau' : K \Rightarrow \Phi'$

Goals follow by AT-TApp on (6) and (8)

- **Case 25:** *T-IAbs.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda i.e : \forall i : S.\tau$

    (2) $\Psi; \Theta, i : S; \Delta; \Omega; \Gamma \vdash e : \tau$
  - **There exist** e', Φ',Γ' **such that:**
  - **Goal 1:**

    $\boxed{|e'| = \Lambda\imath.e}$
  - **Goal 2:**

    $\boxed{\Theta; \Delta \vDash \Phi'}$
  - **Goal 3:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow \forall \alpha : K.\tau \Rightarrow \Phi', \Gamma'}$

  By IH on (2), there are e', Φ, Γ' such that

  (3) $|e'| = e$

  (4) $\Theta, i : S; \Delta \vDash \Phi$

  (5) $\Psi; \Theta, i : S; \Delta; \Omega; \Gamma \vdash e' \downarrow \tau \Rightarrow \Phi, \Gamma'$

  Goals follow by AT-IAbs on (5)

- **Case 26:** *T-IApp.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e[I] : \tau[I/i]$

    (2) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \forall i : S.\tau$

    (3) $\Theta; \Delta \vdash I : S$
  - **There exist** e', Φ',Γ' **such that:**
  - **Goal 1:**

$\boxed{|e'| = e\,[I]}$

- **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

- **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \uparrow \tau[I/i] \Rightarrow \Phi', \Gamma'}$

By IH on (2), there are e', Φ, Γ' such that

(4) $|e'| = e$

(5) $\Theta; \Delta \vDash \Phi$

(6) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e \uparrow \forall i : S.\tau \Rightarrow \Phi, \Gamma'$

By Theorem 8.10 on (3), there is some Φ' such that

(7) $\Theta; \Delta \vDash \Phi'$

(8) $\Theta; \Delta \vdash I : S \Rightarrow \Phi'$

Goals follow by AT-IApp on (6) and (8)

- **Case 27:** *T-Fix.*

- **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{fix}\ x.e : \tau$

(2) $\Psi; \Theta; \Delta; \Omega, x : \tau; \cdot \vdash e : \tau$

- **There exist** e', Φ',Γ' **such that:**

- **Goal 1:**

$\boxed{|e'| = \texttt{fix}\ x.e}$

- **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

- **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow \tau \Rightarrow \Phi', \Gamma'}$

By IH on (2), there are e', Φ, Γ' such that

(3) $|e'| = e$

(4) $\Theta; \Delta \vDash \Phi$

(5) $\Psi; \Theta; \Delta; \Omega, x : \tau; \cdot \vdash e' \downarrow \tau \Rightarrow \Phi, \Gamma'$

Goals follow by AT-Fix on (5)

- **Case 28:** *T-Ret.*

- **Given:**

  (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash \texttt{ret}\ e : \mathbb{M}\,(I,\vec{p})\,\tau$ by way of

  (2) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \tau$.

- **There exist** e', Φ',Γ' **such that:**

- **Goal 1:**

  $\boxed{|e'| = \texttt{ret}\ e}$

- **Goal 2:**

  $\boxed{\Theta;\Delta \vDash \Phi'}$

- **Goal 3:**

  $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e' \downarrow \tau \Rightarrow \Phi',\Gamma'}$

By IH on (2), there are e', Φ, Γ' such that

(3) $|e'| = e$

(4) $\Theta;\Delta \vDash \Phi$

(5) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e' \downarrow M\,(I,\vec{p})\,\tau \Rightarrow \Phi,\Gamma'$

Goals follow by AT-Ret on (5)

- **Case 29:** *T-Bind.*

- **Given:**

  (1) $\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash \texttt{bind}\ \ x = e_1\ \ \texttt{in}\ \ e_2 : \mathbb{M}\,\phi(I,\vec{p}+\vec{q})\,\tau_2$

  (2) $\Psi;\Theta;\Delta;\Omega;\Gamma_1 \vdash e_1 : \mathbb{M}\,\phi(I,\vec{p})\,\tau_1$

  (3) $\Psi;\Theta;\Delta;\Omega;\Gamma_2, x:\tau_1 \vdash e_2 : \mathbb{M}\,\phi(I,\vec{q})\,\tau_2$

- **There exist** e', Φ',Γ' **such that:**

- **Goal 1:**

  $\boxed{|e'| = \texttt{bind}\ \ x = e_1\ \ \texttt{in}\ \ e_2}$

- **Goal 2:**

  $\boxed{\Theta;\Delta \vDash \Phi'}$

- **Goal 3:**

  $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2 \vdash_p e' \downarrow \mathbb{M}\,\phi(I,\vec{p}+\vec{q})\,\tau_2 \Rightarrow \Phi',\Gamma'}$

By IH on (2), there are e_1', Φ_1, Γ_1' such that

(4) $|e_1'| = e_1$

(5) $\Theta; \Delta \vDash \Phi_1$

(6) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1' \downarrow \mathbb{M}\, \phi(I, \vec{p})\, \tau_1 \Rightarrow \Phi_1, \Gamma_1'$

By Theorem 8.22 on (6), there are e_1'', Γ_1'', Φ_1' such that

(7) $|e_1''| = |e_1'|$

(8) $\Theta; \Delta \vDash \Phi_1'$

(9) $\Psi; \Theta; \Delta \vdash \Gamma_1'' \sqsubseteq (\Gamma_1, \Gamma_2) \setminus \Gamma_1$

(10) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash e_1'' \downarrow \mathbb{M}\, \phi(I, \vec{p})\, \tau_1 \Rightarrow \Phi_1', \Gamma_1''$

By IH on (3), there are e_2', Φ_2, Γ_2' such that

(11) $|e_2'| = e_2$

(12) $\Theta; \Delta \vDash \Phi_2$

(13) $\Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau_1 \vdash e_2' \downarrow \mathbb{M}\, \phi(I, \vec{q})\, \tau_2 \Rightarrow \Phi_2, \Gamma_2'$

By Theorem 8.22 on (9) and (11), there are e_2'', Γ_2'', Φ_2' such that

(14) $|e_2''| = |e_2'|$

(15) $\Theta; \Delta \vDash \Phi_2'$

(16) $\Psi; \Theta; \Delta; \Omega; \Gamma_1'', x : \tau_1 \vdash e_2'' \uparrow \mathbb{M}\, \phi(I, \vec{q})\, \tau_2 \Rightarrow \Phi_2', \Gamma_2''$

By Theorem 8.16, there is some Φ_3 such that $\Theta; \Delta \vDash \Phi_3$ and

(17) $\Psi; \Theta; \Delta \vdash \tau_2 <: \tau_2 : \star \Rightarrow \Phi_3$.

By AS-Monad on (17)

(18) $\Psi; \Theta; \Delta \vdash \mathbb{M}\, \phi(I, \vec{q})\, \tau_2 <: \mathbb{M}\, \phi(I, (\vec{p} + \vec{q}) - \vec{p})\, \tau_2 : \star \Rightarrow \Phi_3 \wedge (I = I) \wedge (\vec{q} \leq (\vec{p} + \vec{q}) - \vec{p})$

By AT-Sub on (16) and (18)

(19) $\Psi; \Theta; \Delta; \Omega; \Gamma_1'', x : \tau_1 \vdash e_2'' \downarrow \mathbb{M}\, \phi(I, (\vec{p} + \vec{q}) - \vec{p})\, \tau_2 \Rightarrow \Phi_2' \wedge \Phi_3 \wedge (I = I) \wedge (\vec{q} \leq (\vec{p} + \vec{q}) - \vec{p}), \Gamma_2''$.

Goals follow by AT-Bind on (10) and (19)

▸ **Case 30:** *T-Tick.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{tick}[I|\vec{p}] : \mathbb{M}\, \phi(I, \vec{p})\, 1$

(2) $\Theta; \Delta \vdash I : \mathbb{N}$

(3) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+}$

▸ **There exist** e', Φ',Γ' **such that:**

▸ **Goal 1:**

$\boxed{|e'| = \texttt{tick}[I|\vec{p}]}$

▸ **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

▸ **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \uparrow \mathbb{M}\, \phi(I, \vec{p})\, 1 \Rightarrow \Phi', \Gamma'}$

By Theorem 8.10 on (2), there is some Φ_1 such that

(4) $\Theta; \Delta \vDash \Phi_1$

(5) $\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi_1$

By Theorem 8.10 on (3), there is some Φ_2 such that

(6) $\Theta; \Delta \vDash \Phi_2$, and

(7) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+}$.

Goals follow by AT-Tick on (5) and (7)

▸ **Case 31:** *T-Release.*

▸ **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{release}\ \ x = e_1\ \ \texttt{in}\ \ e_2 : \mathbb{M}\,(I, \vec{p})\, \tau_2$

(2) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : [I|\vec{q}]\tau_1$

(3) $\Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau \vdash e_2 : \mathbb{M}\, \phi(I, \vec{p} + \vec{q})\, \tau_2$

▸ **There exist** e', Φ',Γ' **such that:**

▸ **Goal 1:**

$\boxed{|e'| = \texttt{release}\ \ x = e_1\ \ \texttt{in}\ \ e_2}$

▸ **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

▸ **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash_p e' \downarrow \mathbb{M}\,(I, \vec{p})\, \tau_2 \Rightarrow \Phi', \Gamma'}$

By IH on (2), there are e_1', Φ_1, Γ_1' such that

(4) $|e_1'| = e_1$

(5) $\Theta; \Delta \vDash \Phi_1$

(6) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1' \uparrow [I|\vec{q}]\tau_1 \Rightarrow \Phi_1, \Gamma_1'$

By Theorem 8.22 on (6) there are e_1'', Φ_1', Γ_1'' such that

(7) $|e_1''| = |e_1'|$

(8) $\Theta; \Delta \vDash \Phi_1'$

(9) $\Psi; \Theta; \Delta \vdash \Gamma_1'' \sqsubseteq (\Gamma_1, \Gamma_2) \setminus \Gamma_1$

(10) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash e_1'' \uparrow [I|\vec{q}]\tau_1 \Rightarrow \Phi_1', \Gamma_1''$

By IH on (3), there are e_2', Φ_2, Γ_2' such that

(11) $|e_2'| = e_2$

(12) $\Theta; \Delta \vDash \Phi_2$

(13) $\Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau \vdash e_2' \downarrow \mathbb{M}\, \phi(I, \vec{p} + \vec{q})\, \tau_2 \Rightarrow \Phi_2, \Gamma_2'$

By Theorem 8.22 on (9) and (13), there are e_2'', Φ_2', Γ_2'' such that

(14) $|e_2''| = |e_2'|$

(15) $\Theta; \Delta \vDash \Phi_2'$

(16) $\Psi; \Theta; \Delta; \Omega; \Gamma_1'', x : \tau \vdash e_2'' \downarrow \mathbb{M}\, \phi(I, \vec{p} + \vec{q})\, \tau \Rightarrow \Phi_2', \Gamma_2''$

Goals follow by AT-Release on (10) and (16)

- **Case 32:** *T-Store.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{store}[I|\vec{p}](e) : \mathbb{M}\, \phi(I, \vec{p})\, ([I|\vec{p}]\, \tau)$

    (2) $\Theta; \Delta \vdash I : \mathbb{N}$

    (3) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+}$

    (4) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau$
  - **There exist** e', Φ',Γ' **such that:**
  - **Goal 1:**

    $\boxed{|e'| = \texttt{store}[I|\vec{p}](e)}$
  - **Goal 2:**

    $\boxed{\Theta; \Delta \vDash \Phi'}$
  - **Goal 3:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow \mathbb{M}\, \phi(I, \vec{p})\, ([I|\vec{p}]\, \tau) \Rightarrow \Phi', \Gamma'}$

  By Theorem 8.10 on (2), there is Φ_1 such that

  (5) $\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi_1$

  (6) $\Theta; \Delta \vDash \Phi_1$

  By Theorem 8.10 on (3), there is Φ_2 such that

  (7) $\Theta; \Delta \vdash \vec{p} : \vec{\mathbb{R}^+} \Rightarrow \Phi_2$

(8) $\Theta; \Delta \vDash \Phi_2$

By IH on (4), there are e', Φ_3, Γ' such that

(9) $|e'| = e$

(10) $\Theta; \Delta \vDash \Phi_3$

(11) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e' \downarrow \tau \Rightarrow \Phi_3, \Gamma'$

Goals follow by AT-Store on (6), (8), and (11)

- **Case 33:** *T-StoreConst.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \texttt{store}[I](e) : \mathbb{M}\,(K, \texttt{const}(I))\,([I]\,\tau)$

    (2) $\Theta; \Delta \vdash I : \mathbb{N}$

    (3) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau.$
  - **There exist** e', Φ',Γ' **such that:**
  - **Goal 1:**

    $\boxed{|e'| = \texttt{store}[I](e)}$
  - **Goal 2:**

    $\boxed{\Theta; \Delta \vDash \Phi'}$
  - **Goal 3:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow \mathbb{M}\,(K, \texttt{const}(I))\,([I]\,\tau)\Phi', \Gamma'}$

  By Theorem 8.10, there is some Φ_1 such that

  (4) $\Theta; \Delta \vDash \Phi_1$

  (5) $\Theta; \Delta \vdash I : \mathbb{N} \Rightarrow \Phi_1$

  By IH on (3), there are e', Φ_2, and Γ' such that

  (6) $|e'| = e$

  (7) $\Theta; \Delta \vDash \Phi_2$

  (8) $\Psi; \Theta; \Delta \vdash e' \downarrow \tau \Rightarrow \Phi_2, \Gamma'$

  Goals follow immediately from AT-StoreConst on (5) and (8)

- **Case 34:** *T-ReleaseConst.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{release}\ \ x = e_1\ \ \texttt{in}\ \ e_2 : \mathbb{M}\,(I, \vec{p})\,\tau_2$

    (2) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : [J]\tau_1$

(3) $\Psi;\Theta;\Delta;\Omega;\Gamma_2,x:\tau\vdash e_2:\mathbb{M}\,\phi(I,\vec{p}+\mathtt{const}(J))\,\tau_2$

- **There exist** e', Φ',Γ' **such that:**
- **Goal 1:**

  $\boxed{|e'| = \mathtt{release}\ \ x = e_1\ \ \mathtt{in}\ \ e_2}$
- **Goal 2:**

  $\boxed{\Theta;\Delta\vDash\Phi'}$
- **Goal 3:**

  $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2\vdash_p e'\downarrow\mathbb{M}\,(I,\vec{p})\,\tau_2,\Gamma'}$

By IH on (2), there are e_1', Φ_1, Γ_1' such that

(4) $|e_1'| = e_1$

(5) $\Theta;\Delta\vDash\Phi_1$

(6) $\Psi;\Theta;\Delta;\Omega;\Gamma_1\vdash e_1'\uparrow[J]\tau_1\Rightarrow\Phi_1,\Gamma_1'$

By Theorem 8.22 there are e_1'', Φ_1', Γ_1'' such that

(7) $|e_1''| = |e_1'|$

(8) $\Theta;\Delta\vDash\Phi_1'$

(9) $\Psi;\Theta;\Delta\vdash\Gamma_1''\sqsubseteq\Gamma_2$

(10) $\Psi;\Theta;\Delta;\Omega;\Gamma_1,\Gamma_2\vdash e_1''\uparrow[J]\tau_1\Rightarrow\Phi_1',\Gamma_1''$

By IH on (3), there are e_2', Φ_2, Γ_2' such that

(11) $|e_2'| = e_2$

(12) $\Theta;\Delta\vDash\Phi_2$

(13) $\Psi;\Theta;\Delta;\Omega;\Gamma_2,x:\tau\vdash e_2'\downarrow\mathbb{M}\,(I,\vec{p}+\mathtt{const}(J))\,\tau_2\Rightarrow\Phi_2,\Gamma_2'$

By Theorem 5.6 and Theorem 8.22 on (10) and (13), there are e_2'', Φ_2', Γ_2'' such that

(14) $|e_2''| = |e_2'|$

(15) $\Theta;\Delta\vDash\Phi_2'$

(16) $\Psi;\Theta;\Delta;\Omega;\Gamma_1'',x:\tau\vdash e_2''\downarrow\mathbb{M}\,(I,\vec{p}+\mathtt{const}(J))\,\tau_2\Rightarrow\Phi_2',\Gamma_2''$

Goals follow by AT-ReleaseConst on (10) and (16)

- **Case 35:** *T-Shift.*
  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma\vdash\mathtt{shift}(e):\mathbb{M}\,(I,\vec{p})\,\tau$

    (2) $\Psi;\Theta;\Delta;\Omega;\Gamma\vdash e:\mathbb{M}\,(I-1,\triangleleft\,\vec{p})\,\tau$

(3) $\Theta; \Delta \vDash I \geq 1$

- **There exist** e', Φ',Γ' **such that:**
- **Goal 1:**

$\boxed{|e'| = \texttt{shift}(e)}$

- **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

- **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow \mathbb{M}\,(I, \vec{p})\,\tau, \Gamma'}$

By IH on (2), there are e', Φ, Γ' such thato

(4) $|e'| = e$

(5) $\Theta; \Delta \vDash \Phi$

(6) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e' \downarrow \mathbb{M}\,(I - 1, \triangleleft\, \vec{p})\,\tau \Rightarrow \Phi, \Gamma'$

Goals follow by AT-Shift on (6)

- **Case 36:** *T-CImpI.*

- **Given:**

(1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda.e : (\Phi' \Rightarrow \tau)$

(2) $\Psi; \Theta; \Delta, \Phi'; \Omega; \Gamma \vdash e : \tau$

- **There exist** e', Φ',Γ' **such that:**
- **Goal 1:**

$\boxed{|e'| = \Lambda.e}$

- **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

- **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow (\Phi' \Rightarrow \tau), \Gamma'}$

By IH on (2), there are e', Φ, Γ' such that

(3) $|e'| = e$

(4) $\Theta; \Delta, \Phi' \vDash \Phi$

(5) $\Psi; \Theta; \Delta, \Phi'; \Omega; \Gamma \vdash e' \downarrow \tau \Rightarrow \Phi, \Gamma'$

Goals follow by AT-CImpI on (5)

- **Case 37:** *T-CImpE.*

- **Given:**

  (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e\{\} : \tau$

  (2) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \Phi' \Rightarrow \tau$

  (3) $\Theta;\Delta \vDash \Phi'$

- **There exist** e', Φ',Γ' **such that:**

- **Goal 1:**

  $\boxed{|e'| = e\,\{\}}$

- **Goal 2:**

  $\boxed{\Theta;\Delta \vDash \Phi'}$

- **Goal 3:**

  $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e' \uparrow \tau, \Gamma'}$

By IH on (2), there are e', Φ, Γ' such that

(4) $|e'| = e$

(5) $\Theta;\Delta \vDash \Phi$

(6) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e' \uparrow \tau \Rightarrow \Phi, \Gamma'$

Goals follow by AT-CImpE on (6)

- **Case 38:** *T-CAndI.*

  - **Given:**

    (1) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash < e >: \Phi' \& \tau$

    (2) $\Psi;\Theta;\Delta;\Omega;\Gamma \vdash e : \tau$

    (3) $\Theta;\Delta \vDash \Phi'$

  - **There exist** e', Φ',Γ' **such that:**

  - **Goal 1:**

    $\boxed{|e'| =< e >}$

  - **Goal 2:**

    $\boxed{\Theta;\Delta \vDash \Phi'}$

  - **Goal 3:**

    $\boxed{\Psi;\Theta;\Delta;\Omega;\Gamma \vdash_p e' \downarrow \Phi' \& \tau, \Gamma'}$

  By IH on (2), there are e', Φ, Γ' such that

(4) $|e'| = e$

(5) $\Theta; \Delta \vDash \Phi$

(6) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e' \downarrow \tau \Rightarrow \Phi, \Gamma'$.

