CIS 500

Software Foundations

Fall 2003

19 November

Plans

- Class will be held as usual next Monday and Wednesday.
- Recitations and office hours next week will not be held on Wednesday afternoon, Thursday, or Friday.
- There will not be a homework assignment over Thanksgiving weekend.

Announcement

On Friday, Penn is hosting the New Jersey Programming Languages Seminar.

If you'd like to attend any of the talks, feel free!

Metatheory of Typing

Issue

For the typing relation, we have just one problematic rule to deal with: subsumption.

$$\frac{\Gamma \vdash \mathtt{t} : \mathtt{S} \qquad \mathtt{S} <: \mathtt{T}}{\Gamma \vdash \mathtt{t} : \mathtt{T}} \quad \text{(T-Sub)}$$

We observed last time that this rule is sometimes required when typechecking applications:

E.g., the term

```
(λr:{x:Nat}. r.x) {x=0,y=1}
```

is *not* typable without using subsumption.

But we conjectured that applications were the *only* critical uses of subsumption.

Plan

1. Investigate how subsumption is used in typing derivations by looking at examples of how it can be "pushed through" other rules
2. Use the intuitions gained from this exercise to design a new, *algorithmic* typing relation that
   - omits subsumption
   - compensates for its absence by enriching the application rule
3. Show that the algorithmic typing relation is essentially equivalent to the original, declarative one

Example (T-Sub with T-Abs)

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma, \mathtt{x}:\mathtt{S}_1 \vdash \mathtt{s}_2 : \mathtt{S}_2} \qquad \dfrac{\vdots}{\mathtt{S}_2 <: \mathtt{T}_2}}{\Gamma, \mathtt{x}:\mathtt{S}_1 \vdash \mathtt{s}_2 : \mathtt{T}_2} \text{(T-Sub)}}{\Gamma \vdash \lambda \mathtt{x}:\mathtt{S}_1.\mathtt{s}_2 : \mathtt{S}_1 \to \mathtt{T}_2} \text{(T-Abs)}$$

Example (T-Sub with T-Abs)

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma, \mathtt{x}:\mathtt{S}_1 \vdash \mathtt{s}_2 : \mathtt{S}_2} \qquad \dfrac{\vdots}{\mathtt{S}_2 <: \mathtt{T}_2}}{\Gamma, \mathtt{x}:\mathtt{S}_1 \vdash \mathtt{s}_2 : \mathtt{T}_2} \text{(T-Sub)}}{\Gamma \vdash \lambda \mathtt{x}:\mathtt{S}_1.\mathtt{s}_2 : \mathtt{S}_1 \to \mathtt{T}_2} \text{(T-Abs)}$$

becomes

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma, \mathtt{x}:\mathtt{S}_1 \vdash \mathtt{s}_2 : \mathtt{S}_2}}{\Gamma \vdash \lambda \mathtt{x}:\mathtt{S}_1.\mathtt{s}_2 : \mathtt{S}_1 \to \mathtt{S}_2} \text{(T-Abs)} \qquad \dfrac{\dfrac{}{\mathtt{S}_1 <: \mathtt{S}_1} \text{(S-Refl)} \qquad \dfrac{\vdots}{\mathtt{S}_2 <: \mathtt{T}_2}}{\mathtt{S}_1 \to \mathtt{S}_2 <: \mathtt{S}_1 \to \mathtt{T}_2} \text{(S-Arrow)}}{\Gamma \vdash \lambda \mathtt{x}:\mathtt{S}_1.\mathtt{s}_2 : \mathtt{S}_1 \to \mathtt{T}_2} \text{(T-Sub)}$$

Example (T-Sub with T-Rcd)

[board]

Intuitions

These examples show that we *do not* need T-Sub to "enable" T-Abs or T-Rcd: given any typing derivation, we can construct a derivation with the same conclusion in which T-Sub is never used immediately before T-Abs or T-Rcd.

What about T-App?

We've already observed that T-Sub is required for typechecking some applications. So we expect to find that we cannot play the same game with T-App as we've done with T-Abs and T-Rcd. Let's see why.

