

Issue

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

$$\frac{\Gamma \vdash t: s \quad s \lt: T}{\Gamma \vdash t: T}$$
(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.

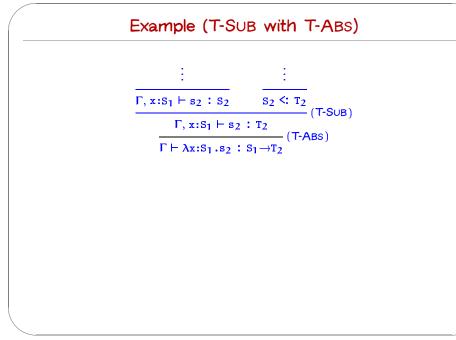
CIS 500, 19 November

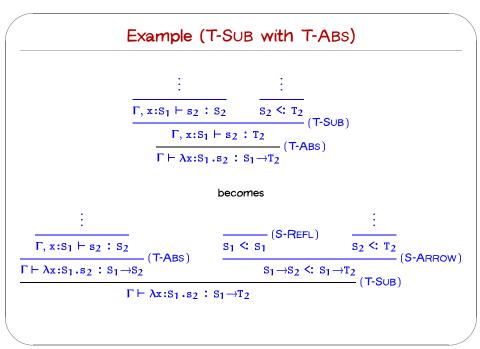
5



- 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

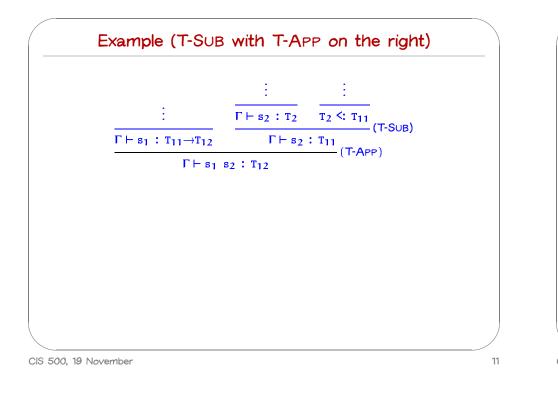
CIS 500, 19 November

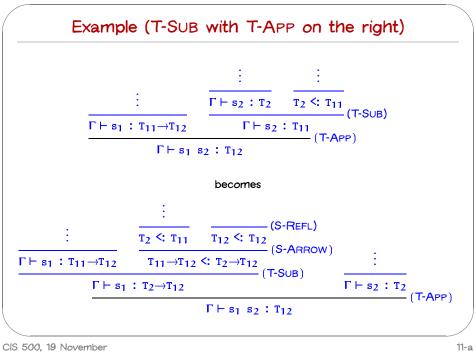




7

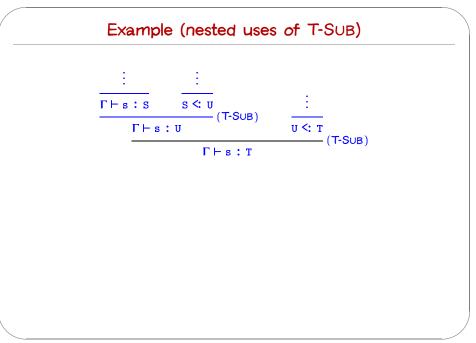


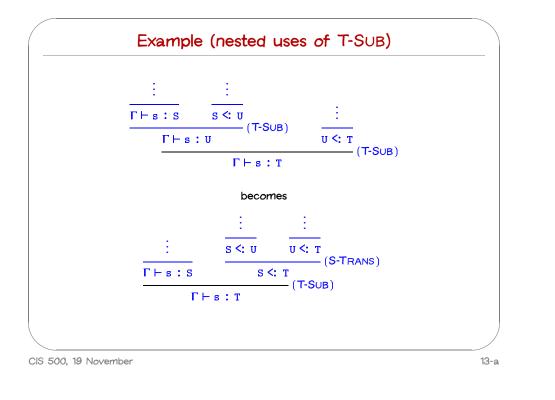




Intuitions

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.





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.

CIS 500, 19 November

14

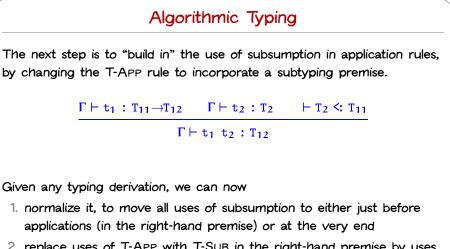
Summary

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
- \blacklozenge no uses of T-SUB anywhere else.



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.

CIS 500, 19 November

16

Final Algorithmic Typing Rules $x:T \in \Gamma$ (TA-VAR) $\Gamma \vdash x : T$ Γ , x:T₁ \vdash t₂ : T₂ (TA-ABS) $\Gamma \vdash \lambda x: T_1.t_2 : T_1 \rightarrow T_2$ $\Gamma \models t_1 : T_1 \qquad T_1 = T_{11} \rightarrow T_{12} \qquad \Gamma \models t_2 : T_2$ ▶ T₂ <: T₁₁ (TA-APP) **Γ** → t₁ t₂ : T₁₂ for each $i \quad \Gamma \vdash t_i : T_i$ (TA-Rcd) $\Gamma \mapsto \{1_1 = t_1 \dots 1_n = t_n\} : \{1_1 : T_1 \dots 1_n : T_n\}$ $\Gamma \mapsto t_1 : R_1 \qquad R_1 = \{l_1 : T_1 \dots l_n : T_n\}$ (TA-PROJ) $\Gamma \models t_1.1_i : T_i$

CIS 500, 19 November

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 $\Gamma \vdash t$: S for some S <: T.

Completeness of the algorithmic rules

Theorem [Minimal Typing]: If $\Gamma \vdash t$: T, then $\Gamma \vdash t$: 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	
<		

CIS 500, 19 November

19-a

CIS 500, 19 November

A Problem with Conditional Expressions

Calculating Meets and Joins