Goals follow by AT-CAndI on (6)

- **Case 39:** *T=CAndE.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash \texttt{clet}\ \ x = e_1\ \ \texttt{in}\ \ e_2 : \tau'$

    (2) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : \Phi' \& \tau$

    (3) $\Psi; \Theta; \Delta, \Phi'; \Omega; \Gamma_2, x : \tau \vdash e_2 : \tau'$
  - **There exist** e', Φ',Γ' **such that:**
  - **Goal 1:**

    $\boxed{|e'| = \texttt{clet}\ \ x = e_1\ \ \texttt{in}\ \ e_2}$
  - **Goal 2:**

    $\boxed{\Theta; \Delta \vDash \Phi'}$
  - **Goal 3:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash_p e' \downarrow \tau', \Gamma'}$

  By IH on (2), there are e_1', Φ_1, Γ_1' such that

  (4) $|e_1'| = e_1$

  (5) $\Theta; \Delta \vDash \Phi_1$

  (6) $\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1' \uparrow \Phi' \& \tau \Rightarrow \Phi_1, \Gamma_1'$

  By Theorem 8.22 on (6), there are e_1'', Φ_1', Γ_1'' such that

  (7) $|e_1''| = |e_1'|$

  (8) $\Theta; \Delta \vDash \Phi_1'$

  (9) $\Psi; \Theta; \Delta \vdash \Gamma_1'' \sqsubseteq (\Gamma_1, \Gamma_2) \setminus \Gamma_1$

  (10) $\Psi; \Theta; \Delta; \Omega; \Gamma_1, \Gamma_2 \vdash e_1'' \uparrow \Phi' \& \tau \Rightarrow \Phi_1', \Gamma_1''$

  By IH on (3), there are e_2', Φ_2, Γ_2' such that

  (11) $|e_2'| = e_2$

  (12) $\Theta; \Delta, \Phi' \vDash \Phi_2$

  (13) $\Psi; \Theta; \Delta, \Phi'; \Omega; \Gamma_2, x : \tau \vdash e_2' \downarrow \tau' \Rightarrow \Phi_2, \Gamma_2'$

  By Theorem 5.6 and Theorem A.14 on (10)

(14) $\Psi; \Theta; \Delta, \Phi' \vdash \Gamma_1'', x : \tau \sqsubseteq \Gamma_2, x : \tau$

So, by Theorem 8.22 on (14) and (14) there are e_2'', Φ_2', Γ_2'' such that

(15) $|e_2''| = |e_2'|$

(16) $\Theta; \Delta, \Phi' \vDash \Phi_2'$

(17) $\Psi; \Theta; \Delta, \Phi'; \Omega; \Gamma_1'', x : \tau \vdash e_2'' \downarrow \tau' \Rightarrow \Phi_2', \Gamma_2''$

Goals follow by AT-CAndE on (10) and (17)

- **Case 40:** *T-Sub.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau$

    (2) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau'$

    (3) $\Psi; \Theta; \Delta \vdash \tau' <: \tau : \star.$
  - **There exist** e', Φ',Γ' **such that:**
  - **Goal 1:**

    $\boxed{|e'| = e}$
  - **Goal 2:**

    $\boxed{\Theta; \Delta \vDash \Phi'}$
  - **Goal 3:**

    $\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow \tau, \Gamma'}$

  By IH on (2), there are e', Φ_1, Γ' so that

  (4) $|e'| = e$

  (5) $\Theta; \Delta \vDash \Phi_1$

  (6) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e' \uparrow \tau' \Rightarrow \Phi_1, \Gamma'$

  By Theorem 8.21 on (3), there is Φ_2 such that

  (7) $\Theta; \Delta \vDash \Phi_2$

  (8) $\Psi; \Theta; \Delta \vdash \tau' <: \tau : \star \Rightarrow \Phi_2$

  Goals follow by AT-Sub on (6) and (8)

- **Case 41:** *T-Weaken.*
  - **Given:**

    (1) $\Psi; \Theta; \Delta; \Omega'; \Gamma' \vdash e : \tau$

    (2) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau$

(3) $\Theta; \Delta \vdash \Omega' \sqsubseteq \Omega$

(4) $\Theta; \Delta \vdash \Gamma' \sqsubseteq \Gamma$

- **There exist** e', Φ',Γ' **such that:**
- **Goal 1:**

$\boxed{|e'| = e}$

- **Goal 2:**

$\boxed{\Theta; \Delta \vDash \Phi'}$

- **Goal 3:**

$\boxed{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash_p e' \downarrow \tau, \Gamma'}$

By IH on (2), there are e', Φ, Γ'' so that

(5) $|e'| = e$

(6) $\Theta; \Delta \vDash \Phi$

(7) $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \Rightarrow \Phi, \Gamma''$

Goals follow by Theorem 8.22 on (7)

□

APPENDIX B

Theorem 2.1 (Admissible structural rules).

- *Resource Weakening: Write $g \geq f$ for the coefficient-wise partial order on resource terms ($a_1x_1 + a_2x_2 + \ldots + \ell \geq b_1x_1 + b_2x_2 + \ldots + \ell'$ iff $a_i \geq b_i$ for all i and $\ell \geq \ell'$). Then if $\Gamma \vdash_f M : A$ and $g \geq f$ then $\Gamma \vdash_g M : A$.*
- *Variable Weakening: If $\Gamma \vdash_f M : A$ and y does not occur in Γ, then $\Gamma, y : B \vdash_{f+0y} M : A$.*
- *Substitution: If $\Gamma \vdash_f M : A$ and $\Gamma, x : A \vdash_g N : B$, then $\Gamma \vdash_{g[f/x]} N[M/x] : B$*

Proof. All follow by straightforward induction on judgments. □

Theorem 2.2 (Fusion Laws).

(1) $!^{k_1k_2}_{\ell_1+k_1\cdot\ell_2}A \dashv\vdash !^{k_1}_{\ell_1}!^{k_2}_{\ell_2}A$

(2) $!^{k}_{\ell_1+\ell_2}(A \otimes B) \dashv\vdash !^{k}_{\ell_1}A \otimes !^{k}_{\ell_1}B$

(3) $!^{k}_{\ell}(A \oplus B) \dashv\vdash !^{k}_{\ell}A \oplus !^{k}_{\ell}B$

Proof. We present terms going in both directions for each caes.

(1) $x :!^{k_1k_2}_{\ell_1+k_1\ell_2} \vdash_x \texttt{transfer}_1\, !^{k_1k_2}_{\ell_1+k_1\ell_2}\, y = x\ \texttt{to}\ \texttt{save}^{k_1}_{\ell_1}\ (\texttt{save}^{k_2}_{\ell_2}\ y) :!^{k_1}_{\ell_1}!^{k_2}_{\ell_2}A$ and $x :!^{k_1}_{\ell_1}!^{k_2}_{\ell_2}A \vdash_x \texttt{transfer}_1\, !^{k_1}_{\ell_1}\, y = x\ \texttt{to}\ \texttt{transfer}_{k_1}\, !^{k_2}_{\ell_2}\, z = y\ \texttt{to}\ \texttt{save}^{k_1k_2}_{\ell_1+k_1\ell_2}\ z :!^{k_1k_2}_{\ell_1+k_1\ell_2}$

(2) $x :!^{k}_{\ell_1+\ell_2}(A \otimes B) \vdash_x \texttt{transfer}_1\, !^{k}_{\ell_1+\ell_2}\, y = x\ \texttt{to}\ \texttt{split}_{k'}(y,\, z_1.z_2.(\texttt{save}^{k}_{\ell_1}\ z_1, \texttt{save}^{k}_{\ell_2}\ z_2)) : !^{k}_{\ell_1}A \otimes !^{k}_{\ell_1}B$ and $x :!^{k}_{\ell_1}A \otimes !^{k}_{\ell_2}B \vdash_x \texttt{split}_1(x,\, z_1.z_2.\texttt{transfer}_1\, !^{k}_{\ell_1}\, y_1 = z_1\ \texttt{to}\ \texttt{transfer}_1\, !^{k}_{\ell_2}\, y_2 = z_2\ \texttt{to}\ \texttt{save}^{k}_{\ell_1+\ell_2}\ (y_1, y_2)) :!^{k}_{\ell_1+\ell_2}(A \otimes B)$.

(3) $x :!^{k}_{\ell}(A \oplus B) \vdash_x \texttt{transfer}_1\, !^{k}_{\ell}\, y = x\ \texttt{to}\ \texttt{case}_k\,(y,\, z_1.\texttt{inl}\,(\texttt{save}^{k}_{\ell}\ z_1),\, z_2.\texttt{inr}\,(\texttt{save}^{k}_{\ell}\ z_2)) : !^{k}_{\ell}A \oplus !^{k}_{\ell}B$ and $x :!^{k}_{\ell}A \oplus !^{k}_{\ell}B \vdash_x \texttt{case}_1\,(x,\, z_1.\texttt{transfer}_1\, !^{k}_{\ell}\, y = z_1\ \texttt{to}\ \texttt{save}^{k}_{\ell}\ (\texttt{inl}\, y),\, z_2.\texttt{transfer}_1\, !^{k}_{\ell}\, y = z_2\ \texttt{to}\ \texttt{save}^{k}_{\ell}\ (\texttt{inr}\, y)) :!^{k}_{\ell}(A \oplus B)$

□

Theorem 2.3 (Preservation Bound). *If $\cdot \vdash_a M : A$ and $M \downarrow^{(n,r)} v$, then $a + r \geq 0$ and $\cdot \vdash_{a+r} v : A$.*

Proof. By induction on $M \downarrow v$.

- (Values): Suppose $\cdot \vdash_a v : A$ and $v \downarrow^{(0,0)} v$. Then, $a + 0 \geq 0$ (because $a \geq 0$), and $\cdot \vdash_{a+0=a} v : A$.

- (Tick): Immediate by IH.
- (!-I): Suppose $\cdot \vdash_b \texttt{save}_l^k\ M :!_l^k A$, and $\texttt{save}_l^k\ M \downarrow^{(\_,kr)} \texttt{save}_l^k\ v$. We must show that $b + kr \geq 0$ and that $\cdot \vdash_{b+kr} \texttt{save}_l^k\ v :!_l^k A$. Inverting the rules, we have that $\cdot \vdash_a M : A$ with $ka + l \leq b$, and that $M \downarrow^{(n,r)} v$. By IH, $\cdot \vdash_{a+r} v : A$ with $a + r \geq 0$. Since $k, l \geq 0$, $0 \leq k(a + r) + l = ka + l + kr$, which, since $ka + l \leq b$, is less than or equal to $b + kr$. So, $b + kr \geq 0$ and $\cdot \vdash_{b+kr} \texttt{save}_l^k\ v :!_l^k A$, as required.
- (!-E): For this case, suppose $\cdot \vdash_{k'a+b} \texttt{transfer}_{k'}\ !_l^k\ x = M\ \texttt{to}\ N : C$, and $\texttt{transfer}_{k'}\ !_l^k\ x = M\ \texttt{to}\ N \downarrow^{(\_,k'r_1+r_2)} v$. We want to show that: $k'a+b+k'r_1+r_2 \geq 0$ and $\cdot \vdash_{k'a+b+k'r_1+r_2} v : C$. By inversion, $\cdot \vdash_a M :!_l^k A$, and $x : A \vdash_{b+k'(kx+l)} N : C$, as well as $M \downarrow^{(\_,r_1)} \texttt{save}_l^k\ v_1$ and $N[v_1/x] \downarrow^{(\_,r_2)} v$. By IH, we know that $\cdot \vdash_{a+r_1} \texttt{save}_l^k\ v_1 :!_l^k A$ and $a + r_1 \geq 0$, so by inversion, there is a d such that $kd + l \leq a + r_1$, and $\cdot \vdash_d v_1 : A$, and so substitution gives that $\cdot \vdash_{b+k'(kd+l)} N[v_1/x] : C$. But $kd + l \leq a + r_1$, so by structural weakening we have $\cdot \vdash_{b+k'a+k'r_1} N[v_1/x] : C$. Again by IH, $\cdot \vdash_{b+r_2+k'a+k'r_1} v : C$ and $b + r_2 + k'a + k'r_1 \geq 0$, as required.
- ($\texttt{create}$): Suppose $\cdot \vdash_a \texttt{create}_l\ M : A$ and $\texttt{create}_l\ M \downarrow^{(n,r+l)} v$. Inverting, we have $\cdot \vdash_{a+l} M : A$, and $M \downarrow^{(n,r)} v$. By IH, we have that $a + l + r \geq 0$, and $\cdot \vdash_{a+r+l} v$. But, we wanted to show that $a + r + l \geq 0$ and that $\cdot \vdash_{a+r+l} v : A$, and so we are done.
- ($\texttt{spend}$): Suppose $\cdot \vdash_{a+l} \texttt{spend}_l\ M : A$, and $\texttt{spend}_l\ M \downarrow^{(\_,r-l)} v$. We want to show that $a + l + r - l = a + r \geq 0$, and that $\cdot \vdash_{a+l+r-l=a+r} v$ Inverting, we have that $\cdot \vdash_a M : A$ and $M \downarrow^{(n,r)} v$. By IH, $a + r \geq 0$ and $\cdot \vdash_{a+r} v : A$, as required.
- ($\otimes$-I): For this case, let $\cdot \vdash_{a+b} (M, N) : A \otimes B$, and $(M, N) \downarrow^{(\_,k_1+k_2)} (v_1, v_2)$. We must show that $a + b + k_1 + k_2 \geq 0$, and that $\cdot \vdash_{a+b+k_1+k_2} (v_1, v_2)$ Inverting, we get the four premises $\cdot \vdash_a M : A$, $\cdot \vdash_b N : B$, and $M \downarrow^{(\_,k_1)} v_1$ and $N \downarrow^{(\_,k_2)} v_2$. Using the IH on these two pairs, we get that $a + k_1 \geq 0$, $b + k_2 \geq 0$, $\cdot \vdash_{a+k_1} v_1 : A$, and $\cdot \vdash_{b+k_2} v_2 : B$. Adding the two inequalities and applying $\otimes$-I to the judgments gives the desired result.
- ($\oplus$-E): Suppose $\cdot \vdash_{a+b_1+b_2} \texttt{case}_{k'}\ (M, x.N_1\ , y.N_2) : C$ and $\texttt{case}_{k'}\ (M, x.N_1\ , y.N_2) \downarrow^{(\_,k'r_1+r_2)} v$. Inverting the typing judgment, $\cdot \vdash_a M : A \oplus B$, $x : A \vdash_{b_1+k'x} N_1 : C$ and $y : B \vdash_{b_2+k'} N_2 : C$. Inverting the evaluation judgment gives two symmetric cases, so suppose that $M \downarrow^{(\_,r_1)} \texttt{inl}\, v_1$ and $N_1[v_1/x] \downarrow^{(\_,r_2)} v$. By IH, $\cdot \vdash_{a+r_1} \texttt{inl}\, v_1 : A \oplus B$ and $a + r_1 \geq 0$. So, $\cdot \vdash_{a+r_1} v_1 : A$. By substitution, $\cdot \vdash_{b_1+k'(a+r_1)} N_1[v_1/x] : C$. By IH,

$\cdot \vdash_{k'a+b_1+k'r_1+r_2} v : C$ and $k'a + b_1 + k'r_1 + r_2 \geq 0$. Since $b_2 \geq 0$, by structural weakening, $\cdot \vdash_{k'a+b_1+b_2+k'r_1+r_2} v : C$ and $k'a + b_1 + b_2 + k'r_1 + r_2 \geq 0$, as required.

- ($\multimap$-E): Let $\cdot \vdash_{a+b} M\,N : B$, and $M\,N \downarrow^{(\_,k_1+k_2+k_3)} v$. We want to show that $a + b + k_1 + k_2 + k_3 \geq 0$, and that $\cdot \vdash_{a+b+k_1+k_2+k_3} v : B$. We invert both judgments to get $\cdot \vdash_a M : A \multimap B$, and $\cdot \vdash_b N : A$, and $M \downarrow^{(\_,k_1)} \lambda x.M'$, and $N \downarrow^{(\_,k_2)} v_1$, and that $M'[v_1/x] \downarrow^{(\_,k_3)} v$. Applying the IH to the first evaluation, we have that $\cdot \vdash_{a+k_1} \lambda x.M' : A \multimap B$. Inverting the proof of that judgment, we get that $x : A \vdash_{a+k_1+x} M' : B$. By IH again, $\cdot \vdash_{b+k_2} v_1 : A$, and by substitution, $\cdot \vdash_{a+b+k_1+k_2} M'[v_1/x]$. By IH once more, $a + b + k_1 + k_2 + k_3 \geq 0$, and $\cdot \vdash_{a+b+k_1+k_2+k_3} v : B$, as required.
- ($\mathbb{N}$-E) Suppose $\cdot \vdash_{a+b_1+b_2} \mathtt{nrec}\,(M, N_1, N_2) : C$. By inversion, $\cdot \vdash_a M : \mathbb{N}$, $\cdot \vdash_{b_1} N_1 : 1 \multimap C$, and $\cdot \vdash_{b_2} N_2 :!_0^\infty(\mathbb{N} \otimes (1 \multimap C) \multimap C)$. We have two evaluation cases to consider.
  - Suppose $\mathtt{nrec}\,(M, N_1, N_2) \downarrow^{(\_,r_1+r_2+r_3+r_3)} v$ by way of $M \downarrow^{(\_,r_1)} 0 : \mathbb{N}$, $N_1 \downarrow^{(\_,r_2)} \lambda x.N_1'$, $N_2 \downarrow^{(\_,r_3)}$ _, and $N_1'[()/x] \downarrow^{(\_,r_4)} v$. Then, by IH, we have the following:
    * $\cdot \vdash_{a+r_1} 0 : \mathbb{N}$, and $a + r_1 \geq 0$
    * $\cdot \vdash_{b_1+r_2} \lambda x.N_1' : 1 \multimap C$, $b_1 + r_2 \geq 0$.
    * $b_2 + r_3 \geq 0$

    Since $\cdot \vdash_0 () : 1$, $\cdot \vdash_{b_1+r_1} N_1'[()/x] : C$. By IH, $\cdot \vdash_{b_1+r_2+r_4} v : C$. By structural weakening, $\cdot \vdash_{a+b_1+b_2+r_1+r_2+r_3+r_4} v : C$, as required.
  - Suppose $\mathtt{nrec}\,(M, N_1, N_2) \downarrow^{(\_,r_1+r_2+r_3+r_3)} v$ by way of $M \downarrow^{(\_,r_1)} S(v_1) : \mathbb{N}$, $N_1 \downarrow^{(\_,r_2)} \lambda x.N_1'$, $N_2 \downarrow^{(\_,r_3)} \mathtt{save}_0^\infty\ (\lambda x.N_2')$, and $N_2'[(v_1, \lambda z.\mathtt{nrec}\,(v_1, \lambda x.N_1', \mathtt{save}_0^\infty\ (\lambda x.N_2')))/x] \downarrow^{(\_,r_4)} v$. Then, by IH we have:
    * $\cdot \vdash_{a+r_1} S(v_1) : \mathbb{N}$, $a + r_1 \geq 0$
    * $\cdot \vdash_{b_1+r_2} \lambda x.N_1' : 1 \multimap C$, $b_1 + r_2 \geq 0$
    * $\cdot \vdash_{b_2+r_3} \mathtt{save}_0^\infty\ (\lambda x.N_2') :!_0^\infty(\mathbb{N} \otimes (1 \multimap C) \multimap C)$, and $b_2 + r_3 \geq 0$.

    By $\mathbb{N}$-strengthening, $\cdot \vdash_0 S(v_1) : \mathbb{N}$, and so $\cdot \vdash_0 v_1 : \mathbb{N}$. Since $\cdot \vdash_{b_2+r_3} \mathtt{save}_0^\infty\ (\lambda x.N_2') : !_0^\infty(\mathbb{N} \otimes (1 \multimap C) \multimap C)$, there is a $c \geq 0$ so that $\infty \cdot c \leq b_2 + r_3$ with $\cdot \vdash_c \lambda x.N_2' : \mathbb{N} \otimes (1 \multimap C) \multimap C$. Then, $\cdot \vdash_{b_1+r_3+\infty \cdot c} \mathtt{nrec}\,(v_1, \lambda x.N_1', \mathtt{save}_0^\infty\ (\lambda x.N_2')) : C$, and so $\cdot \vdash_{a+b_1+r_1+r_2+\infty \cdot c} (v_1, \lambda z.\mathtt{nrec}\,(v_1, \lambda x.N_1', \mathtt{save}_0^\infty\ (\lambda x.N_2'))) : \mathbb{N} \otimes (1 \multimap C)$. Thus, since $x : \mathbb{N} \otimes (1 \multimap C) \vdash_{x+c} N_2' : C$,

$$\cdot \vdash_{a+b_1+r_1+r_2+\infty \cdot c} N_2'[(v_1, \lambda z.\mathtt{nrec}\,(v_1, \lambda x.N_1', \mathtt{save}_0^\infty\ (\lambda x.N_2')))/x] : C$$

since $\infty \cdot c + c = \infty \cdot c$. So, by IH, $\cdot \vdash_{a+b_1+r_1+r_2+\infty\cdot c+r_4} v : C$, and because $\infty \cdot c \le b_2 + r_3$, we have by weakening that $\cdot \vdash_{a+b_1+b_2+r_1+r_2+r_3+r_4} v : C$ as required.