Example (T-Sub with T-App on the left)

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash s_1 : S_{11}\to S_{12}} \quad \dfrac{\dfrac{\vdots}{T_{11} <: S_{11}} \quad \dfrac{\vdots}{S_{12} <: T_{12}}}{S_{11}\to S_{12} <: T_{11}\to T_{12}}\;(\text{S-Arrow})}{\Gamma \vdash s_1 : T_{11}\to T_{12}}\;(\text{T-Sub}) \quad \dfrac{\vdots}{\Gamma \vdash s_2 : T_{11}}}{\Gamma \vdash s_1\; s_2 : T_{12}}\;(\text{T-App})$$

Example (T-Sub with T-App on the left)

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash s_1 : S_{11}\to S_{12}} \quad \dfrac{\dfrac{\vdots}{T_{11} <: S_{11}} \quad \dfrac{\vdots}{S_{12} <: T_{12}}}{S_{11}\to S_{12} <: T_{11}\to T_{12}}\;(\text{S-Arrow})}{\Gamma \vdash s_1 : T_{11}\to T_{12}}\;(\text{T-Sub}) \quad \dfrac{\vdots}{\Gamma \vdash s_2 : T_{11}}}{\Gamma \vdash s_1\; s_2 : T_{12}}\;(\text{T-App})$$

becomes

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash s_1 : S_{11}\to S_{12}} \quad \dfrac{\dfrac{\vdots}{\Gamma \vdash s_2 : T_{11}} \quad \dfrac{\vdots}{T_{11} <: S_{11}}}{\Gamma \vdash s_2 : S_{11}}\;(\text{T-Sub})}{\Gamma \vdash s_1\; s_2 : S_{12}}\;(\text{T-App}) \quad \dfrac{\vdots}{S_{12} <: T_{12}}}{\Gamma \vdash s_1\; s_2 : T_{12}}\;(\text{T-Sub})$$

Example (T-Sub with T-App on the right)

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash s_1 : T_{11} \to T_{12}} \qquad \dfrac{\dfrac{\vdots}{\Gamma \vdash s_2 : T_2} \quad \dfrac{\vdots}{T_2 <: T_{11}}}{\Gamma \vdash s_2 : T_{11}}\ (\text{T-Sub})}{\Gamma \vdash s_1\ s_2 : T_{12}}\ (\text{T-App})$$

Example (T-Sub with T-App on the right)

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash s_1 : T_{11} \to T_{12}} \qquad \dfrac{\dfrac{\vdots}{\Gamma \vdash s_2 : T_2} \quad \dfrac{\vdots}{T_2 <: T_{11}}}{\Gamma \vdash s_2 : T_{11}}\ (\text{T-Sub})}{\Gamma \vdash s_1\ s_2 : T_{12}}\ (\text{T-App})$$

becomes

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash s_1 : T_{11} \to T_{12}} \qquad \dfrac{\dfrac{\vdots}{T_2 <: T_{11}} \quad \dfrac{}{T_{12} <: T_{12}}\ (\text{S-Refl})}{T_{11} \to T_{12} <: T_2 \to T_{12}}\ (\text{S-Arrow})}{\Gamma \vdash s_1 : T_2 \to T_{12}}\ (\text{T-Sub}) \qquad \dfrac{\vdots}{\Gamma \vdash s_2 : T_2}}{\Gamma \vdash s_1\ s_2 : T_{12}}\ (\text{T-App})$$

Intuitions

So we've seen that uses of subsumption can be "pushed" from one of immediately before T-App's premises to the other, but cannot be completely eliminated.

Example (nested uses of T-Sub)

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash s : S} \quad \dfrac{\vdots}{S <: U}}{\Gamma \vdash s : U}\ (\text{T-Sub}) \qquad \dfrac{\vdots}{U <: T}}{\Gamma \vdash s : T}\ (\text{T-Sub})$$

Example (nested uses of T-Sub)

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash s : S} \qquad \dfrac{\vdots}{S <: U}}{\Gamma \vdash s : U}\;(\text{T-Sub}) \qquad \dfrac{\vdots}{U <: T}}{\Gamma \vdash s : T}\;(\text{T-Sub})$$

becomes

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash s : S} \qquad \dfrac{\dfrac{\vdots}{S <: U} \qquad \dfrac{\vdots}{U <: T}}{S <: T}\;(\text{S-Trans})}{\Gamma \vdash s : T}\;(\text{T-Sub})$$

Summary

What we've learned:

- Uses of the T-Sub rule can be "pushed down" through typing derivations until they encounter either
  1. a use of T-App or
  2. the root fo the derivation tree.
- In both cases, multiple uses of T-Sub can be collapsed into a single one.