- ($[A]$-E) Suppose $\cdot \vdash_{a+b_1+b_2} \texttt{lrec}\,(M, N_1, N_2) : C$. Then, $\cdot \vdash_a M : A$, $\cdot \vdash_{b_1} N_1 : 1 \to C$, and $\cdot \vdash_{b_2} : N_2 :!_0^{\infty}(A \otimes (\,[A]\,\&C) \multimap C)$. We have two evaluation cases to consider.

  Firstly, suppose $\texttt{lrec}\,(M, N_1, N_2) \downarrow^{(\_, r_1+r_2+r_3+r_4)} v$ by way of $M \downarrow^{(\_, r_1)} v$, $N_1 \downarrow^{(\_, r_2)} \lambda x.N_1'$, $N_2 \downarrow^{(\_, r_3)} v'$, and $N_1'[()/x] \downarrow^{(\_, r_4)} v$. Then, by IH, $a + r_1 \ge 0$, which means that $\cdot \vdash_{a+r_1} () : 1$. By IH, $\cdot \vdash_{b_1+r_2} \lambda x.N_1'$ and $b_1 + r_2 \ge 0$. So, by inversion and then substitution, $\cdot \vdash_{a+b_1+r_1+r_2} N_1'[()/x] : C$. By IH, $b_2 + r_3 \ge 0$, so by weakening, $\cdot \vdash_{a+b_1+b_2+r_1+r_2+r_3} N_1'[()/x]$. Finally, by IH, $a + b_1 + b_2 + r_1 + r_2 + r_3 + r_4 \ge 0$ and $\cdot \vdash_{a+b_1+b_2+r_1+r_2+r_3+r_4} v : C$.

  Now, suppose $\texttt{lrec}\,(M, N_1, N_2) \downarrow^{(\_, r_1+r_2+r_3+r_4)} v$ by way of $M \downarrow^{(\_, r_1)} v_1 :: v_2$, $N_1 \downarrow^{(\_, r_2)} \lambda x.N_1'$, $N_2 \downarrow^{(\_, r_3)} \texttt{save}_0^{\infty}\,(\lambda x.N_2')$, and $N_2'[(v_1, \langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^{\infty}\,(\lambda x.N_2'))\rangle)/x] \downarrow^{(\_, r_4)} v$. By IH, $\cdot \vdash_{a+r_1} v_1 :: v_2 : [A]$ and $a + r_1 \ge 0$. By inversion, there are $d_1, d_2 \ge 0$ so that $a + r_1 = d_1 + d_2$ and $\cdot \vdash_{d_1} v_1 : A$ and $\cdot \vdash_{d_2} v_2 : [A]$. By two more applications of the IH, $\cdot \vdash_{b_1+r_2} \lambda x.N_1' : 1 \multimap C$, $\cdot \vdash_{b_2+r_3} \texttt{save}_0^{\infty}\,(\lambda x.N_2') :!_0^{\infty}(A \otimes (\,[A]\,\&C) \multimap C)$, with $b_1 + r_2 \ge 0$ and $b_2 + r_3 \ge 0$. By inversion, there is some $c \ge 0$ with $\infty \cdot c \le b_2 + r_3$ such that $\cdot \vdash_c \lambda x.N_2' : A \otimes (\,[A]\,\&C) \multimap C$. Next,

$$\frac{\cdot \vdash_{d_2} v_2 : [A] \quad \cdot \vdash_{b_1+r_2} \lambda x.N_1' : 1 \multimap C \quad \cdot \vdash_{\infty\cdot c} \texttt{save}_0^{\infty}\,(\lambda x.N_2') :!_0^{\infty}(A \otimes (\,[A]\,\&C) \multimap C)}{\cdot \vdash_{d_2+b_1+r_2+\infty\cdot c} \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^{\infty}\,(\lambda x.N_2')) : C}$$

  then, with $\cdot \vdash_{d_2+b_1+r_2+\infty\cdot c} v_2 : [A]$, we have that

$$\cdot \vdash_{d_2+b_1+r_2+\infty\cdot c} \langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^{\infty}\,(\lambda x.N_2'))\rangle : [A]\,\&C$$

  and since $\cdot \vdash_{d_1} v_1 : A$,

$$\cdot \vdash_{a+r_1+b_1+r_2+\infty\cdot c} (v_1, \langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^{\infty}\,(\lambda x.N_2'))\rangle) : A \otimes (\,[A]\,\&C)$$

  and so by substitution, and using the fact that $c + \infty \cdot c = \infty \cdot c$, $\cdot \vdash_{a+b_1+r_1+r_2+\infty\cdot c} N_2'[(v_1, \langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^{\infty}\,(\lambda x.N_2'))\rangle)/x] : C$. By weakening, since $\infty \cdot c \le b_2 + r_3$, $\cdot \vdash_{a+b_1+b_2+r_1+r_2+r_3} N_2'[(v_1, \langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^{\infty}\,(\lambda x.N_2'))\rangle)/x] : C$. Finally, by IH, $\cdot \vdash_{a+b_1+b_2+r_1+r_2+r_3+r_4} v : C$, and $a + b_1 + b_2 + r_1 + r_2 + r_3 + r_4 \ge 0$, as required.

- (&-I): Immediate.

- (&-E): By symmetry, it suffices to only consider the π_1 case. Let $\cdot \vdash_a \pi_1 M : A$ and $\pi_1 M \downarrow^{(-,r_1+r_2)} v$. By inversion, we have that $\cdot \vdash_a M : A\&B$, and that $M \downarrow^{(-,r_1)} \langle N_1, N_2\rangle$ and $N_1 \downarrow^{(-,r_2)}$. We must show that $\cdot \vdash_{a+r_1+r_2} v : A$, and that $a + r_1 + r_2 \geq 0$. By IH, $\cdot \vdash_{a+r_1} \langle N_1, N_2\rangle : A\&B$. Inverting this, we get that $\cdot \vdash_{a+r_1} N_1 : A$, and so again by IH, $\cdot_{a+r_1+r_2} v : A$, and $a + r_1 + r_2 \geq 0$, as required.

□

THEOREM 2.4. *If v is a value, and $v \downarrow^{(n,r)} v$, then $n = r = 0$.*

PROOF. By inspection of cases. □

THEOREM 2.5 (Resource strengthening for $\mathbb{N}$). *If $\cdot \vdash_a v : \mathbb{N}$, then $\cdot \vdash_0 v : \mathbb{N}$*

PROOF. By canonical forms, $v = \overline{n}$, proceed by induction on n. □

THEOREM 3.1 (Extraction Preserves Types). *If $\Gamma \vdash_a M : A$ then $\langle\!\langle \Gamma \rangle\!\rangle \vdash \|M\| : \|A\|$*

PROOF. By induction on $\Gamma \vdash_f M : A$ □

THEOREM 3.3 (Weakening).

(1) If $M \sqsubseteq^{A,a} E$, and $E \leq_{\|A\|} E'$, then $M \sqsubseteq^{A,a} E'$

(2) If $v \sqsubseteq^{A,a}_{\mathtt{val}} E$, and $E \leq_{\langle\!\langle A \rangle\!\rangle} E'$, then $v \sqsubseteq^{A,a}_{\mathtt{val}} E'$

PROOF. We prove 1 and 2 simultaneously by induction on A.

(1) Suppose $M \sqsubseteq^{A,a} E$, and $E \leq_{\mathbb{C}\times\langle\!\langle A \rangle\!\rangle} E'$. We need to show that $M \sqsubseteq^{A,a} E'$. Suppose $M \downarrow^{(n,r)} v$. We need to show:
   - $n \leq E'_c - r$
   - $v \sqsubseteq^{A,a+r}_{\mathtt{val}} E'_p$

   But, since $M \sqsubseteq^{A,a} E$
   - $n \leq E_c - r$
   - $v \sqsubseteq^{A,a+r}_{\mathtt{val}} E_p$

   so, it suffices to show that $E_c \leq_{\mathbb{C}} E'_c$ and $E_p \leq_{\langle\!\langle A \rangle\!\rangle} E'_p$, which is true by the $\pi_1(-)$ and $\pi_2(-)$ congruences, recalling that $(-)_c$ and $(-)_p$ are simply π_1 and π_2.

(2) Let $E \leq_{\langle\!\langle A \rangle\!\rangle} E'$. We have a few cases to consider.

(!) Suppose $\mathtt{save}_l^k\ v \sqsubseteq_{\mathtt{val}}^{!_l^k A,a} E$. We must show that $\mathtt{save}_l^k\ v \sqsubseteq_{\mathtt{val}}^{!_l^k A,a} E'$. We know that there is a $d \geq 0$ such that $ka + l \leq d$, and $v \sqsubseteq_{\mathtt{val}}^{A,d} E$. So, by IH, $v \sqsubseteq_{\mathtt{val}}^{A,d} E'$, and hence $\mathtt{save}_l^k\ v \sqsubseteq_{\mathtt{val}}^{!_l^k A,a} E'$, as required.

($\multimap$) Suppose $\lambda x.M \sqsubseteq_{\mathtt{val}}^{A\multimap B,a} E$. We need to show that $\lambda x.M \sqsubseteq_{\mathtt{val}}^{A\multimap B,a} E'$. Let $v \sqsubseteq_{\mathtt{val}}^{A,b} E_v$. Then, $M[v/x] \sqsubseteq^{B,a+b} E\ E_v$. Using the application congruence and 1, $M[v/x] \sqsubseteq^{B,a+b} E'\ E_v$. Since v, b, E_v were chosen arbitrarily, $\lambda x.M \sqsubseteq_{\mathtt{val}}^{A\multimap B,a} E'$ as required.

($\otimes$) Suppose $(v_1, v_2) \sqsubseteq_{\mathtt{val}}^{A_1\otimes A_2,a} E$. Then, there are a_1, a_2 such that $a_1 + a_2 = a$, and $v_i \sqsubseteq_{\mathtt{val}}^{A_i,a_i} \pi_i E$, and so by π_i-congruence and the IH, $v_i \sqsubseteq_{\mathtt{val}}^{A_i,a_i} \pi_i E'$, so $(v_1, v_2) \sqsubseteq_{\mathtt{val}}^{A_1\otimes A_2,a} E'$, as required.

($[A]$) Both cases are immediate by transitivity.

($\mathbb{N}$) Both cases are immediate by transitivity.

($\oplus$) Both cases are immediate by transitivity.

($A\&B$) Suppose $\langle M_1, M_2\rangle \sqsubseteq_{\mathtt{val}}^{A_1\&A_2,a} E$. Then, for $i \in \{1,2\}$, $M_i \sqsubseteq^{A_i,a} \pi_i E$. By π_i-congruence, $\pi_i E \leq_{\|A\|} \pi_i E'$, and so by IH from 1, we know that $M_i \sqsubseteq^{A_i,a} \pi_i E'$, and are done.

□

Theorem 3.4 (Credit Weakening). *If $a_1 \leq a_2$, then:*

(1) If $M \sqsubseteq^{A,a_1} E$, then $M \sqsubseteq^{A,a_2} E$

(2) If $v \sqsubseteq_{val}^{A,a_1} E$, then $v \sqsubseteq_{val}^{A,a_2} E$

Proof. We prove the two claims simultaneously.

(1) Suppose $M \sqsubseteq^{A,a_1} E$. To show $M \sqsubseteq^{A,a_2} E$, suppose $M \Downarrow^{(n,r)} v$. We must show that such that

- $n \leq E_c - r$
- $v \sqsubseteq_{\mathtt{val}}^{A,a_2+r} E_p$

But, since $M \sqsubseteq^{A,a_1} E$, we have

- $n \leq E_c - r$
- $v \sqsubseteq_{\mathtt{val}}^{A,a_1+r} E_p$

Since $a_1 \leq a_2$, $a_1 + r \leq a_2 + r$, so we are done by (2).

(2) By lexicographic induction on first A and then the size of v.

(!) Let $\mathtt{save}_l^k\ v \sqsubseteq_{\mathtt{val}}^{!_l^k A, a_1} E$. Then, there is a $d \geq 0$ such that $kd + l \leq a_1$ and $v \sqsubseteq_{\mathtt{val}}^{A,d} E$. But $kd + l \leq a_1 \leq a_2$, and so $\mathtt{save}_l^k\ v \sqsubseteq_{\mathtt{val}}^{!_l^k A, a_2} E$

($\multimap$) Let $\lambda x.M \sqsubseteq_{\mathtt{val}}^{A \multimap B, a_1} E$. Suppose $v \sqsubseteq_{\mathtt{val}}^{A,b} E'$. Then, $M[v/x] \sqsubseteq^{B, a_1 + b} E\ E'$, and so by (1), $M[v/x] \sqsubseteq^{B, a_2 + b} E\ E'$. Since v was chosen arbitrarily, $\lambda x.M \sqsubseteq_{\mathtt{val}}^{A \multimap B, a_2} E$, as required.

($\otimes$) Let $(v_1, v_2) \sqsubseteq_{\mathtt{val}}^{A_1 \otimes A_2, a_1} E$. Then, there are b_1, b_2 with $b_1 + b_2$ such that $b_1 + b_2 = a_1$, and $v_i \sqsubseteq_{\mathtt{val}}^{A_i, a_1} \pi_i E$ for $i \in \{1, 2\}$. By the IH on v_1, we have that $v_1 \sqsubseteq_{\mathtt{val}}^{A_1, b_1 + a_2 - a_1} \pi_1 E$, and so $(v_1, v_2) \sqsubseteq_{\mathtt{val}}^{A_1 \otimes A_2, a_2} E$, as required.

($\mathtt{[}A\mathtt{]}$) The empty case is immediate. Suppose $v_1 :: v_2 \sqsubseteq_{\mathtt{val}}^{\mathtt{[}A\mathtt{]}, a_1} E$. Then, there are E_1, E_2, b_1, b_2 such that $b_1 + b_2 = a_1$, $E_1 :: E_2 \leq E$, $v_1 \sqsubseteq_{\mathtt{val}}^{A, b_1} E_1$, and $v_1 \sqsubseteq_{\mathtt{val}}^{\mathtt{[}A\mathtt{]}, b_2} E_2$. By IH, $v_1 \sqsubseteq_{\mathtt{val}}^{A, b_1 + a_2 - a_1} E_1$, and so $v_1 :: v_2 \sqsubseteq_{\mathtt{val}}^{\mathtt{[}A\mathtt{]}, a_2} E$.

($\mathbb{N}$) The zero case is immediate. Suppose $S(v) \sqsubseteq_{\mathtt{val}}^{\mathbb{N}, a_1} E$. Then, there is E' such that $S(E') \leq E$, and $v \sqsubseteq_{\mathtt{val}}^{\mathbb{N}, a_1} E'$. Since v is a smaller term than $S(v)$, we can apply the IH to see that $v \sqsubseteq_{\mathtt{val}}^{\mathbb{N}, a_2} E'$, and so $S(v) \sqsubseteq_{\mathtt{val}}^{\mathbb{N}, a_2} E$, as desired.

($\oplus$) The two cases are symmetric, so we present only one. Suppose $\mathtt{inl}\, v \sqsubseteq_{\mathtt{val}}^{A \oplus B, a_1} E$. Then we have E' such that $\mathtt{inl}\, E' \leq E$, and $v \sqsubseteq_{\mathtt{val}}^{A, a_1} E'$, which, by IH, means that $v \sqsubseteq_{\mathtt{val}}^{A, a_2} E'$, and so $\mathtt{inl}\, v \sqsubseteq_{\mathtt{val}}^{A \oplus B, a_2} E$.

($A \& B$) Immediate by IH.

□

THEOREM 3.5 ($\mathbb{N}$-Recursor). *If* $\lambda x.N_1' \sqsubseteq_{val}^{1 \multimap C, c_3} E_1$, $\lambda x.N_2' \sqsubseteq_{val}^{\mathbb{N} \otimes (1 \multimap C) \multimap C, d} E_2$ *with* $d \geq 0$, *then* $\forall n \geq 0$, *if* $\overline{n} \sqsubseteq_{val}^{\mathbb{N}, 0} E$, *then* $\mathtt{nrec}\,(\overline{n}, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')) \sqsubseteq^{C, c_3 + \infty \cdot d} \mathtt{nrec}\,(E, E_1, \lambda p.E_2\ (\pi_1 p, \lambda z.\pi_2 p))$

PROOF. Proceed by induction on n.

For notational simplicity, let $E_2^* = \lambda p.E_2(\pi_1 p, (\lambda z.\pi_2 p))$

- ($n = 0$): To show $\mathtt{nrec}\,(0, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')) \sqsubseteq^{C, c_3 + \infty \cdot d} \mathtt{nrec}\,(E, E_1, E_2^*)$, suppose that $\mathtt{nrec}\,(0, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')) \downarrow^{(n,r)} v$ by way of $N_1'[()/x] \downarrow^{(n,r)}$.

  We must show that :

  - $n \leq \mathtt{nrec}\,(E, E_1, E_2^*)_c - r$
  - $v \sqsubseteq_{\mathtt{val}}^{C, c_3 + \infty \cdot d} \mathtt{nrec}\,(E, E_1, E_2^*)_p$

We know $N_1[()/x] \sqsubseteq^{C,c_3} E_1\ ()$, since $() \sqsubseteq^{1,0}_{\mathtt{val}} ()$, and so:

- $n \leq (E_1\ ())_c - r$
- $v \sqsubseteq^{C,c_3}_{\mathtt{val}} (E_1\ ())_p$

Since $0 \sqsubseteq^{\mathbb{N},0}_{\mathtt{val}} E$, $0 \leq_{\mathbb{N}} E$, and so $E_1\ () \leq \mathtt{nrec}\,(0, E_1, E_2^*) \leq \mathtt{nrec}\,(E, E_1, E_2^*)$.

- $(n > 0)$: Suppose that $\mathtt{nrec}\,(S(\overline{n}), \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')) \downarrow^{(n',r)} v$ by way of

$$N_2'[(\overline{n}, \lambda z.\mathtt{nrec}\,(\overline{n}, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')))] \downarrow^{(n',r)} v.$$

We must show that:

- $n' \leq \mathtt{nrec}\,(E, E_1, E_2^*)_c - r$;
- $v \sqsubseteq^{C,c_3+\infty\cdot d+r}_{\mathtt{val}} \mathtt{nrec}\,(E, E_1, E_2^*)_p$.

Since $S(\overline{n}) \sqsubseteq^{\mathbb{N},0}_{\mathtt{val}} E$, there is an E' so that $S(E') \leq_{\mathbb{N}} E$, and $\overline{n} \sqsubseteq^{\mathbb{N},0}_{\mathtt{val}} E'$. For notational convenience, let $E^* = (E', \mathtt{nrec}\,(E', E_1, E_2^*))$. Note that $E' \leq \pi_1 E^*$, and that $\mathtt{nrec}\,(E', E_1, E_2^*) \leq \pi_2 E^*$. By IH, $\mathtt{nrec}\,(\overline{n}, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')) \sqsubseteq^{C,c_3+\infty\cdot d} \mathtt{nrec}\,(E', E_1, E_2^*)$, and thus by weakening $\mathtt{nrec}\,(\overline{n}, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')) \sqsubseteq^{C,c_3+\infty\cdot d} \pi_2 E^*$. For some variable z not free in the term on the left, $\lambda z.\mathtt{nrec}\,(\overline{n}, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')) \sqsubseteq^{1\multimap C,c_3+\infty\cdot d}_{\mathtt{val}} \lambda z.\pi_2 E^*$, and so $(\overline{n}, \lambda z.\mathtt{nrec}\,(\overline{n}, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2'))) \sqsubseteq^{\mathbb{N}\otimes(1\multimap C),c_3+\infty\cdot d}_{\mathtt{val}} (\pi_1 E^*, \lambda z.\pi_2 E^* s)$, and since $\lambda x.N_2' \sqsubseteq^{\mathbb{N}\times(1\multimap C)\multimap C,d}_{\mathtt{val}} E_2$, using the fact that $\infty\cdot d + d = \infty\cdot d$

$$N_2'[(\overline{n}, \lambda z.\mathtt{nrec}\,(\overline{n}, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')))/x] \sqsubseteq^{C,c_3+\infty\cdot d} E_2\ (\pi_1 E^*, \lambda z.\pi_2 E^*)$$

but,

$$\begin{aligned} E_2\ (\pi_1 E^*, \lambda z.\pi_2 E^*) &\leq (\lambda p.E_2\ (\pi_1 p, \lambda z.\pi_2 p))E^* \\ &= E_2^*(E', \mathtt{nrec}\,(E', E_1, E_2^*)) \\ &\leq \mathtt{nrec}\,(S(E'), E_1, E_2^*) \\ &\leq \mathtt{nrec}\,(E, E_1, E_2^*) \end{aligned}$$

and so we are done by weakening.

□

Theorem 3.6 ($[A]$-Recursor). *If $\lambda x.N_1' \sqsubseteq^{1\multimap C,c_1}_{val} E_1$ and $\lambda x.N_2' \sqsubseteq^{A\otimes([A]\&C)\multimap C,c_2}_{val} E_2$, then for all values $\cdot \vdash_d v : [A]$ such that $v \sqsubseteq^{[A],d}_{val} E$, we have that*

$$\mathtt{lrec}(v, \lambda x.N_1', \mathtt{save}_0^{\infty}\ (\lambda x.N_2')) \sqsubseteq^{C,c_1+d+\infty\cdot c_2} \mathtt{lrec}(E, E_1, \lambda x.E_2(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x)))$$

PROOF. We proceed by induction on the derivation of $\cdot \vdash_d v : \texttt{[}A\texttt{]}$. First, suppose $v = []$. To show that $\texttt{lrec}\,([], \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2')) \sqsubseteq^{C,c_1+d+\infty\cdot c_2} \texttt{lrec}\,(E, E_1, \lambda x.E_2(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x)))$, assume that $\texttt{lrec}\,([], \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2')) \downarrow^{(n,r)} v$. By inversion, it was by way of $N_1'[()/x] \downarrow^{(n,r)} v$. It suffices to show

- $n \le \texttt{lrec}\,(E, E_1, \lambda x.E_2(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x)))_c - r$
- $v \sqsubseteq_{\texttt{val}}^{C,c_1+d+\infty\cdot c_2+r} \texttt{lrec}\,(E, E_1, \lambda x.E_2(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x)))_p$

Since $() \le_1 ()$, $() \sqsubseteq_{\texttt{val}}^{1,d} ()$, so $N_1'[()/x] \sqsubseteq^{C,c_1+d} E_1\ ()$, and so

- $n \le (E_1\ ())_c - r$
- $v \sqsubseteq_{\texttt{val}}^{C,c_1+d+r} (E_1\ ())_p$

But, $\infty \cdot c_2 > 0$ since $c_2 > 0$, and so by credit weakening, $v \sqsubseteq_{\texttt{val}}^{C,c_1+d+\infty\cdot c_2+r} (E_1\ ())_p$. Note that, by assumption, $[] \sqsubseteq_{\texttt{val}}^{1,d} E$, which means that $[] \le_{\texttt{[}\langle\!\langle A\rangle\!\rangle\texttt{]}} E$. So,

$$E_1\ () \le \texttt{lrec}\,([], E_1, \lambda x.E_2(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x))) \le \texttt{lrec}\,(E, E_1, \lambda x.E_2(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x)))$$

and so we are done by weakening.

Otherwise, suppose $v = v_1 :: v_2$. To show that $\texttt{lrec}\,(v_1 :: v_2, \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2')) \sqsubseteq^{C,c_1+d+\infty\cdot c_2}$ $\texttt{lrec}\,(E, E_1, \lambda x.E_2(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x)))$, suppose $\texttt{lrec}\,(v_1 :: v_2, \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2')) \downarrow^{(n,r)} v$. By inversion, it was by $N_2'[(v_1, \langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2'))\rangle)/x] \downarrow^{(n,r)} v$. It suffices to show:

- $n \le \texttt{lrec}\,(E, E_1, \lambda x.E_2(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x)))_c - r$
- $v \sqsubseteq_{\texttt{val}}^{C,c_1+d+\infty\cdot c_2+r} \texttt{lrec}\,(E, E_1, \lambda x.E_2(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x)))_p$

Since $v_1 :: v_2 \sqsubseteq_{\texttt{val}}^{\texttt{[}A\texttt{]},d} E$, there are $d_1, d_2 \ge 0$ such that $d_1 + d_2 = d$, along with E',E'' such that $v_1 \sqsubseteq_{\texttt{val}}^{A,d_1} E'$ and $v_2 \sqsubseteq_{\texttt{val}}^{\texttt{[}A\texttt{]},d_2} E''$, and $E' :: E'' \le E$

By IH, $\texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2')) \sqsubseteq^{C,c_1+d_2+\infty\cdot c_2} \texttt{lrec}\,(E'', E_1, \ldots)$. Since $v_2 \sqsubseteq_{\texttt{val}}^{\texttt{[}A\texttt{]},d_2}$ E'', $v_2 \sqsubseteq^{\texttt{[}A\texttt{]},d_2} (0, E'')$, and since $c_1+\infty\cdot c_2 \ge 0$, we have by credit weakening that $v_2 \sqsubseteq^{\texttt{[}A\texttt{]},c_1+d_2+\infty\cdot c_2}$ $(0, E'')$. So, $\langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2'))\rangle \sqsubseteq_{\texttt{val}}^{\texttt{[}A\texttt{]}\&C,c_1+d_2+\infty\cdot c_2} ((0, E''), \texttt{lrec}\,(E'', E_1, \ldots))$. Further, using the fact that $d_1 + d_2 = d$,

$$\begin{aligned}&(v_1, \langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2'))\rangle) \sqsubseteq_{\texttt{val}}^{A\otimes(\texttt{[}A\texttt{]}\&C),c_1+d+\infty\cdot c_2}\\ &(E', ((0, E''), \texttt{lrec}\,(E'', E_1, \ldots)))\end{aligned}$$

Thus, since $\lambda x.N_2' \sqsubseteq_{\mathtt{val}}^{A\otimes([A]\&C)\multimap C,c_2} E_2$, we have (using the fact that $c_2 + \infty \cdot c_2 = \infty \cdot c_2$)

$$N_2'[(v_1, \langle v_2, \mathtt{lrec}\,(v_2, \lambda x.N_1', \mathtt{save}_0^\infty\;(\lambda x.N_2'))\rangle)/x] \sqsubseteq^{C,c_1+d+\infty\cdot c_2} E_2\;(E', ((0,E''), \mathtt{lrec}\,(E'', E_1, \ldots)))$$

By definition, this means that

- $n \le (E_2\;(E', ((0,E''), \mathtt{lrec}\,(E'', E_1, \ldots))))_c - r$
- $v \sqsubseteq_{\mathtt{val}}^{c_1+d+\infty\cdot c_2+r} (E_2\;(E', ((0,E''), \mathtt{lrec}\,(E'', E_1, \ldots))))_p$

We then compute:

$$E_2\;(E', ((0,E''), \mathtt{lrec}\,(E'', E_1, \ldots)))$$
$$\le (\lambda x.E_2(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x)))\;(E', (E'', \mathtt{lrec}\,(E'', E_1, \lambda x.E_2(\pi_1 x, ((0,\pi_1\pi_2 x), \pi_2\pi_2 x)))))$$
$$\le \mathtt{lrec}\,(E' :: E'', E_1, \lambda x.E_2(\pi_1 x, ((0,\pi_1\pi_2 x), \pi_2\pi_2 x))s)$$
$$\le \mathtt{lrec}\,(E, E_1, \lambda x.E_2(\pi_1 x, ((0,\pi_1\pi_2 x), \pi_2\pi_2 x)))$$

and hence we are done by weakening. □

THEOREM 3.7 (Bounding Theorem). *If* $\Gamma \vdash_f M : A$, *then* $M \sqsubseteq^A \|M\|$

PROOF. By induction on $\Gamma \vdash_f M : A$.

(!-I) Let $\Gamma \vdash_g \mathtt{save}_l^k\;M :!_l^k A$. By inversion, we have $\Gamma \vdash_f M : A$ with $kf + l \le g$. Let $\theta \sqsubseteq_{\mathtt{sub}}^{\Gamma,\sigma} \Theta$. To show $\mathtt{save}_l^k\;M[\theta] \sqsubseteq^{!_l^k A, g[\sigma]} (k\,\|M\|\,[\Theta]_c, \|M\|\,[\Theta]_p)$, it suffices to show $\mathtt{save}_l^k\;M[\theta] \sqsubseteq^{!_l^k A, kf[\sigma]+l} (k\,\|M\|\,[\Theta]_c, \|M\|\,[\Theta]_p)$ by credit weakening. So, let $\mathtt{save}_l^k\;M \downarrow^{(n,kr)} \mathtt{save}_l^k\;v$ by way of $M \downarrow^{(n,r)} v$. It suffices to show, using the fact that $kf[\sigma] + l + kr = k(f[\sigma] + r) + l$

– $n \le k\,\|M\|\,[\Theta]_c - kr$

– $\mathtt{save}_l^k\;v \sqsubseteq_{\mathtt{val}}^{!_l^k A, k(f[\sigma]+r)+l} \|M\|\,[\Theta]_p$

To show $\mathtt{save}_l^k\;v \sqsubseteq_{\mathtt{val}}^{!_l^k A, k(f[\sigma]+r)+l} \|M\|\,[\Theta]_p$, it suffices to provide $d \ge 0$ such that $kd + l \le k(f[\sigma] + r) + l$, and $v \sqsubseteq_{\mathtt{val}}^{A,d} \|M\|\,[\Theta]_p$. By IH, $M[\theta] \sqsubseteq^{A,f[\sigma]} \|M\|\,[\Theta]$, which means that

– $n \le \|M\|\,[\Theta]_c - r$

– $v \sqsubseteq_{\mathtt{val}}^{A,f[\sigma]+r} \|M\|\,[\Theta]_p$

So, $d = f[\sigma] + r$, and the inequality $n \le kn \le k\,\|M\|\,[\Theta]_c - kr$ follows by multiplying the above one by k (since $k \ge 1$).

(!-E) Let $\Gamma \vdash_{k'f+g} \texttt{transfer}_{k'}\, !_l^k\, x = M \texttt{ to } N : C$. By inversion, we have that $\Gamma \vdash_f M : !_l^k A$, as well as $\Gamma, x : A \vdash_{g+k'(kx+l)} N : C$. Suppose $\theta \sqsubseteq_{\texttt{sub}}^{\Gamma,\sigma} \Theta$. We need to show that $\texttt{transfer}_{k'}\, !_l^k\, x = M[\theta] \texttt{ to } N[\theta] \sqsubseteq^{C,k'f[\sigma]+g[\sigma]} \|M\|[\Theta]_c +_c \|N\|[\Theta, \|M\|[\Theta]_p/x]$. Suppose $\texttt{transfer}_{k'}\, !_l^k\, x = M[\theta] \texttt{ to } N[\theta] \downarrow^{(n_1+n_2, k'r_1+r_2)} v$. By inversion, it was by $M[\theta] \downarrow^{(n_1,r_1)} \texttt{save}_l^k\, v_1$ and $N[\theta, v_1/x] \downarrow^{(n_2,r_2)} v$. It suffices to show that

- $n_1 + n_2 \le k' \|M\|[\Theta]_c + \|N\|[\Theta, \|M\|[\Theta]_p/x]_c - (k'r_1 + r_2)$
- $v \sqsubseteq_{\texttt{val}}^{C,k'f[\sigma]+g[\sigma]+k'r_1+r_2} \|N\|[\Theta, \|M\|[\Theta]_p/x]_p$

By IH, we have that $M[\theta] \sqsubseteq^{!_l^k A, f[\sigma]} \|M\|[\Theta]$, which means that there are b_1, c_1 with $b_1 + c_1 = f[\sigma]$ and

- $n_1 \le \|M\|[\Theta]_c - r_1$
- $\texttt{save}_l^k\, v_1 \sqsubseteq_{\texttt{val}}^{!_l^k A, f[\sigma]+r_1} \|M\|[\Theta]_p$

Since $\texttt{save}_l^k\, v_1 \sqsubseteq_{\texttt{val}}^{!_l^k A, f[\sigma]+r_1} \|M\|[\Theta]_p$, there is a $d \ge 0$ such that $kd + l \le f[\sigma] + r_1$, and $v_1 \sqsubseteq_{\texttt{val}}^{A,d} \|M\|[\Theta]_p$. Thus, $(\theta, v_1/x) \sqsubseteq_{\texttt{sub}}^{(\Gamma,x:A),(\sigma,x\mapsto d)} (\Theta, \|M\|[\Theta]_p/x)$, and so by IH, $N[\theta, v_1/x] \sqsubseteq^{C,g[\sigma]+k'(kd+l)} \|N\|[\Theta, \|M\|[\Theta]_p/x]$. By credit weakening, since $kd + l \le f[\sigma] + r_1$, $N[\theta, v_1/x] \sqsubseteq^{C,g[\sigma]+k'(f[\sigma]+r_1)} \|N\|[\Theta, \|M\|[\Theta]_p/x]$. This gives us that

- $n_2 \le \|N\|[\Theta, \|M\|[\Theta]_p/x]_c - r_2$
- $v \sqsubseteq_{\texttt{val}}^{C,g[\sigma]+k'(f[\sigma]+r_1)+r_2} \|N\|[\Theta, \|M\|[\Theta]_p/x]_p$

To establish the desired inequality, we multiply the first inequality by k', to find that $k'n_1 \le k' \|M\|[\Theta]_c - k'r_1$. But $k' \ge 1$, so $n_1 \le k'n_1 \le k' \|M\|[\Theta]_c - k'r_1$. Therefore, $n_1 + n_2 \le k' \|M\|[\Theta]_c + \|N\|[\Theta, \|M\|[\Theta]_p/x]_c - (k'r_1 + r_2)$ as required. For value bounding, we note that $g[\sigma] + k'(f[\sigma] + r_1) + r_2 = k'f[\sigma] + g[\sigma] + k'r_1 + r_2$, and are done.

(spend) Let $\Gamma \vdash_{f+l} \texttt{spend}_l\, M : A$. By inversion, $\Gamma \vdash_f M : A$. To show $\texttt{spend}_l\, M \sqsubseteq^A (-l) +_c \|M\|$, suppose $\theta \sqsubseteq_{\texttt{sub}}^{\Gamma,\sigma} \Theta$. To show $\texttt{spend}_l\, M[\theta] \sqsubseteq^{A,f[\sigma]+l} (-l) +_c \|M\|[\Theta]$, suppose $\texttt{spend}_l\, M[\theta] \downarrow^{(n,r-l)} v$. By inversion we also have that $M[\theta] \downarrow^{(n,r)} v$. It suffices to show

- $n \le -l + \|M\|[\Theta]_c - (r - l)$
- $v \sqsubseteq_{\texttt{val}}^{A,f[\sigma]+l+r-l} \|M\|[\Theta]_p$

or, canceling, it suffices to show $n \le \|M\|[\Theta]_c - r$ and $v \sqsubseteq_{\texttt{val}}^{A,f[\sigma]+r} \|M\|[\Theta]_p$, which is precisely what we get from the IH.

(`create`) Let $\Gamma \vdash_f \texttt{create}_l\ M : A$. By inversion, $\Gamma \vdash_{f+l} M : A$. To show $\texttt{create}_l\ M \sqsubseteq^A l +_c \|M\|$, suppose $\theta \sqsubseteq^{\Gamma,\sigma}_{\texttt{sub}} \Theta$, to show $\texttt{create}_l\ M[\theta] \sqsubseteq^{A,f[\sigma]} l +_c \|M\|[\Theta]$, suppose $\texttt{create}_l\ M[\theta] \downarrow^{(n,r+l)} v$. By inversion, $M[\theta] \downarrow^{(n,r)} v$. It suffices to show

- $n \leq l + \|M\|[\theta]_c - (r+l)$
- $v \sqsubseteq^{f[\sigma]+r+l}_{\texttt{val}}$

By IH, we have that $M[\theta] \sqsubseteq^{A,f[\sigma]+l} \|M\|[\Theta]$, so

- $n \leq \|M\|[\Theta]_c - r$
- $v \sqsubseteq^{A,f[\sigma]+l+r}_{\texttt{val}}$

and so we are done, canceling the ls in the first inequality.

(tick) Immediate from IH, canceling 1s.

($\multimap$-I) Let $\Gamma \vdash_f \lambda x.M : A \multimap B$. By inversion, $\Gamma, x : A \vdash_{f+x} M : B$. Let $\theta \sqsubseteq^{\Gamma,\sigma}_{\texttt{sub}} \Theta$. To show $\lambda x.M[\theta] \sqsubseteq^{A\multimap B,f[\sigma]} (0, \lambda x. \|M\|[\Theta])$, let $\lambda x.M[\theta] \downarrow^{(0,0)} \lambda x.M[\theta]$. The first condition is trivial ($0 \leq 0$). We need to show that $\lambda x.M \sqsubseteq^{f[\sigma]}_{\texttt{val}} \lambda x. \|M\|[\Theta]$. Let $v \sqsubseteq^{A,d}_{\texttt{val}} E$. We must show that $M[\theta, v/x] \sqsubseteq^{B,f[\sigma]+d} (\lambda x. \|M\|[\Theta])E$, or by weakening, that $M[\theta, v/x] \sqsubseteq^{B,f[\sigma]+d} \|M\|[\Theta, E/x]$. But, since $v \sqsubseteq^{A,d}_{\texttt{val}} E$, we have $(\theta, v/x) \sqsubseteq^{(\Gamma,x:A),(\sigma,x\mapsto d)}_{\texttt{sub}} (\Theta, E/x)$, and so by IH, $M[\theta, v/x] \sqsubseteq^{B,f[\sigma]+d} \|M\|[\Theta, E/x]$, as required.

($\multimap$-E) Let $\Gamma \vdash_{f+g} M\,N : B$. Inversion gives $\Gamma \vdash_f M : A \multimap B$ and $\Gamma \vdash_g N : A$. Let $\theta \sqsubseteq^{\Gamma,\sigma}_{\texttt{sub}} \Theta$. We must show $M[\theta]\,N[\theta] \sqsubseteq^{B,f[\sigma]+g[\sigma]} (\|M\|[\Theta]_c + \|N\|[\Theta]_c) +_c \|M\|[\Theta]_p \|N\|[\Theta]_p$. Suppose $M[\theta]\,N[\theta] \downarrow^{(n_1+n_2+n_3, r_1+r_2+r_3)} v$. Inversion gives us that $M[\theta] \downarrow^{(n_1,r_1)} \lambda x.M'$, $N[\theta] \downarrow^{(n_2,r_2)} v_1$, and $M'[v_1/x] \downarrow^{(n_3,r_3)} v$. It remains to show that

- $n_1 + n_2 + n_3 \leq \|M\|[\Theta]_c + \|N\|[\Theta]_c + (\|M\|[\Theta]_p \|N\|[\Theta]_p)_c - (r_1 + r_2 + r_3)$
- $v \sqsubseteq^{B,f[\sigma]+g[\sigma]+r_1+r_2+r_3}_{\texttt{val}} (\|M\|[\Theta]_p \|N\|[\Theta]_p)_p$

By the IH applied to $\Gamma \vdash_f M : A \multimap B$, we know that $M[\theta] \sqsubseteq^{A\multimap B,f[\sigma]} \|M\|[\Theta]$, so

- $n_1 \leq \|M\|[\Theta]_c - r_1$
- $\lambda x.M' \sqsubseteq^{A\multimap B,f[\sigma]+r_1}_{\texttt{val}} \|M\|[\Theta]_p$.