Summary

What we've learned:

- Uses of the T-Sub rule can be "pushed down" through typing derivations until they encounter either
  1. a use of T-App or
  2. the root fo the derivation tree.
- In both cases, multiple uses of T-Sub can be collapsed into a single one.

This suggests a notion of "normal form" for typing derivations, in which there is

- exactly one use of T-Sub before each use of T-App
- one use of T-Sub at the very end of the derivation
- no uses of T-Sub anywhere else.

Algorithmic Typing

The next step is to "build in" the use of subsumption in application rules, by changing the T-App rule to incorporate a subtyping premise.

$$\dfrac{\Gamma \vdash t_1 : T_{11} \to T_{12} \qquad \Gamma \vdash t_2 : T_2 \qquad \vdash T_2 <: T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}}$$

Given any typing derivation, we can now

1. normalize it, to move all uses of subsumption to either just before applications (in the right-hand premise) or at the very end
2. replace uses of T-App with T-Sub in the right-hand premise by uses of the extended rule above

This yields a derivation in which there is just one use of subsumption, at the very end!

Minimal Types

But... if subsumption is only used at the very end of derivations, then it is actually not needed in order to show that any term is typable!

It is just used to give more types to terms that have already been shown to have a type.

In other words, if we dropped subsumption completely (after refining the application rule), we would still be able to give types to exactly the same set of terms — we just would not be able to give as many types to some of them.

If we drop subsumption, then the remaining rules will assign a unique, minimal type to each typable term.

For purposes of building a typechecking algorithm, this is enough.

Final Algorithmic Typing Rules

$$\frac{x{:}T \in \Gamma}{\Gamma \mapsto x : T} \qquad \text{(TA-VAR)}$$

$$\frac{\Gamma, x{:}T_1 \mapsto t_2 : T_2}{\Gamma \mapsto \lambda x{:}T_1.t_2 : T_1{\to}T_2} \qquad \text{(TA-ABS)}$$

$$\frac{\Gamma \mapsto t_1 : T_1 \qquad T_1 = T_{11}{\to}T_{12} \qquad \Gamma \mapsto t_2 : T_2 \qquad \mapsto T_2 <: T_{11}}{\Gamma \mapsto t_1\ t_2 : T_{12}} \qquad \text{(TA-APP)}$$

$$\frac{\text{for each } i \qquad \Gamma \mapsto t_i : T_i}{\Gamma \mapsto \{l_1{=}t_1 \ldots l_n{=}t_n\} : \{l_1{:}T_1 \ldots l_n{:}T_n\}} \qquad \text{(TA-RCD)}$$

$$\frac{\Gamma \mapsto t_1 : R_1 \qquad R_1 = \{l_1{:}T_1 \ldots l_n{:}T_n\}}{\Gamma \mapsto t_1.l_i : T_i} \qquad \text{(TA-PROJ)}$$

Soundness of the algorithmic rules

Theorem: If $\Gamma \mapsto t : T$, then $\Gamma \vdash t : T$.

Completeness of the algorithmic rules

Theorem [Minimal Typing]: If $\Gamma \vdash t : T$, then $\Gamma \mapsto t : S$ for some $S <: T$.

Completeness of the algorithmic rules

Theorem [Minimal Typing]: If $\Gamma \vdash t : T$, then $\Gamma \mapsto t : S$ for some $S <: T$.

Proof: Homework.

(N.b.: All the messing around with transforming derivations was just to build intuitions and decide what algorithmic rules to write down and what property to prove: the proof itself is a straightforward induction on typing derivations.)

Meets and Joins

A Problem with Conditional Expressions

Calculating Meets and Joins