Again applying the IH to $\Gamma \vdash_g N : A$, we know $N[\theta] \sqsubseteq^{A,g[\sigma]} \|N\|[\Theta]$, so

- $n_2 \leq \|N\|[\Theta]_c - r_2$
- $v_1 \sqsubseteq^{A,g[\sigma]+r_2}_{\texttt{val}} \|N\|[\Theta]_p$

But since $\lambda x.M' \sqsubseteq^{A\multimap B,f[\sigma]+r_1}_{\texttt{val}} \|M\|[\Theta]_p$ and $v_1 \sqsubseteq^{A,g[\sigma]+r_2}_{\texttt{val}} \|N\|[\Theta]_p$, we have $M'[v_1/x] \sqsubseteq^{B,f[\sigma]+g[\sigma]+r_1+r_2} \|M\|[\Theta]_p \|N\|[\Theta]_p$, which means that:

- $n_3 \leq (\|M\|[\Theta]_p \|N\|[\Theta]_p)_c - r_3$
- $v \sqsubseteq_{\texttt{val}}^{B,f[\sigma]+g[\sigma]+r_1+r_2+r_3} (\|M\|[\Theta]_p \|N\|[\Theta]_p)_p$

We add the inequalities together, and are done.

($\otimes$-I) Let $\Gamma \vdash_{f_1+g_1} (M_1, M_2) : A_1 \otimes A_2$. By inversion, we have that $\Gamma \vdash_{f_i} M_i : A_i$ for $i = 1, 2$. Let $\theta \sqsubseteq_{\texttt{sub}}^{\Gamma,\sigma} \Theta$. Towards proving $(M_1[\theta], M_2[\theta]) \sqsubseteq^{A_1 \otimes A_2, f_1[\sigma]+f_2[\sigma]} (\|M_1\|[\Theta]_c + \|M_2\|[\Theta]_c, (\|M_1\|[\Theta]_p, \|M_2\|[\Theta]_p))$, assume $(M_1[\theta], M_2[\theta]) \downarrow^{(n_1+n_2, r_1+r_2)} (v_1, v_2)$. By inversion, it must also be that $M_i[\theta] \downarrow^{(n_i, r_i)} v_i$ for $i = 1, 2$. If suffices to show:

- $n_1 + n_2 \leq \|M_1\|[\Theta]_c + \|M_2\|[\Theta] - (r_1 + r_2)$
- $(v_1, v_2) \sqsubseteq_{\texttt{val}}^{A_1 \otimes A_2, f_1[\sigma]+f_2[\sigma]+r_1+r_2} (\|M_1\|[\Theta]_p, \|M_2\|[\Theta]_p)$

By IH, we have that, for $i \in \{1, 2\}$

- $n_i \leq \|M_i\|[\Theta]_c - r_i$
- $v_i \sqsubseteq_{\texttt{val}}^{A_i, f[\sigma]_i+r_i} \|M_i\|[\Theta]_p$

Adding the two inequalities and applying the definition of value bounding at $\otimes$, we are done.

($\otimes$-E) Let $\Gamma \vdash_{k'f+g} \texttt{split}_{k'}(M, x.y.N) : C$. Inversion gives $\Gamma \vdash_f M : A \otimes B$, and $\Gamma, x : A, y : B \vdash_{g+k'(x+y)} N : C$. Let $\theta \sqsubseteq_{\texttt{sub}}^{\Gamma,\sigma} \Theta$. We must show that

$$\texttt{split}_{k'}(M[\theta], x.y.N[\theta]) \sqsubseteq^{C,k'f[\sigma]+g[\sigma]} k' \|M\|[\Theta]_c +_c \|N\|[\Theta, \pi_1 \|M\|[\Theta]_p/x, \pi_2 \|M\|[\Theta]_p/y]$$

Suppose that $\texttt{split}_{k'}(M[\theta], x.y.N[\theta]) \downarrow^{(n_1+n_2, k'r_1+r_2)} v$ by way of $M[\theta] \downarrow^{(n_1,r_1)} (v_1, v_2)$ and $N[\theta, v_1/x, v_2/y] \downarrow^{(n_2,r_2)} v$. It remains to show that

- $n_1 + n_2 \leq k' \|M\|[\Theta]_c + \|N\|[\Theta, \pi_1 \|M\|[\Theta]_p/x, \pi_2 \|M\|[\Theta]_p/y]_c - (k'r_1 + r_2)$
- $v \sqsubseteq_{\texttt{val}}^{C,k'f[\sigma]+g[\sigma]+k'r_1+r_2} \|N\|[\Theta, \pi_1 \|M\|[\Theta]_p/x, \pi_2 \|M\|[\Theta]_p/y]_p$

By IH, $M[\theta] \sqsubseteq^{A \otimes B, f[\sigma]} \|M\|[\Theta]$, so

- $n_1 \leq \|M\|[\Theta]_c - r_1$
- $(v_1, v_2) \sqsubseteq_{\texttt{val}}^{A \otimes B, f[\sigma]+r_1} \|M\|[\Theta]_p$

and so there are $c_1, c_2 \geq 0$ so that $c_1 + c_2 = f[\sigma] + r_1$ and $v_1 \sqsubseteq_{\texttt{val}}^{A,c_1} \pi_1 \|M\|[\Theta]_p$ and $v_2 \sqsubseteq_{\texttt{val}}^{B,c_2} \pi_2 \|M\|[\Theta]_p$. So, $(\theta, v_1/x, v_2/y) \sqsubseteq_{\texttt{sub}}^{(\Gamma,x:A,y:B),(\sigma,x\mapsto c_1,y\mapsto c_2)} (\Theta, \pi_1 \|M\|[\Theta]_p/x, \pi_2 \|M\|[\Theta]_p/y)$. So, by IH, $N[\theta, v_1/x, v_2/y] \sqsubseteq^{C,g[\sigma]+k'(f[\sigma]+r_1)} \|N\|[\Theta, \pi_1 \|M\|[\Theta]_p/x, \pi_2 \|M\|[\Theta]_p/y]$, so

- $n_2 \leq \|N\|[\Theta, \pi_1 \|M\|[\Theta]_p/x, \pi_2 \|M\|[\Theta]_p/y]_c - r_2$
- $v \sqsubseteq_{\texttt{val}}^{C,k'f[\sigma]+g[\sigma]+k'r_1+r_2} \|N\|[\Theta, \pi_1 \|M\|[\Theta]_p/x, \pi_2 \|M\|[\Theta]_p/y]_p$

Then,

$$n_1 + n_2 \le k' n_1 + n_2$$
$$\le k' \left\| M \right\| [\Theta]_c + \left\| N \right\| [\Theta, \pi_1 \left\| M \right\| [\Theta]_p / x, \pi_2 \left\| M \right\| [\Theta]_p / y]_c - (k' r_1 + r_2)$$

as required.

(⊕-E) Let $\Gamma \vdash_{k'f+g_1+g_2} \texttt{case}_{k'} (M, x.N_1 , y.N_2) : C$. By inversion, $\Gamma \vdash_f M : A \oplus B$, $\Gamma, x : A \vdash_{g_1+k'x} N_1 : C$, and $\Gamma, y : B \vdash_{g_2+k'y} N_2 : C$. Let $\theta \sqsubseteq_{\texttt{sub}}^{\Gamma,\sigma} \Theta$. We must show that

$$\texttt{case}_{k'} (M[\theta], x.N_1[\theta] , y.N_2[\theta]) \sqsubseteq^{C,k' f[\sigma]+g_1[\sigma]+g_2[\sigma]} k' \left\| M \right\| [\Theta]_c +_c \texttt{case} (\left\| M \right\| [\Theta]_p, x. \left\| N_1 \right\| , y. \left\| N_2 \right\|)$$

Because the two cases are symmetric, we consider only the following evaluation: $\texttt{case}_{k'} (M[\theta], x.N_1[\theta] , y.N_2[\theta]) \downarrow^{(n_1+n_2, k' r_1 + r_2)} v$ by way of $M[\theta] \downarrow^{(n_1,r_1)} \texttt{inl}\, v_1$ and $N_1[\theta, v_1/x] \downarrow^{(n_2,r_2)} v$. We must show that

- $n_1 + n_2 \le k' \left\| M \right\| [\Theta]_c + \texttt{case} (\left\| M \right\| [\Theta]_p, x. \left\| N_1 \right\| , y. \left\| N_2 \right\|)_c - (k' r_1 + r_2)$
- $v \sqsubseteq_{\texttt{val}}^{C,k' f[\sigma]+g_1[\sigma]+g_2[\sigma]+k' r_1+r_2} \texttt{case} (\left\| M \right\| [\Theta]_p, x. \left\| N_1 \right\| , y. \left\| N_2 \right\|)_p$

By IH, $M[\theta] \sqsubseteq^{A\oplus B, f[\sigma]} \left\| M \right\| [\Theta]$, so

- $n_1 \le \left\| M \right\| [\Theta]_c - r_1$
- $\texttt{inl}\, v_1 \sqsubseteq_{\texttt{val}}^{A\oplus B, f[\sigma]+r_1} \left\| M \right\| [\Theta]_p$

so there is an E such that $\texttt{inl}\, E \le_{\langle\!\langle A \rangle\!\rangle + \langle\!\langle B \rangle\!\rangle} \left\| M \right\| [\Theta]_p$ and $v_1 \sqsubseteq_{\texttt{val}}^{A, f[\sigma]+r_1} E$. So, $(\theta, v_1/x) \sqsubseteq_{\texttt{sub}}^{(\Gamma, x:A),(\sigma, x\mapsto f[\sigma]+r_1)} (\Theta, E/x)$ and hence by IH, $N_1[\theta, v_1/x] \sqsubseteq^{g_1[\sigma]+k'(f[\sigma]+r_!)} \left\| N_1 \right\| [\Theta, E/x]$, so

- $n_2 \le \left\| N_1 \right\| [\Theta, E/x]_c - r_2$
- $v \sqsubseteq_{\texttt{val}}^{C,k' f[\sigma]+g_1[\sigma]+k' r_1+r_2} \left\| N_1 \right\| [\Theta, E/x]_p$

Since $g_2[\sigma] \ge 0$, we have by credit weakening that $v \sqsubseteq_{\texttt{val}}^{C,k' f[\sigma]+g_1[\sigma]+g_2[\sigma]+k' r_1+r_2} \left\| N_1 \right\| [\Theta, E/x]_p$. Then, we compute:

$$\left\| N_1 \right\| [\Theta, E/x] \le_{\left\| C \right\|} \texttt{CASE} (\texttt{INL}\, E, x. \left\| N_1 \right\| [\Theta] , y. \left\| N_2 \right\| [\Theta])$$
$$\le \texttt{case} (\left\| M \right\| [\Theta]_p, x. \left\| N_1 \right\| [\Theta] , y. \left\| N_2 \right\| [\Theta])$$

which gives us the value bounding condition, and again compute:

$$n_1 + n_2 \le k' n_1 + n_2$$
$$\le k' \left\| M \right\| [\Theta]_c + \texttt{case} (\left\| M \right\| [\Theta]_p, x. \left\| N_1 \right\| , y. \left\| N_2 \right\|)_c - (k' r_1 + r_2)$$

which gives us the cost bounding condition.

($\oplus$-I) The cases for $\mathtt{inl}\,M$ and $\mathtt{inr}\,M$ are symmetric, so we let $\Gamma \vdash_f \mathtt{inl}\,M : A \oplus B$. Inversion gives $\Gamma \vdash_f M : A$. Let $\theta \sqsubseteq^{\Gamma,\sigma}_{\mathtt{sub}} \Theta$. To show that $\mathtt{inl}\,M[\theta] \sqsubseteq^{A\oplus B, f[\sigma]} (\|M\|[\Theta]_c, \mathtt{inl}\,\|M\|[\Theta]_p)$, we let $\mathtt{inl}\,M[\theta] \downarrow^{(n,r)} \mathtt{inl}\,v$. Inverting, we have $M[\theta] \downarrow^{(n,r)} v$. It suffices to show that $n \le \|M\|[\Theta]_c - r$, and that $\mathtt{inl}\,v \sqsubseteq^{A\oplus B, f[\sigma]+r}_{\mathtt{val}} \mathtt{inl}\,\|M\|[\Theta]_p$. By IH we have $n \le \|M\|[\Theta]_c - r$, and $v \sqsubseteq^{A, f[\sigma]+r}_{\mathtt{val}} \|M\|[\Theta]_p$. So we are done by the definition of value bounding at $\oplus$ for $\mathtt{inl}$.

($\mathtt{[}A\mathtt{]}$-I, cons) Let $\Gamma \vdash_{f+g} M :: N : \mathtt{[}A\mathtt{]}$. By inversion, $\Gamma \vdash_f M : A$ and $\Gamma \vdash_g N : \mathtt{[}A\mathtt{]}$. Let $\theta \sqsubseteq^{\Gamma,\sigma}_{\mathtt{sub}} \Theta$. To show that $M :: N \sqsubseteq^{\mathtt{[}A\mathtt{]}, f[\sigma]+g[\sigma]} (\|M\|[\Theta]_c + \|N\|[\Theta]_c, \|M\|[\Theta]_p :: \|N\|[\Theta]_p)$, let $M :: N \downarrow^{(n_1+n_2, r_1+r_2)} v_1 :: v_2$. By inversion, $M \downarrow^{(n_1,r_1)} v_1$ and $N \downarrow^{(n_2,r_2)} v_2$. It suffices to provide b, c where $c \ge 0$ and $b + c = f[\sigma] + g[\sigma]$ and that

– $n_1 + n_2 \le \|M\|[\Theta]_c + \|N\|[\Theta]_c - (r_1 + r_2)$

– $v_1 :: v_2 \sqsubseteq^{\mathtt{[}A\mathtt{]}, f[\sigma]+g[\sigma]+r_1+r_2}_{\mathtt{val}} \|M\|[\Theta]_p :: \|N\|[\Theta]_p$.

By IH,

– $n_1 \le \|M\|[\Theta]_c - r_1$

– $v_1 \sqsubseteq^{A, f[\sigma]+r_1}_{\mathtt{val}} \|M\|[\Theta]_p$

and by IH on N, there are b_2, c_2 with $c_2 \ge 0$ and $b_2 + c_2 = g[\sigma]$ such that

– $n_2 \le \|N\|[\Theta]_c - r_2$

– $v_2 \sqsubseteq^{\mathtt{[}A\mathtt{]}, g[\sigma]+r_2}_{\mathtt{val}} \|N\|[\Theta]_p$

Thus, the desired inequality follows from adding the two inductively computed ones, and the value bounding relation for $v_1 :: v_2$ is immediate by the definition.

($\mathtt{[}A\mathtt{]}$-E) Suppose $\Gamma \vdash_{f+g_1+g_2} \mathtt{lrec}\,(M, N_1, N_2) : C$. By inversion, $\Gamma \vdash_f M : \mathtt{[}A\mathtt{]}$, $\Gamma \vdash_{g_1} N_1 : 1 \multimap C$, $\Gamma \vdash_{g_2} N_2 :!^{\infty}_0 (A \otimes (\mathtt{[}A\mathtt{]} \& C) \multimap C)$. Let $\theta \sqsubseteq^{\Gamma,\sigma}_{\mathtt{sub}} \Theta$. To show that

$$\mathtt{lrec}\,(M[\theta], N_1[\theta], N_2[\theta]) \sqsubseteq^{C, f[\sigma]+g_1[\sigma]+g_2[\sigma]} (\|M\|[\Theta]_c + \|N_1\|[\Theta]_c + \|N_2\|[\Theta]_c) +_c$$
$$\mathtt{lrec}\,(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \lambda(a,(as,r)).\,\|N_2\|[\Theta]_p\,(a,((0,as),r)))$$

we break into the two evaluation cases. Firstly, suppose that $\mathtt{lrec}\,(M[\theta], N_1[\theta], N_2[\theta]) \downarrow^{(n_1+n_2+n_3+n_4, r_1+r_2+r_3+r_4)} v$ by way of $M[\theta] \downarrow^{(n_1,r_1)} \mathtt{[]}$, $N_1[\theta] \downarrow^{(n_2,r_2)} \lambda x.N_1'$, $N_2[\theta] \downarrow^{(n_3,r_3)} \mathtt{save}^{\infty}_0\ (\lambda x.N_2')$, and $N_1'[()/x] \downarrow^{(n_4,r_4)} v$. From here, denote we denote $\lambda x.\,\|N_2\|[\Theta]_p(\pi_1 x, ((0, \pi_1\pi_2 x), \pi_2\pi_2 x))$ as $\|N_2\|^*$.

It suffices to show that:

- $n_1 + n_2 + n_3 + n_4 \leq \|M\|[\Theta]_c + \|N_1\|[\Theta]_c + \|N_2\|[\Theta]_c + \texttt{lrec}\left(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|^*\right)_c - (r_1 + r_2 + r_3 + r_4)$
- $v \sqsubseteq_{\texttt{val}}^{C, f[\sigma]+g_1[\sigma]+g_2[\sigma]+r_1+r_2+r_3+r_4} \texttt{lrec}\left(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|^*\right)_p$

By IH, $M[\theta] \sqsubseteq^{[A], f[\sigma]} \|M\|[\Theta]$, so

- $n_1 \leq \|M\|[\Theta]_c - r_1$
- $\texttt{[]} \sqsubseteq_{\texttt{val}}^{f[\sigma]+r_1} \|M\|[\Theta]_p$

The second condition tells us that $\texttt{[]} \leq \|M\|[\Theta]_p$. Again by IH, $N_2[\theta] \sqsubseteq^{!_0^\infty(A\otimes(\texttt{[}A\texttt{]}\&C)\multimap C), g_2[\sigma]} \|N_2\|[\Theta]$, so

- $n_3 \leq \|N_2\|[\Theta]_p - r_3$
- $\texttt{save}_0^\infty\ (\lambda x.N_2') \sqsubseteq_{\texttt{val}}^{!_0^\infty(A\otimes(\texttt{[}A\texttt{]}\&C)\multimap C), g_2[\sigma]+r_3} \|N_2\|[\Theta]_p$

In particular, by preservation, $g_2[\sigma]+r_3 \geq 0$. Thirdly by IH, $N_1[\theta] \sqsubseteq^{1\multimap C, g_1[\sigma]} \|N_1\|[\Theta]$, which means

- $n_2 \leq \|N_1\|[\Theta]_c - r_2$
- $\lambda x.N_1' \sqsubseteq_{\texttt{val}}^{1\multimap C, g_1[\sigma]+r_2} \|N_1\|[\Theta]_p$

Since $() \leq ()$, $() \sqsubseteq_{\texttt{val}}^{1, f[\sigma]+r_1} ()$. Hence, $N_2'[()/x] \sqsubseteq^{C, g_1[\sigma]+f[\sigma]+r_1+r_2} \|N_1\|[\Theta]_p\ ()$. This means that

- $n_4 \leq (\|N_1\|[\Theta]_p\ ())_c - r_4$
- $v \sqsubseteq_{\texttt{val}}^{C, f[\sigma]+g_1[\sigma]+r_1+r_2+r_4} (\|N_1\|[\Theta]_p\ ())_p$

But by credit weakening, since $g_2[\sigma] + r_3 \geq 0$, we have $v \sqsubseteq_{\texttt{val}}^{C, f[\sigma]+g_1[\sigma]+g_2[\sigma]+r_1+r_2+r_3+r_4}$ $(\|N_1\|[\Theta]_p\ ())_p$. But, we can compute:

$$\|N_1\|[\Theta]_p\ () \leq \texttt{lrec}\,(\texttt{[]}, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p) \leq \texttt{lrec}\left(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|^*\right)$$

and we are done by weakening.

Otherwise, assume $M[\theta] \downarrow^{(n_1,r_1)} v_1 :: v_2$, $N_2[\theta] \downarrow^{(n_2,r_2)} \texttt{save}_0^\infty\ (\lambda x.N_2')$, $N_1[\theta] \downarrow^{(n_3,r_3)} \lambda x.N_1'$, and

$$N_2'[(v_1, \langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2'))\rangle)] \downarrow^{(n_4,r_4)} v$$

. Just like the previous case, it suffices to show

- $n_1 + n_2 + n_3 + n_4 \leq \|M\|[\Theta]_c + \|N_1\|[\Theta]_c + \|N_2\|[\Theta]_c + \texttt{lrec}\left(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|^*\right)_c - (r_1 + r_2 + r_3 + r_4)$

- $v \sqsubseteq_{\texttt{val}}^{C, f[\sigma]+g_1[\sigma]+g_2[\sigma]+r_1+r_2+r_3+r_4} \texttt{lrec}\,(\|M\|\,[\Theta]_p, \|N_1\|\,[\Theta]_p, \|N_2\|\,[\Theta]_p)_p$

By IH, $M[\theta] \sqsubseteq^{\texttt{[}A\texttt{]}, f[\sigma]} \|M\|\,[\Theta]$, so

- $n_1 \leq \|M\|\,[\Theta]_c - r_1$
- $v_1 :: v_2 \sqsubseteq_{\texttt{val}}^{\texttt{[}A\texttt{]}, f[\sigma]+r_1} \|M\|\,[\Theta]_p$

By the second condition, we know that there are $d_1, d_2 \geq 0$ with $d_1 + d_2 = f[\sigma] + r_1$, and E_1, E_2 with $E_1 :: E_2 \leq \|M\|\,[\Theta]_p$ such that $v_1 \sqsubseteq_{\texttt{val}}^{A,d_1} E_1$, and $v_2 \sqsubseteq_{\texttt{val}}^{\texttt{[}A\texttt{]},d_2} E_2$. By IH, $N_1[\theta] \sqsubseteq^{1 \multimap C, g_1[\sigma]} \|N_1\|\,[\Theta]$, so

- $n_3 \leq \|N_1\|\,[\Theta]_c - r_3$
- $\lambda x.N_1' \sqsubseteq_{\texttt{val}}^{1\multimap C, g_1[\sigma]+r_3} \|N_1\|\,[\Theta]_p$

Again by IH, $N_2[\theta] \sqsubseteq^{!_0^\infty (A \otimes (\texttt{[}A\texttt{]} \& C) \multimap C), g_2[\sigma]} \|N_2\|\,[\Theta]$, which means that

- $n_2 \leq \|N_2\|\,[\theta]_c - r_2$
- $\texttt{save}_0^\infty\ (\lambda x.N_2') \sqsubseteq_{\texttt{val}}^{!^\infty 0 (A \otimes (\texttt{[}A\texttt{]} \& C) \multimap C), g_2[\sigma]+r_2} \|N_2\|\,[\Theta]_p$

The second condition means by definition that there is a $c \geq 0$ such that $\infty \cdot c \leq g_2[\sigma] + r_2$, and $\lambda x.N_2' \sqsubseteq_{\texttt{val}}^{A \otimes (\texttt{[}A\texttt{]} \& C) \multimap C, c} \|N_2\|\,[\Theta]_p$. We claim that

$$N_2'[(v_1, \langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_2', \texttt{save}_0^\infty\ (\lambda x.N_2'))\rangle)] \sqsubseteq^{f[\sigma]+g_1[\sigma]+g_2[\sigma]+r_1+r_2+r_3}$$
$$\|N_2\|\ (E_1, ((0, E_2), \texttt{lrec}\big(E_2, \|N_1\|\,[\Theta]_p, \|N_2\|^*\big)))$$

To prove this claim, we split into cases on the finitude of $g_2[\sigma] + r_2$.

- Suppose $g_2[\sigma] + r_2$ is finite. Then $c = 0$, and $g_2[\sigma] + r_2 = 0$, and so by the list recursor lemma with $E_1 = \|N_1\|\,[\Theta]_p$, $E_2 = \|N_2\|\,[\Theta]_p$, and $E = E_2$, we have that $\texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2')) \sqsubseteq^{C, d_2+g_1[\sigma]+r_3} \texttt{lrec}\big(E_2, \|N_1\|\,[\Theta]_p, \|N_2\|^*\big)$. Since $v_2 \sqsubseteq_{\texttt{val}}^{\texttt{[}A\texttt{]},d_2} E_2$, we have that $v_2 \sqsubseteq^{\texttt{[}A\texttt{]},d_2} (0, E_2)$, and by credit weakening, since $g_1[\sigma] + r_3 \geq 0$, $v_2 \sqsubseteq^{\texttt{[}A\texttt{]}, d_2+g_1[\sigma]+r_3} (0, E_2)$. Thus:

  $$\langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2'))\rangle \sqsubseteq^{\texttt{[}A\texttt{]}\& C, d_2+g_1[\sigma]+r_3}$$
  $$((0, E_2), \texttt{lrec}\big(E_2, \|N_1\|\,[\Theta]_p, \|N_2\|^*\big))$$

  Next, since $v_1 \sqsubseteq_{\texttt{val}}^{A,d_1} E_1$, $d_1 + d_2 = f[\sigma] + r_1$, and $\lambda x.N_2' \sqsubseteq_{\texttt{val}}^{A \otimes (\texttt{[}A\texttt{]} \& C), 0} \|N_2\|\,[\Theta]_p$,

  $$N_2'[(v_1, \langle v_2, \texttt{lrec}\,(v_2, \lambda x.N_1', \texttt{save}_0^\infty\ (\lambda x.N_2'))\rangle)/x] \sqsubseteq^{f[\sigma]+g_1[\sigma]+r_1+r_3}$$
  $$\|N_2\|\,[\Theta]_p\ (E_1, ((0, E_2), \texttt{lrec}\big(E_2, \|N_1\|\,[\Theta]_p, \|N_2\|^*\big)))$$

  Which is exactly what we wanted to show, since $g_2[\sigma] + r_2 = 0$.

– Suppose $g_2[\sigma] + r_2 = \infty$. Then, there is a $c \geq 0$ such that $\infty \cdot c \leq \infty$, and $\lambda x.N_2' \sqsubseteq^{A\otimes([A]\&C)\multimap C,c} \|N_2\|[\Theta]_p$ By credit weakening, we may assume $c > 0$. By the list recursor lemma, $\texttt{lrec}(v_2, \lambda x.N_1', \texttt{save}_0^\infty(\lambda x.N_2')) \sqsubseteq^{C,d_2+g_1[\sigma]+r_3+\infty\cdot c}$ $\texttt{lrec}(E_2, \|N_1\|[\Theta]_p, \|N_2\|^*)$. By the same reasoning as in the previous case,

$(v_1, \langle v_2, \texttt{lrec}(v_2, \lambda x.N_1', \texttt{save}_0^\infty(\lambda x.N_2'))\rangle) \sqsubseteq^{A\otimes([A]\&C),f[\sigma]+g_1[\sigma]+r_1+r_3}$

$(E_1, ((0,E_2), \texttt{lrec}(E_2, \|N_1\|[\Theta]_p, \|N_2\|^*)))$

Then, since $\infty \cdot c + c = \infty \cdot c$,

$N_2'[(v_1, \langle v_2, \texttt{lrec}(v_2, \lambda x.N_1', \texttt{save}_0^\infty(\lambda x.N_2'))\rangle)/x] \sqsubseteq^{C,f[\sigma]+g_1[\sigma]+r_1+r_3+\infty\cdot c}$

$\|N_2\|[\Theta]_p\,(E_1, ((0,E_2), \texttt{lrec}(E_2, \|N_1\|[\Theta]_p, \|N_2\|^*)))$

which, $\infty \cdot c \leq g_2[\sigma] + r_2$, gives us our goal by credit weakening.

From this result, we have by definition that

– $n_4 \leq (\|N_2\|[\Theta]_p\,(E_1, ((0,E_2), \texttt{lrec}(E_2, \|N_1\|[\Theta]_p, \|N_2\|^*))))_c - r_4$

– $v \sqsubseteq_{\texttt{val}}^{C,f[\sigma]+g_1[\sigma]+g_2[\sigma]+r_1+r_2+r_3+r_4} (\|N_2\|[\Theta]_p\,(E_1, ((0,E_2), \texttt{lrec}(E_2, \|N_1\|[\Theta]_p, \|N_2\|^*))))_p$

Then we can compute:

$$\begin{aligned}\|N_2\|[\Theta]_p\,(E_1, ((0,E_2), \texttt{lrec}(E_2, \|N_1\|[\Theta]_p, \|N_2\|^*))) &\leq \|N_2\|^*\ (E_1, (E_2, \texttt{lrec}(E_2, \|N_1\|[\Theta]_p, \|N_2\|^*)))\\ &\leq \texttt{lrec}(E_1 :: E_2, \|N_1\|[\Theta]_p, \|N_2\|^*)\\ &\leq \texttt{lrec}(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|^*)\end{aligned}$$

and so we are done by weakening.

(ℕ-E) Suppose $\Gamma \vdash_{f+g_1+g_2} \texttt{nrec}(M, N_1, N_2) : C$. By inversion, we have that $\Gamma \vdash_f M : \mathbb{N}$, $\Gamma \vdash_{g_1} N_1 : 1 \multimap C$, and $\Gamma \vdash_{g_2} N_2 :!_0^\infty(\mathbb{N} \otimes (1 \multimap C) \multimap C)$. Let $\theta \sqsubseteq_{\texttt{sub}}^{\Gamma,\sigma} \Theta$. For convenience, let $\|N_2\|[\Theta]_p^* = \lambda p.\,\|N_2\|[\Theta]_p(\pi_1 p, \lambda z.\pi_2 p)$. We must show:

$\texttt{nrec}(M[\theta], N_1[\theta], N_2[\theta]) \sqsubseteq^{C,f[\sigma]+g_1[\sigma]+g_2[\sigma]}$

$(\|M\|[\Theta]_c + \|N_1\|[\Theta]_c + \|N_2\|[\Theta]_c) +_c \texttt{nrec}(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p^*)$

In order to show this, we have two evaluation cases to consider. Suppose

$$\texttt{nrec}(M[\theta], N_1[\theta], N_2[\theta]) \downarrow^{(n_1+n_2+n_3+n_4, r_1+r_2+r_3+r_4)} v$$

by way of $M[\theta] \downarrow^{(n_1,r_1)} 0$, $N_1[\theta] \downarrow^{(n_2,r_2)} \lambda x.N_1'$, $N_2[\theta] \downarrow^{(n_3,r_3)} v'$, and $N_1'[()/x] \downarrow^{(n_4,r_4)}$ v. It suffices to show that:

– $n_1 + n_2 + n_3 + n_4 \leq b + \|M\|[\Theta]_c + \|N_1\|[\Theta]_c + \|N_2\|[\Theta]_c + \texttt{nrec}(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p^*)_c - (r_1 + r_2 + r_3 + r_4)$

- $v \sqsubseteq_{\texttt{val}}^{C,f[\sigma]+g_1[\sigma]+g_2[\sigma]+r_1+r_2+r_3+r_4} \texttt{nrec}\left(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p^*\right)_p$

By IH, $M[\theta] \sqsubseteq^{\mathbb{N},f[\sigma]} \|M\|[\Theta]$, so

- $n_1 \leq \|M\|[\Theta]_c - r_1$
- $0 \sqsubseteq_{\texttt{val}}^{\mathbb{N},f[\sigma]+r_1} \|M\|[\Theta]_p$

since $0 \sqsubseteq_{\texttt{val}}^{\mathbb{N},f[\sigma]+r_1} \|M\|[\Theta]_p$, $0 \leq_{\mathbb{N}} \|M\|[\Theta]_p$. By IH, $N_1[\theta] \sqsubseteq^{1\multimap C,g_1[\sigma]} \|N_1\|[\Theta]$, so

- $n_2 \leq \|N_1\|[\Theta]_c - r_2$
- $\lambda x.N_1' \sqsubseteq_{\texttt{val}}^{1\multimap C,g_1[\sigma]+r_2} \|N_1\|[\Theta]_p$

By IH, $N_2[\theta] \sqsubseteq^{!_0^\infty(\cdots),g_2[\sigma]} \|N_2\|[\Theta]$, and so

- $n_3 \leq \|N_2\|[\Theta]_c - r_3$

We omit the value bounding condition since it does not factor into the rest of the proof. Since $() \leq_1 ()$, $() \sqsubseteq_{\texttt{val}}^{1,f[\sigma]+r_1} ()$. So: $N_1'[()/x] \sqsubseteq^{C,f[\sigma]+g_1[\sigma]+r_1+r_2} \|N_1\|[\Theta]\,()$. Thus,

- $n_4 \leq (\|N_1\|[\Theta]\,())_c - r_4$
- $v \sqsubseteq_{\texttt{val}}^{C,f[\sigma]+g_1[\sigma]+r_1+r_2+r_4} (\|N_1\|[\Theta]\,())_p$

Since $g_2[\sigma] + r_3 \geq 0$, we know by credit weakening, $v \sqsubseteq_{\texttt{val}}^{C,f[\sigma]+g_1[\sigma]+g_2[\sigma]+r_1+r_2+r_3+r_4} (\|N_1\|[\Theta]\,())_p$. Since $0 \leq \|M\|[\Theta]_p$, we compute:

$$\begin{aligned}\|N_1\|[\Theta]\,() &\leq_C \texttt{nrec}\left(0, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p^*\right)\\ &\leq \texttt{nrec}\left(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p^*\right)\end{aligned}$$

So we are done by weakening.

Suppose $\texttt{nrec}\,(M[\theta], N_1[\theta], N_2[\theta]) \downarrow^{(n_1+n_2+n_3+n_4, r_1+r_2+r_3+r_4)} v$ by way of $M[\theta] \downarrow^{(n_1,r_1)} S(v_1)$, $N_2[\theta] \downarrow^{(n_2,r_2)} \texttt{save}_0^\infty\, \lambda x.N_2'$, $N_1[\theta] \downarrow^{(n_3,r_3)} \lambda x.N_1'$, and

$$N_2'[(v_1, \texttt{nrec}\,(v_1, \lambda z.(\texttt{nrec}\,(v_1, \lambda x.N_1', \texttt{save}_0^\infty\, \lambda x.N_2')),))/x] \downarrow^{(n_4,r_4)} v$$

- $n_1 + n_2 + n_3 + n_4 \leq b + \|M\|[\Theta]_c + \|N_1\|[\Theta]_c + \|N_2\|[\Theta] + \texttt{nrec}\left(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p^*\right)_c - (r_1 + r_2 + r_3 + r_4)$
- $v \sqsubseteq_{\texttt{val}}^{C,f[\sigma]+g_1[\sigma]+g_2[\sigma]+r_1+r_2+r_3+r_4} \texttt{nrec}\left(\|M\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p^*\right)_p$

By IH, $M[\theta] \sqsubseteq^{\mathbb{N},f[\sigma]} \|M\|[\Theta]$, so

- $n_1 \leq \|M\|[\Theta]_c - r_1$
- $S(v_1) \sqsubseteq_{\texttt{val}}^{\mathbb{N},f[\sigma]+r_1} \|M\|[\Theta]_p$

Since $S(v_1) \sqsubseteq_{\mathtt{val}}^{\mathbb{N},f[\sigma]+r_1} \|M\|[\Theta]_p$, there is an E such that $v_1 \sqsubseteq_{\mathtt{val}}^{\mathbb{N},f[\sigma]+r_1} E$, and $S(E) \le \|M\|[\Theta]_p$.

By IH, $N_1[\theta] \sqsubseteq^{1\multimap C,g_1[\sigma]} \|N_1\|[\Theta]$, and so

- $n_3 \le \|N_1\|[\Theta]_c - r_3$
- $\lambda x.N_1' \sqsubseteq_{\mathtt{val}}^{1\multimap C,g_1[\sigma]+r_3} \|N_1[\Theta]\|_p$

By IH, $N_2[\theta] \sqsubseteq^{!_0^\infty(\mathbb{N}\otimes(1\multimap C)\multimap C),g_2[\sigma]} \|N_2\|[\Theta]$, so by definition,

- $n_2 \le \|N_2\|[\Theta]_c - r_2$
- $\mathtt{save}_0^\infty\ \lambda x.N_2' \sqsubseteq_{\mathtt{val}}^{!_0^\infty(\mathbb{N}\otimes(1\multimap C)\multimap C),g_2[\sigma]+r_2} \|N_2\|[\Theta]_p$

and so there is a $d \ge 0$ so that $\lambda x.N_2' \sqsubseteq_{\mathtt{val}}^{\mathbb{N}\otimes(1\multimap C)\multimap C,d} \|N_2\|[\Theta]_p$, and $\infty \cdot d \le g_2[\sigma] + r_3$. By the $\mathbb{N}$-recursor lemma, $\mathtt{nrec}\,(v_1, \lambda x.N_1', \mathtt{save}_0^\infty\ (\lambda x.N_2')) \sqsubseteq^{C,g_1[\sigma]+r_2+\infty\cdot d}$ $\mathtt{nrec}\big(E, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p^*\big)$. Let $E^* = (E, \mathtt{nrec}\big(E, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p^*\big))$. Note that $v_1 \sqsubseteq_{\mathtt{val}}^{\mathbb{N},f[\sigma]+r_1} \pi_1 E^*$ and

$$\lambda z.\mathtt{nrec}\,(v_1, \lambda x.N_1', \mathtt{save}_0^\infty\ (\lambda x.N_2')) \sqsubseteq_{\mathtt{val}}^{1\multimap C,g_1[\sigma]+r_2+\infty\cdot d} \lambda z.\pi_2 E^*$$

, and so:

$$(v_1, \lambda z.\mathtt{nrec}\,(v_1, \lambda x.N_1', \mathtt{save}_0^\infty\ (\lambda x.N_2'))) \sqsubseteq_{\mathtt{val}}^{\mathbb{N}\otimes(1\multimap C),f[\sigma]+g_1[\sigma]+\infty\cdot d+r_1+r_2} (\pi_1 E^*, \lambda z.\pi_2 E^*)$$

.

Thus, because $\lambda z.N_2' \sqsubseteq_{\mathtt{val}}^{\mathbb{N}\otimes(1\multimap C)\multimap C,d} \|N_2\|[\Theta]_p$, and $\infty \cdot d + d = \infty \cdot d$

$$N_2'[(v_1, \lambda z.\mathtt{nrec}\,(v_1, \lambda x.N_1', \mathtt{save}_0^\infty\ (\lambda x.N_2')))] \sqsubseteq^{C,f[\sigma]+g_1[\sigma]+\infty\cdot d+r_1+r_2} \|N_2\|[\Theta]_p(\pi_1 E^*, \lambda z.\pi_2 E^*)$$

and so:

- $n_4 \le (\|N_2\|[\Theta]_p\ (\pi_1 E^*, \lambda z.\pi_2 E^*))_c - r_4$
- $v \sqsubseteq_{\mathtt{val}}^{C,f[\sigma]+g_1[\sigma]+\infty\cdot d+r_1+r_2+r_4} (\|N_2\|[\Theta]_p\ (\pi_1 E^*, \lambda z.\pi_2 E^*))_p$

but, $\infty \cdot d \le g_2[\sigma] + r_3$, and

$$\begin{aligned}
\|N_2\|[\Theta]_p\ (\pi_1 E^*, \lambda z.\pi_2 E^*) &\le (\lambda p.\,\|N_2\|[\Theta]_p(\pi_1 p, \lambda z.\pi_2 p))E^* \\
&\le \|N_2\|[\Theta]_p^*\ (E, \mathtt{nrec}\big(E, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p^*\big)) \\
&\le \mathtt{nrec}\,(S(E), \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p) \\
&\le \mathtt{nrec}\,(\|N_1\|[\Theta]_p, \|N_1\|[\Theta]_p, \|N_2\|[\Theta]_p)
\end{aligned}$$

and so we are done by weakening and credit weakening.

(&-I) Suppose $\Gamma \vdash_f \langle M, N\rangle : A\&B$, and let $\theta \sqsubseteq^{\Gamma,\sigma}_{\texttt{sub}} \Theta$. We can invert to find that $\Gamma \vdash_f M : A$ and $\Gamma \vdash_f N : B$. Since $\langle M[\theta], N[\theta]\rangle \downarrow^{(0,0)} \langle M[\theta], N[\theta]\rangle$, to show that $\langle M[\theta], N[\theta]\rangle \sqsubseteq^{A\&B,f[\sigma]} (0, (\|M\|[\Theta], \|N\|[\Theta]))$ we must show that $0 \leq 0$ (done!) and that $\langle M[\theta], N[\theta]\rangle \sqsubseteq^{A\&B,f[\sigma]}_{\texttt{val}} (\|M\|[\Theta], \|N\|[\Theta])$. For this, it suffices by weakening to show that $M[\theta] \sqsubseteq^{A,f[\sigma]} \|M\|[\Theta]$ and $N[\theta] \sqsubseteq^{B,f[\sigma]} \|N\|[\Theta]$, which are precisely the inductive hypotheses.

(&-E) By symmetry, it suffices to present the pi_1 case. Suppose $\Gamma \vdash_f \pi_1 M : A$. By inversion, $\Gamma \vdash_f M : A\&B$. Let $\theta \sqsubseteq^{\Gamma,\sigma}_{\texttt{sub}} \Theta$. To show that $\pi_1 M[\theta] \sqsubseteq^{A,f[\sigma]} \|M\|[\Theta]_c +_c \pi_1(\|M\|[\Theta]_p)$, assume $\pi_1 M[\theta] \downarrow^{(n_1+n_2, r_1+r_2)} v$. By inversion, it was by way of $M[\theta] \downarrow^{(n_1,r_1)} \langle N_1, N_2\rangle$ and $N_1 \downarrow^{(n_2,r_2)} v$. We must show that:

- $n_1 + n_2 \leq \|M\|[\Theta]_c + (\pi_1 \|M\|[\Theta]_p)_c - (r_1 + r_2)$
- $v \sqsubseteq^{A,f[\sigma]+r_1+r_2}_{\texttt{val}} (\pi_1 \|M\|[\Theta]_p)_p$

By IH, $M[\theta] \sqsubseteq^{A\&B,f[\sigma]} \|M\|[\Theta]$, so

- $n_1 \leq \|M\|[\Theta]_c - r_1$
- $\langle N_1, N_2\rangle \sqsubseteq^{A\&B,f[\sigma]+r_1}_{\texttt{val}} \|M\|[\Theta]_p$

where the second condition means, in particular, that $N_1 \sqsubseteq^{A,f[\sigma]+r_1} \pi_1 \|M\|[\Theta]_p$. So, since $N_1 \downarrow^{(n_2,r_2)} v$,

- $n_2 \leq (\pi_1 \|M\|[\Theta]_p)_c - r_2$
- $v \sqsubseteq^{A,f[\sigma]+r_1+r_2}_{\texttt{val}} (\pi_1 \|M\|[\Theta]_p)_p$

as required.

(var) Suppose $\Gamma, x : A \vdash_{x+f} x : A$. Let $(\theta, v/x) \sqsubseteq^{(\Gamma,x:A),(\sigma,x\mapsto a)}_{\texttt{sub}} (\Theta, E/x)$. We know that $v \sqsubseteq^{A,a}_{\texttt{val}} E$. We must show that $v \sqsubseteq^{A,a+f[\sigma]} (0, E)$. We know that $v \downarrow^{(0,0)} v$. Of course, $0 \leq 0$. Since $f[\sigma] \geq 0$, we are done by credit weakening.

$\square$

THEOREM 4.1. *For any posets A, B, C, G with ∞ and $\vee$,*

(1) $\texttt{snrec} \in \mathrm{Hom}_{Poset}\left(\left(C^1\right)^G \times \left(C^{\mathbb{N}\times C}\right)^G, C^{G\times\mathbb{N}}\right)$

(2) $\texttt{slrec} \in \mathrm{Hom}_{Poset}\left(\left(C^1\right)^G \times \left(C^{A\times(\mathbb{N}\times C)}\right)^G, C^{G\times\mathbb{N}}\right)$

(3) $\texttt{scase} \in \mathrm{Hom}_{Poset}\left(C^{G\times A} \times C^{G\times B}, C^{G\times(A+B)}\right)$

Proof. Let A, B, C, G be posets. Note that these are not required to be in the image of $[\![\cdot]\!]$. For each case we must show two statements: the function is in fact montonic, and that the functions in its image (an exponential poset) are themselves monotonic.

(1) Suppose $(f,g) \leq (f',g')$ as elements of $\left(C^1\right)^G \times \left(C^{\mathbb{N}\times C}\right)^G$. To show that $\texttt{snrec}(f,g) \leq \texttt{snrec}(f',g')$, it suffices to show that for all γ, n, that $\texttt{snrec}(f,g)(\gamma,n) \leq \texttt{snrec}(f',g')(\gamma,n)$. Proceed by induction on n.
   - $n = 0$. By the definition of $\texttt{snrec}$, it suffices to show that $f(\gamma)() \leq f'(\gamma)()$, which is true since $f \leq f'$.
   - $n{+}1$ By definition of $\texttt{snrec}$, it suffces to show $g(\gamma)(n, \texttt{snrec}(f,g)(\gamma,n)) \vee f(\gamma)() \leq g'(\gamma)(n, \texttt{snrec}(f',g')(\gamma,n)) \vee f(\gamma)()$. We have already shown that $f(\gamma)() \leq f'(\gamma)()$, so it remains to show $g(\gamma)(n, \texttt{snrec}(f,g)(\gamma,n)) \leq g'(\gamma)(n, \texttt{snrec}(f',g')(\gamma,n))$. Since $g \leq g'$, $g(\gamma) \leq g'(\gamma)$. By reflexivity, $n \leq n$. By IH, $\texttt{snrec}(f,g)(\gamma,n) \leq \texttt{snrec}(f',g')(\gamma)(n)$, and so we are done.

     Now, let $(f,g) \in \left(C^1\right)^G \times \left(C^{\mathbb{N}\times C}\right)^G$. We must show that if $(\gamma,n) \leq (\gamma',n')$ in $G\times\mathbb{N}$, then $\texttt{snrec}(f,g)(\gamma,n) \leq \texttt{snrec}(f,g)(\gamma',n')$. Proceed by induction on n. We have three cases to consider.
     - $n = n' = 0$. By definition of $\texttt{snrec}$, it suffices to show that $f(\gamma)() \leq f(\gamma')()$, which is true since $\gamma \leq \gamma'$.
     - $n = 0$, $n' + 1$: By the definition of $\texttt{snrec}$, we must show that $f(\gamma)() \leq g(\gamma')(n', \texttt{snrec}(f,g)(\gamma',n')) \vee f(\gamma')()$, for which it suffices to show $f(\gamma)() \leq f(\gamma')()$, which we already argued was true.
     - $n + 1$, $n' + 1$. Expanding definitions again and simplifying, it suffices to show that $g(\gamma)(n, \texttt{snrec}(f,g)(\gamma,n)) \leq g(\gamma')(n', \texttt{snrec}(f,g)(\gamma',n'))$. Since g is monotonic, $g(\gamma) \leq g(\gamma')$. Since $n+1 \leq n'+1$, $n \leq n'$. By IH, $\texttt{snrec}(f,g)(\gamma,n) \leq \texttt{snrec}(f,g)(\gamma',n')$, and so $g(\gamma)(n, \texttt{snrec}(f,g)(\gamma,n)) \leq g(\gamma')(n', \texttt{snrec}(f,g)(\gamma',n'))$, as required.

(2) Let $(f,g) \leq (f',g) \in \left(C^1\right)^G \times \left(C^{A\times(\mathbb{N}\times C)}\right)^G$. We want to show that $\texttt{slrec}(f,g) \leq \texttt{slrec}(f',g')$. Fix $\gamma \in G$, we prove by induction on n that for all $n \in \mathbb{N}$, $\texttt{slrec}(f,g)(\gamma,n) \leq \texttt{slrec}(f',g')(\gamma,n)$.

- $n = 0$: expanding the definition of $\texttt{slrec}$, we must show that $f(\gamma)() \leq f'(\gamma)()$, which is true because $f \leq f'$.
- $n > 0$. It suffices to show that $g(\gamma)(\infty, (n, \texttt{slrec}(f, g)(\gamma, n))) \leq g'(\gamma)(\infty, (n, \texttt{slrec}(f', g')(\gamma, n)))$. Since $g \leq g'$, $g(\gamma) \leq g'(\gamma)$. Further, $\infty \leq \infty$, $n \leq n$, and by IH, $\texttt{slrec}(f, g)(\gamma, n) \leq \texttt{slrec}(f', g')(\gamma, n)$, as required.

Now, let $(f, g) \in \left(C^1\right)^G \times \left(C^{A\times(\mathbb{N}\times C)}\right)^G$. We must show that if $(\gamma, n) \leq (\gamma', n')$, $\texttt{slrec}(f, g)(\gamma, n) \leq \texttt{slrec}(f, g)(\gamma, n')$. We again prove this by induction on n. There are three cases we must consider.

- $n = n' = 0$. Immediate.
- $n = 0, n' > 0$. Identical to the similar case for $\texttt{snrec}$.
- $n, n' > 0$. To show that

$$g(\gamma)(\infty, (n, \texttt{snrec}(f, g)(\gamma, n))) \vee f(\gamma)() \leq g(\gamma')(\infty, (n', \texttt{snrec}(f, g)(\gamma', n))) \vee f(\gamma')()$$

it suffices to show that $f(\gamma) \leq f(\gamma')$ (which is true because $\gamma \leq \gamma'$ and f is monotonic) and $g(\gamma)(\infty, (n, \texttt{snrec}(f, g)(\gamma, n))) \leq g(\gamma')(\infty, (n', \texttt{snrec}(f, g)(\gamma', n')))$. Since $n + 1 \leq n' + 1$, $n \leq n'$, and so the desired result follows from IH and the fact that $g(\gamma)$

(3) Let $(f, g) \leq (f', g') \in C^{G\times A} \times C^{G\times B}$. We must show that $\texttt{scase}(f, g) \leq \texttt{scase}(f', g')$ in $C^{G\times(A+B)}$ Let $(\gamma, x) \in G \times (A+B)$. The two cases for x are symmetrical, so we consider when $x = \texttt{inl}\, a$. Then,

$$\begin{aligned} \texttt{scase}(f, g)(\gamma, \texttt{inl}\, a) &= f(\gamma, a) \vee g(\gamma, \infty) \\ &\leq f'(\gamma, a) \vee g(\gamma, \infty) \\ &= \texttt{scase}(f', g')(\gamma, \texttt{inl}\, a) \end{aligned}$$

as required.

Now, fix $(f, g) \in C^{G\times A} \times C^{G\times B}$. We must show that for all $(\gamma, x) \leq (\gamma', y)$, $\texttt{scase}(f, g)(\gamma, x) \leq \texttt{scase}(f, g)(\gamma', y)$. We have two symmetric cases to consider, so we present the case where $x = \mathtt{inl}\, a$ and $y = \mathtt{inl}\, a'$. Then,

$$\begin{aligned}\mathtt{scase}(f,g)(\gamma,\mathtt{inl}\, a) &= f(\gamma,a) \vee g(\gamma,\infty)\\ &\leq f(\gamma',a') \vee g(\gamma',\infty)\\ &= \mathtt{scase}(f,g)(\gamma',\mathtt{inl}\, a')\end{aligned}$$

as required.

□

Theorem 4.2 (Compositionality). *If $\Gamma, x : T_1 \vdash E : T_2$, and $\Gamma \vdash E' : T_1$, then* $[\![\Gamma \vdash E[E'/x] : T_2]\!] = \big(1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E' : T_1]\!]\big) ; [\![\Gamma, x : T_1 \vdash E : T_2]\!]$

Proof. By induction on $\Gamma, x : T_1 \vdash E : T_2$.

- ($\mathtt{nrec}$): Suppose $\Gamma, x : T_1 \vdash \mathtt{nrec}\,(E, E_1, E_2) : T_2$. By inversion, $\Gamma x : T_1 \vdash E : \mathbb{N}$, $\Gamma, x : T_1 \vdash E_1 : 1 \to T_2$, and $\Gamma, x : T_1 \vdash E_2 : \mathbb{N} \times T_2 \to T_2$. By IH,
  - $[\![\Gamma \vdash E[E'/x] : \mathbb{N}]\!] = (1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E' : T_1]\!]); [\![\Gamma, x : T_1 \vdash E : \mathbb{N}]\!]$
  - $[\![\Gamma \vdash E_1[E'/x] : 1 \to T_2]\!] = (1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E' : T_1]\!]); [\![\Gamma, x : T_1 \vdash E_1 : 1 \to T_2]\!]$
  - $[\![\Gamma \vdash E_2[E'/x] : \mathbb{N} \times T_2 \to T_2]\!] = (1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E' : T_1]\!]); [\![\Gamma, x : T_1 \vdash E_2 : \mathbb{N} \times T_2 \to T_2]\!]$

  . For ease of notation, we let $f = [\![\Gamma \vdash E' : T_1]\!]$, $g = [\![\Gamma, x : T_1 \vdash E : \mathbb{N}]\!]$, $h_1 = [\![\Gamma, x : T_1 \vdash E_1 : 1 \to T_2]\!]$, and $h_2 = [\![\Gamma, x : T_1 \vdash E_2 : \mathbb{N} \times T_2 \to T_2]\!]$. Then, we compute:

$$\begin{aligned}&[\![\Gamma \vdash (\mathtt{nrec}\,(E, E_1, E_2))\,[E'/x] : T_2]\!]\\ &= [\![\Gamma \vdash \mathtt{nrec}\,(E[E'/x], E_1[E'/x], E_2[E'/x])]\!]\\ &= (1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E[E'/x] : \mathbb{N}]\!]); \mathtt{snrec}([\![\Gamma \vdash E_1[E'/x] : 1 \to T_2]\!], [\![\Gamma \vdash E_2[E'/x] : \mathbb{N} \times T_2 \to T_2]\!])\\ &= (1_{[\![\Gamma]\!]}, (1_{[\![\Gamma]\!]}, f); g); \mathtt{snrec}((1_{[\![\Gamma]\!]}, f); h_1, (1_{[\![\Gamma]\!]}, f); h_2)\end{aligned}$$

It remains to show that

$$(1_{[\![\Gamma]\!]}, (1_{[\![\Gamma]\!]}, f); g); \mathtt{snrec}((1_{[\![\Gamma]\!]}, f); h_1, (1_{[\![\Gamma]\!]}, f); h_2) = (1_{[\![\Gamma]\!]}, f); (1_{[\![\Gamma, x:T_1]\!]}, g); \mathtt{snrec}(h_1, h_2)$$

Let $\gamma \in [\![\Gamma]\!]$. Applying the left hand side to γ, we get

$$\mathtt{snrec}((1_{[\![\Gamma]\!]}, f); h_1, (1_{[\![\Gamma]\!]}, f); h_1)(\gamma, g(\gamma, f(\gamma)))$$

and on the right:

$$\mathtt{snrec}(h_1, h_2)((\gamma, f(\gamma)), g(\gamma, f(\gamma)))$$

Letting $\gamma' = (\gamma, f(\gamma))$, we must show that $\texttt{snrec}((1_{[\![\Gamma]\!]}, f); h_1, (1_{[\![\Gamma]\!]}, f); h_1)(\gamma, g(\gamma')) = \texttt{snrec}(h_1, h_2)(\gamma', g(\gamma'))$. We proceed by induction on $n = g(\gamma')$.

– $n = 0$.

$$\begin{aligned}\texttt{snrec}((1_{[\![\Gamma]\!]}, f); h_1, (1_{[\![\Gamma]\!]}, f); h_1)(\gamma, 0) &= ((1_{[\![\Gamma]\!]}, f); h_1)(\gamma)() \\ &= h_1(\gamma, f(\gamma))() \\ &= h_1(\gamma')() \\ \texttt{snrec}(h_1, h_2)(\gamma', 0) &= h_1(\gamma')()\end{aligned}$$

as required.

– $n + 1$:

$$\begin{aligned}&\texttt{snrec}((1_{[\![\Gamma]\!]}, f); h_1, (1_{[\![\Gamma]\!]}, f); h_1)(\gamma, n+1) \\ &= ((1_{[\![\Gamma]\!]}, f); h_1)(\gamma)(n, \texttt{snrec}((1_{[\![\Gamma]\!]}, f); h_1, (1_{[\![\Gamma]\!]}, f); h_1)(\gamma, n)) \vee h_1(\gamma')() \\ &= h_1(\gamma')(n, \texttt{snrec}(h_1, h_2)(\gamma', n)) \vee h_1(\gamma')() \\ &= \texttt{snrec}(h_1, h_2)(\gamma', n+1)\end{aligned}$$

- ($\texttt{lrec}$): Suppose $\Gamma, x : T_1 \vdash \texttt{lrec}\,(E', E_1, E_2) : T_2$, and $\Gamma \vdash E : T_1$. By inversion, $\Gamma, x : T_1 \vdash E : \texttt{[}T\texttt{]}$, $\Gamma, x : T_1 \vdash E_1 : 1 \to T_2$, and $\Gamma, x : T_1 \vdash E_2 : T \times (\texttt{[}T\texttt{]} \times T_2) \to T_2$. By IH, we have that:
  - $[\![\Gamma \vdash E'[E/x] : \texttt{[}T\texttt{]}]\!] = (1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E : T_1]\!]); [\![\Gamma, x : T_1 \vdash E : \texttt{[}T\texttt{]}]\!]$
  - $[\![\Gamma \vdash E_1[E/x] : 1 \to T_2]\!] = (1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E : T_1]\!]); [\![\Gamma, x : T_1 \vdash E_1 : 1 \to T_]\!]$
  - $[\![\Gamma \vdash E_2[E/x] : T \times (\texttt{[}T\texttt{]} \times T_2) \to T_2]\!] = (1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E : T_1]\!]); [\![\Gamma, x : T_1 \vdash E_2 : T \times (\texttt{[}T\texttt{]} \times T_2) \to T_2]\!]$

  Let $f = [\![\Gamma \vdash E : T_1]\!]$, $g = [\![\Gamma, x : T_1 \vdash E' : \texttt{[}T\texttt{]}]\!]$, $h_1 = [\![\Gamma, x : T_1 \vdash E_1 : 1 \to T_2]\!]$, and $h_2 = [\![\Gamma, x : T_1 \vdash E : T \times (\texttt{[}T\texttt{]} \times T_2) \to T_2]\!]$

  We must show that

$$(1_{[\![\Gamma]\!]}, f); (1_{[\![\Gamma]\!] \times [\![T_1]\!]}, g); \texttt{slrec}(h_1, h_2) = (1_{[\![\Gamma]\!]}, (1_{[\![\Gamma]\!]}, f); g); \texttt{slrec}((1_{[\![\gamma]\!]}, f); h_1, (1_{[\![\gamma]\!]}, f); h_2)$$

  Let $\gamma \in [\![\Gamma]\!]$, and let $\gamma' = (\gamma, f(\gamma))$. We must then show that

$$\texttt{slrec}(h_1, h_2)(\gamma', g(\gamma')) = \texttt{slrec}((1_{[\![\gamma]\!]}, f); h_1, (1_{[\![\gamma]\!]}, f); h_2)(\gamma, g(\gamma'))$$

We proceed by induction on $n = g(\gamma')$.

– ($n = 0$): The LHS is $\texttt{slrec}(h_1, h_2)(\gamma', 0) = h_1(\gamma')()$, and the RHS is

$$\texttt{slrec}((1_{[\![\gamma]\!]}, f); h_1, (1_{[\![\gamma]\!]}, f); h_2)(\gamma, 0) = h_1(\gamma, f(\gamma))() = h_1(\gamma')()$$

.

– ($n > 0$): The LHS is:

$$\begin{aligned}&\texttt{slrec}(h_1, h_2)(\gamma', n+1)\\ &= h_2(\gamma')(\infty, (n, \texttt{slrec}(h_1, h_2)(\gamma', n))) \vee h_1(\gamma')()\end{aligned}$$

and the RHS (applying the IH in the 2nd step) is

$$\begin{aligned}&\texttt{slrec}((1_{[\![\Gamma]\!]}, f); h_1, (1_{[\![\Gamma]\!]}, f); h_2)(\gamma, n+1)\\ &= h_2(\gamma')(\infty, (n, \texttt{slrec}((1_{[\![\Gamma]\!]}, f); h_1, (1_{[\![\Gamma]\!]}, f); h_2)(\gamma, n))) \vee h_1(\gamma')()\\ &= h_2(\gamma')(\gamma, (n, \texttt{slrec}(h_1, h_2)(\gamma', n))) \vee h_1(\gamma')()\end{aligned}$$

as required.

□

Proof. By induction on $\Gamma \vdash E : T$.

- ($\texttt{nrec}$): Let $\Gamma \vdash \texttt{nrec}(E, E_1, E_2) : T$. By inversion, $\Gamma \vdash E : \mathbb{N}$, $\Gamma \vdash E_1 : 1 \to T$, and $\Gamma \vdash E_2 : \mathbb{N} \times C \to C$. By IH, $[\![\Gamma \vdash E : \mathbb{N}]\!] \in \mathrm{Hom}([\![\Gamma]\!], \mathbb{N})$. Then, $(1_\Gamma, [\![\Gamma \vdash E : \mathbb{N}]\!]) \in \mathrm{Hom}([\![\Gamma]\!], [\![\Gamma]\!] \times \mathbb{N})$. By IH, $[\![\Gamma \vdash E_1 : 1 \to T]\!] \in \mathrm{Hom}([\![\Gamma]\!], [\![T]\!]^1)$ and $[\![\Gamma \vdash E_2 : \mathbb{N} \times T \to T]\!] \in \mathrm{Hom}([\![\Gamma]\!], [\![T]\!]^{\mathbb{N} \times [\![T]\!]})$. Then, by Theorem 4.1 and composition, $(1_\Gamma, [\![\Gamma \vdash E : \mathbb{N}]\!]); \texttt{snrec}([\![\Gamma \vdash E_1 : 1 \to T]\!], [\![\Gamma \vdash E_2 : \mathbb{N} \times T \to T]\!]) \in \mathrm{Hom}([\![\Gamma]\!], [\![T]\!])$, as required.
- ($\texttt{lrec}$): Let $\Gamma \vdash \texttt{lrec}(E, E_1, E_2) : T$. By inversion, $\Gamma \vdash E : [T']$, $\Gamma \vdash E_2 : 1 \to T$, and $\Gamma \vdash E_2 : T' \times ([T'] \times T) \to T$. Applying the IH to all of these premises, we have that $[\![\Gamma \vdash E : [T']]\!] \in \mathrm{Hom}([\![\Gamma]\!], \mathbb{N})$, $[\![\Gamma \vdash E_2 : 1 \to T]\!] \in \mathrm{Hom}([\![\Gamma]\!], [\![T]\!]^1)$, and $[\![\Gamma \vdash E_2 : T' \times ([T'] \times T) \to T]\!] \in \mathrm{Hom}([\![\Gamma]\!], [\![T]\!]^{[\![T']\!] \times (\mathbb{N} \times [\![T]\!])})$. By Theorem 4.1 and composition, $(1_{[\![\Gamma]\!]}, [\![\Gamma \vdash E : [T']]\!]); \texttt{slrec}([\![\Gamma \vdash E_2 : 1 \to T]\!], [\![\Gamma \vdash E_2 : T' \times ([T'] \times T) \to T]\!]) \in \mathrm{Hom}([\![\Gamma]\!], [\![T]\!])$ as required.

□

Theorem 4.4 (Soundness (Inequality)). *If $\Gamma \vdash E \leq E'$, then for all $\gamma \in [\![\Gamma]\!]$, $[\![\Gamma \vdash E : T]\!](\gamma) \leq [\![\Gamma \vdash E' : T]\!](\gamma)$*

Proof. By induction on $\Gamma \vdash E \leq E'$. The new cases (`snrec` and `slrec`) follow easily from the definitions. □

Theorem 5.1 (Substitution).

- *If $\Delta \vdash c$ $\texttt{credit}$ and $\Delta, \alpha \vdash c'$ $\texttt{credit}$, then $\Delta \vdash c'[c/\alpha]$ $\texttt{credit}$*
- *If $\Delta \vdash c$ $\texttt{credit}$ and $\Delta, \alpha | \Gamma \vdash_f M : A$, then $\Delta | \Gamma[c/\alpha] \vdash_{f[c/\alpha]} M[c/\alpha] : A[c/\alpha]$*

Proof. By induction on $\Delta, \alpha \vdash c'$ `credit` and $\Delta, \alpha | \Gamma \vdash_f M : A$, respectively. □

Theorem 5.2 (Preservation). *If $\cdot | \cdot \vdash_a M : A$ and $M \downarrow^{(n,r)} v$, then $a + r \geq 0$ and $\cdot | \cdot \vdash_{a+r} v : A$.*

Proof. The cases for all pre-existing rules are identical– the only new cases are for `pack`, `unpack`, and `trec`. We present only the final case of `trec`, as it is the most illustritive.

- (`pack`): Suppose that $\cdot | \cdot \vdash_a \texttt{pack}_{\alpha=\ell} M : \exists \alpha$ and $\texttt{pack}_{\alpha=\ell} M \downarrow^{(n,r)} \texttt{pack}_{\alpha=\ell} v$ by way of $\cdot | \cdot \vdash_a M : A[\ell/\alpha]$ and $M \downarrow^{(n,r)} v$. By IH, $\cdot | \cdot \vdash_{a+r} v : A[\ell/\alpha]$ and $a + r \geq 0$. By the rule for `pack`, $\cdot | \cdot \vdash_{a+r} \texttt{pack}_{\alpha=\ell} v : \exists \alpha.A$, as required.
- (`unpack`): Suppose that $\cdot | \cdot \vdash_{a+b} \texttt{unpack}\ (\alpha, x) = M\ \texttt{in}\ N : C$ by way of $\cdot | \cdot \vdash_a M : \exists \alpha.A$ and $\alpha | x : A \vdash_{b+x} N : C$ with $\Delta \vdash C$, and that $\texttt{unpack}\ (\alpha, x) = M\ \texttt{in}\ N \downarrow^{(n_1+n_2, r_1+r_2)} v$ by way of $M \downarrow^{(n_1,r_1)} \texttt{pack}_{\alpha=\ell} v_1$ and $N[\ell/\alpha, v_1/x] \downarrow^{(n_2,r_2)} v$. By IH, $\cdot | \cdot \vdash_{a+r_1} v_1 : A[\ell/\alpha]$. By credit variable substituion, $\cdot | x : A[\ell/\alpha] \vdash_{b+x} N[\ell/\alpha] : C$. By substitution, $\cdot | \cdot \vdash_{b+a+r_1} N[\ell/\alpha, v_1/x] LC$ By IH, $\cdot | \cdot \vdash_{a+b+r_1+r_2} v : C$ and $a + b + r_1 + r_2 \geq 0$ as required.
- (`trec`): Suppose:

$$\frac{\begin{array}{l} \cdot | \cdot \vdash_f M : \texttt{tree}\,(A) \\ \cdot | \cdot \vdash_{b_1} N_1 :!_0^\infty (1 \multimap C) \\ \cdot | \cdot \vdash_{b_2} N_2 :!_0^\infty (A \multimap C) \\ \cdot | \cdot \vdash_{b_3} N_3 :!_0^\infty (A \otimes \mathbb{N} \otimes A \otimes N \otimes (\texttt{tree}\,(A)\,\&C)^2 \multimap C) \\ \cdot | \cdot \vdash_{b_4} N_4 :!_0^\infty (A \otimes \mathbb{N} \otimes A \otimes N \otimes (\texttt{tree}\,(A)\,\&C)^2 \multimap C) \\ \cdot | \cdot \vdash_{b_5} N_5 :!_0^\infty (A \otimes \mathbb{N} \otimes A \otimes N \otimes A \otimes N (\texttt{tree}\,(A)\,\&C)^4 \multimap C) \end{array}}{\cdot | \cdot \vdash_{a+\sum b_i} \texttt{trec}\,(M, N_1, N_2, N_3, N_4, N_4)}$$

and

$$\frac{\begin{array}{l} M \downarrow^{(n_0,r_0)} N(v_1, n_1, N(v_2, n_2, t_{00}, t_{01}), N(v_3, n_7, t_{10}, t_{11})) \\ N_i \downarrow^{(n_i,r_i)} \mathtt{save}_0^\infty\ v_i' \qquad (1 \leq i \leq 4) \\ N_5 \downarrow^{(n_5,r_5)} \mathtt{save}_0^\infty\ (\lambda x.N_5') \\ N_5'[(v_1, n_1, v_2, n_2, v_3, n_3, \langle t_{00}, \mathtt{trec}(t_{00}, \mathtt{save}_0^\infty\ v_1', \ldots, \mathtt{save}_0^\infty\ (\lambda x.N_5')), , \rangle \ldots)/x] \end{array}}{\mathtt{trec}\,(M, N_1, N_2, N_3, N_4, N_5) \downarrow^{(\sum n_i, \sum r_i)} v}$$

By IH, $\cdot|\cdot \vdash_{a+r_0} N(\ldots)$. Hence, there are $d_1, \ldots, d_n$, all non-negative, so that $\sum d_i = a + r_0$, and $\cdot|\cdot \vdash_{d_i} w_i : A_i$ where w_i is the ith value in the value which M evaluates to (in particular, $\cdot|\cdot \vdash_{d_1} v_1 : A$, and $\cdot|\cdot \vdash_{d_6} t_{00} : \mathtt{tree}\,(A)$). Again by IH, there are $c_1, \ldots, c_5$ so that $\infty c_i \leq b_i + r_i$, with $\cdot|\cdot \vdash_{c_i} v_i'$. Thus, $\cdot|\cdot \vdash_{d_6 + \sum c_i} \langle t_{00}, \mathtt{trec}(t_0 0, \mathtt{save}_0^\infty\ v_1', \ldots)\rangle$, and similarly for the rest of the subtrees. This immediately implies

$$\cdot|\cdot \vdash_{\sum d_i + 4\sum c_i} (v_1, n_1, v_2, n_2, v_3, n_3, \langle t_{00}, \mathtt{trec}(t_0 0, \mathtt{save}_0^\infty\ v_1', \ldots)\rangle, \ldots) : (A \otimes \mathbb{N})^3 \otimes (\mathtt{tree}\,(A)\ \& C)^4$$

then by substitution

$$\cdot|\cdot \vdash_{\sum d_i + c_5 + 4\sum c_i} N_5'[(v_1, n_1, v_2, n_2, v_3, n_3, \langle t_{00}, \mathtt{trec}(t_0 0, \mathtt{save}_0^\infty\ v_1', \ldots)\rangle, \ldots)/x] : C$$

The result follows immediately by weakening ($\infty c_i \leq b_i + r_i$) and IH.

□

Theorem 5.3 (Extraction Preserves Types). *If* $\Delta|\Gamma \vdash_f M : A$, *then* $\langle\!\langle \Delta \rangle\!\rangle, \langle\!\langle \Gamma \rangle\!\rangle \vdash \|M\| : \|A\|$

Proof. By induction on $\Delta|\Gamma \vdash_f M : A$. □

Theorem 5.4 (Bounding Theorem). *If* $\Delta|\Gamma \vdash_f M : A$, *then* $M \sqsubseteq^A \|M\|$

Proof.

- ($\mathtt{pack}$): Suppose $\Delta|\Gamma \vdash_f \mathtt{pack}_{\alpha=c} M : \exists\alpha.A$ by way of $\Delta|\Gamma \vdash_f M : A[c/\alpha]$. Let $\omega \sqsubseteq^{\Delta}_{\mathtt{credit}} \Omega$ and $\theta \sqsubseteq^{\Gamma[\omega],\sigma}_{\mathtt{sub}} \Theta$. We must show that $\mathtt{pack}_{\alpha=c[\omega]} M[\omega,\theta] \sqsubseteq^{\exists\alpha.A[\omega], f[\omega,\sigma]} (\|M\|_c[\Omega,\Theta], (c[\Omega], \|M\|_p[\Omega,\Theta]))$. Suppose $\mathtt{pack}_{\alpha=c[\omega]} M[\omega,\theta] \downarrow^{(n,r)} \mathtt{pack}_{\alpha=c[\omega]} v$ by way of $M[\omega,\theta] \downarrow^{(n,r)} v$. It suffices to show
  - $n + r \leq \|M\|_c[\Omega,\Theta]$
  - $\mathtt{pack}_{\alpha=c[\omega]} v \sqsubseteq^{\exists\alpha.A[\omega], f[\omega,\sigma]+r}_{\mathtt{val}} (c[\Omega], \|M\|_p[\Omega,\Theta])$

The second item is equivalent to proving that $c[\omega] \le c[\Omega]$ (which is true because credit terms are monotone), and that $v \sqsubseteq_{\texttt{val}}^{A[c/\alpha,\omega],f[\omega,\sigma]+r} \|M\|_p[\Omega,\Theta]$, which follows immediately by IH.

- ($\texttt{unpack}$): Suppose that $\Delta|\Gamma \vdash_{f+g} \texttt{unpack } (\alpha,x) = M \texttt{ in } N : C$ by way of $\Delta|\Gamma \vdash_f M : \exists\alpha.A$ and $\Delta,\alpha|\Gamma,x : A \vdash_{g+x} N : C$ with α not free in C. Let $\omega \sqsubseteq_{\texttt{credit}}^{\Delta} \Omega$ and $\theta \sqsubseteq_{\texttt{sub}}^{\Gamma[\omega],\sigma} \Theta$. Suppose $\texttt{unpack } (\alpha,x) = M[\omega,\theta] \texttt{ in } N[\omega,\theta] \downarrow^{(n_1+n_2,r_1+r_2)} v$ by way of $M[\omega,\theta] \downarrow^{(n_1,r_1)} \texttt{pack}_{\alpha=\ell}v_1$ and $N[\omega,\theta,\ell/\alpha,v_1/x] \downarrow^{(n_2,r_2)} v$. It suffices to show that
  - $n_1+n_2+r_1+r_2 \le \|M\|_c[\Omega,\Theta] + \|N\|_c[\Omega,\Theta,\pi_1\|M\|_p[\Omega,\Theta]\alpha,\pi_2\|M\|_p[\Omega,\Theta]/x]$
  - $v \sqsubseteq_{\texttt{val}}^{C[\omega],f[\omega,\sigma]+g[\omega,\sigma]+r_1+r_2} \|N\|_p[\Omega,\Theta,\pi_1\|M\|_p[\Omega,\Theta]\alpha,\pi_2\|M\|_p[\Omega,\Theta]/x]$

  By IH, $M[\omega,\theta] \sqsubseteq^{\exists\alpha.A[\omega],f[\omega,\sigma]} \|M\|[\Omega,\Theta]$, and so
  - $n_1+r_1 \le \|M\|_c[\Omega,\Theta]$
  - $\texttt{pack}_{\alpha=\ell}v_1 \sqsubseteq_{\texttt{val}}^{\exists\alpha.A[omega],f[\omega,\sigma]+r_1} \|M\|_p[\Omega,Theta]$

  which means that $\ell \le \pi_1\|M\|_p[\Omega,Theta]$ and that $v_1 \sqsubseteq_{\texttt{val}}^{A[\omega,\ell/\alpha],f[\omega,\sigma]+r_1} \pi_2\|M\|_p[\Omega,Theta]$. Hence, $(\omega,\ell/\alpha) \sqsubseteq_{\texttt{credit}}^{\Delta,\alpha} (\Omega,\pi_1\|M\|_p[\Omega,\Theta]/\alpha)$, and $(\theta,v_1/x) \sqsubseteq_{\texttt{sub}}^{\Gamma[\omega],x:A[\ell/\alpha],\sigma,x\mapsto f[\omega,\sigma]+r_1} (\Theta,\pi_2\|M\|_p[\Omega,\Theta]/x)$. Thus, by IH,

$$N[\omega,\theta,\ell/\alpha,v_1/x] \sqsubseteq^{C[\omega],g[\omega,\sigma]+f[\omega,\sigma]+r_1} \|N\|[\Omega,\Theta,\pi_1\|M\|_p[\Omega,\Theta]/\alpha,\pi_2\|M\|_p[\Omega,\Theta]/x]$$

  By definition,
  - $n_2+r_2 \le \|N\|_c[\Omega,\Theta,\pi_1\|M\|_p[\Omega,\Theta]/\alpha,\pi_2\|M\|_p[\Omega,\Theta]/x]$
  - $v \sqsubseteq_{\texttt{val}}^{C[\omega],f[\omega,\sigma]+g[\omega,\sigma]+r_1+r_2} \|N\|_p[\Omega,\Theta,\pi_1\|M\|_p[\Omega,\Theta]/\alpha,\pi_2\|M\|_p[\Omega,\Theta]/x]$

  as required.

$\square$

$$\lambda^{A} \text{ rules:} \qquad \frac{}{\Delta|\Gamma \vdash_f \texttt{Emp} : \texttt{tree}\,(A)}$$

$$\frac{\Delta|\Gamma \vdash_{f_1} M_1 : A \quad \Delta|\Gamma \vdash_{f_2} M_2 : \mathbb{N} \quad \Delta|\Gamma \vdash_{g_1} N_1 : \texttt{tree}\,(A) \quad \Delta|\Gamma \vdash_{g_2} N_2 : \texttt{tree}\,(A)}{\Delta|\Gamma \vdash_{f_1+f_2+g_1+g_2} N(M_1, M_2, N_1, N_2) : \texttt{tree}\,(A)}$$

$$\frac{\begin{array}{l}
\Delta|\Gamma \vdash_f M : \texttt{tree}\,(A) \\
\Delta|\Gamma \vdash_{g_1} N_1 :!_0^{\infty}(1 \multimap C) \\
\Delta|\Gamma \vdash_{g_2} N_2 :!_0^{\infty}(A \multimap C) \\
\Delta|\Gamma \vdash_{g_3} N_3 :!_0^{\infty}(A \otimes \mathbb{N} \otimes A \otimes N \otimes (\texttt{tree}\,(A)\,\&C)^2 \multimap C) \\
\Delta|\Gamma \vdash_{g_4} N_4 :!_0^{\infty}(A \otimes \mathbb{N} \otimes A \otimes N \otimes (\texttt{tree}\,(A)\,\&C)^2 \multimap C) \\
\Delta|\Gamma \vdash_{g_5} N_5 :!_0^{\infty}(A \otimes \mathbb{N} \otimes A \otimes N \otimes A \otimes N(\texttt{tree}\,(A)\,\&C)^4 \multimap C)
\end{array}}{\Delta|\Gamma \vdash_{f+\sum_{i=1}^{5} g_i} \texttt{trec}\,(M, N_1, N_2, N_3, N_4, N_5) : C}$$

$$\frac{\begin{array}{l}
M \downarrow^{(n_1,r_1)} N(v_1, n_1, N(v_2, n_2, t_{00}, t_{01}), N(v_3, n_3, t_{10}, t_{11})) \\
N_1 \downarrow^{(n_2,r_2)} v_1' \\
N_2 \downarrow^{(n_3,r_3)} v_2' \\
N_3 \downarrow^{(n_4,r_4)} v_3' \\
N_4 \downarrow^{(n_5,r_5)} v_4' \\
N_5 \downarrow^{(n_6,r_6)} \texttt{save}_0^{\infty}\ \lambda x.N_5' \\
N_5'\,[(v_1, n_1, v_2, n_2, v_3, n_3, \langle t_{00}, \texttt{trec}\,(t_{00}, v_1', v_2', v_3', v_4', \texttt{save}_0^{\infty}\ \lambda x.N_5')\rangle, \dots)\,/x] \downarrow^{(n_7,r_7)} v
\end{array}}{\texttt{trec}\,(M, N_1, N_2, N_3, N_4, N_5) \downarrow^{(\Sigma_{i=1}^{7} n_i, \Sigma_{i=1}^{7} r_i)} v}$$

$$\lambda^{\mathbb{C}} \text{ rules:} \qquad \frac{}{\Gamma \vdash \texttt{Emp} : \texttt{tree}\,(T)}$$

$$\frac{\Gamma \vdash E_1 : T \quad \Gamma \vdash E_2 : \mathbb{N} \quad \Gamma \vdash E_1' : \texttt{tree}\,(T) \quad \Gamma \vdash E_2' : \texttt{tree}\,(T)}{\Gamma \vdash N(E_1, E_2, E_1', E_2') : \texttt{tree}\,(T)}$$

$$\frac{\begin{array}{l}
\Gamma \vdash E : \texttt{tree}\,(T) \\
\Gamma \vdash E_1 : 1 \to T' \\
\Gamma \vdash E_2 : T \to T' \\
\Gamma \vdash E_3 : A \times \mathbb{N} \times A \times \mathbb{N} \times (\texttt{tree}\,(T) \times T')^2 \to T' \\
\Gamma \vdash E_4 : A \times \mathbb{N} \times A \times \mathbb{N} \times (\texttt{tree}\,(T) \times T')^2 \to T' \\
\Gamma \vdash E_5 : A \times \mathbb{N} \times A \times \mathbb{N} \times A \times \mathbb{N} \times (\texttt{tree}\,(T) \times T')^4 \to T'
\end{array}}{\Gamma \vdash \texttt{trec}\,(E, E_1, E_2, E_3, E_4, E_5) : T'}$$

$$\begin{aligned}
&\|\texttt{trec}\,(M, N_1, N_2, N_3, N_4, N_5)\| = \left(\|M\|_c + \textstyle\sum_{i=1}^{5} \|N_i\|_c\right) +_c \\
&\quad \texttt{trec}(\|M\|_p, \|N_1\|_p, \|N_2\|_p, \lambda(x, n_1, y, n_2, (r_1, t_1), (r_2, t_2)).\, \|N_3\|_p\,(x_1, n_1, y, n_2, ((0, r_1), t_1), ((0, r_2), t_2)), \dots)
\end{aligned}$$