CIS 500

Software Foundations

Fall 2003

17 November

Administrivia

- Midterm results available in LVN 302
- Rough grade breakdown:

  65-80: A (22%)
  48-64: B (33%)
  35-49: C (30%)
  $\leq$34: D/F (20%)

  60+ points is on-target for WPE-I
- Grading questions? See your TA.

Comments on Exam

- Performance on "proof" parts was very bimodal (and strongly correlated with ability to draw derivation trees!)
- A number of people did poorly on the final question, which was taken verbatim from a recent homework.
  - There will be a homework problem on the final exam.
- In general, MT2 is a good indication of the difficulty of the final exam

Subtyping (Review)

Subtype relation

$$S <: S \qquad \text{(S-Refl)}$$

$$\frac{S <: U \qquad U <: T}{S <: T} \qquad \text{(S-Trans)}$$

$$\{l_i{:}T_i\ ^{i\in 1..n+k}\} <: \{l_i{:}T_i\ ^{i\in 1..n}\} \qquad \text{(S-RcdWidth)}$$

$$\frac{\text{for each } i \quad S_i <: T_i}{\{l_i{:}S_i\ ^{i\in 1..n}\} <: \{l_i{:}T_i\ ^{i\in 1..n}\}} \qquad \text{(S-RcdDepth)}$$

$$\frac{\{k_j{:}S_j\ ^{j\in 1..n}\} \text{ is a permutation of } \{l_i{:}T_i\ ^{i\in 1..n}\}}{\{k_j{:}S_j\ ^{j\in 1..n}\} <: \{l_i{:}T_i\ ^{i\in 1..n}\}} \qquad \text{(S-RcdPerm)}$$

$$\frac{T_1 <: S_1 \qquad S_2 <: T_2}{S_1 \to S_2 <: T_1 \to T_2} \qquad \text{(S-Arrow)}$$

$$S <: \mathtt{Top} \qquad \text{(S-Top)}$$

Subsumption Rule

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

Other typing rules as in $\lambda_{\to}$

Metatheory of Subtyping (Preview)

Syntax-directed rules

In the simply typed lambda-calculus (without subtyping), each rule can be "read from bottom to top" in a straightforward way.

$$\frac{\Gamma \vdash t_1 : T_{11}{\to}T_{12} \qquad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}} \qquad \text{(T-App)}$$

If we are given some Γ and some t of the form $t_1\ t_2$, we can try to find a type for t by

1. finding (recursively) a type for t_1
2. checking that it has the form $T_{11}{\to}T_{12}$
3. finding (recursively) a type for t_2
4. checking that it is the same as T_{11}

Technically, the reason this works is that We can divide the "positions" of the typing relation into input positions (Γ and t) and output positions (T).

- For the input positions, all metavariables appearing in the premises also appear in the conclusion (so we can calculate inputs to the "subgoals" from the subexpressions of inputs to the main goal)
- For the output positions, all metavariables appearing in the conclusions also appear in the premises (so we can calculate outputs from the main goal from the outputs of the subgoals)

$$\frac{\Gamma \vdash t_1 : T_{11}{\to}T_{12} \qquad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}} \qquad \text{(T-App)}$$

Syntax-directed sets of rules

The second important point about the simply typed lambda-calculus is that the set of typing rules is syntax-directed, in the sense that, for every "input" Γ and t, there *one* rule that can be used to derive typing statements involving t.

E.g., if t is an application, then we must proceed by trying to use T-App. If we succeed, then we have found a type (indeed, the unique type) for t. If it fails, then we know that t is not typable.

$\longrightarrow$ no backtracking!

Non-syntax-directedness of typing

When we extend the system with subtyping, both aspects of syntax-directedness get broken.

1. The set of typing rules now includes two rules that can be used to give a type to terms of a given shape (the old one plus T-Sub)

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

2. Worse yet, the new rule T-Sub itself is *not* syntax directed: the inputs to the left-hand subgoal are exactly the same as the inputs to the main goal!

   (Hence, if we translated the typing rules naively into a typechecking function, the case corresponding to T-Sub would cause divergence.)

Non-syntax-directedness of subtyping

Moreover, the subtyping relation is not syntax directed either.

1. There are lots of ways to derive a given subtyping statement.
2. The transitivity rule

$$\frac{\texttt{S <: U} \qquad \texttt{U <: T}}{\texttt{S <: T}} \qquad \text{(S-TRANS)}$$

   is badly non-syntax-directed: the premises contain a metavariable (in an "input position") that does not appear at all in the conclusion.

   To implement this rule naively, we'd have to guess a value for `U`!

What to do?

What to do?

1. Observation: We don't need 1000 ways to prove a given typing or subtyping statement — one is enough.

   $\longrightarrow$ Think more carefully about the typing and subtyping systems to see where we can get rid of excess flexibility
2. Use the resulting intuitions to formulate new "algorithmic" (i.e., syntax-directed) typing and subtyping relations
3. Prove that the algorithmic relations are "the same as" the original ones in an appropriate sense.

Metatheory of Subtyping

Subtype relation

$$\mathtt{S} <: \mathtt{S} \qquad \text{(S-Refl)}$$

$$\frac{\mathtt{S} <: \mathtt{U} \qquad \mathtt{U} <: \mathtt{T}}{\mathtt{S} <: \mathtt{T}} \qquad \text{(S-Trans)}$$

$$\{\mathtt{l}_i{:}\mathtt{T}_i\ ^{i\in 1..n+k}\} <: \{\mathtt{l}_i{:}\mathtt{T}_i\ ^{i\in 1..n}\} \qquad \text{(S-RcdWidth)}$$

$$\frac{\text{for each } i \quad \mathtt{S}_i <: \mathtt{T}_i}{\{\mathtt{l}_i{:}\mathtt{S}_i\ ^{i\in 1..n}\} <: \{\mathtt{l}_i{:}\mathtt{T}_i\ ^{i\in 1..n}\}} \qquad \text{(S-RcdDepth)}$$

$$\frac{\{\mathtt{k}_j{:}\mathtt{S}_j\ ^{j\in 1..n}\} \text{ is a permutation of } \{\mathtt{l}_i{:}\mathtt{T}_i\ ^{i\in 1..n}\}}{\{\mathtt{k}_j{:}\mathtt{S}_j\ ^{j\in 1..n}\} <: \{\mathtt{l}_i{:}\mathtt{T}_i\ ^{i\in 1..n}\}} \qquad \text{(S-RcdPerm)}$$

$$\frac{\mathtt{T}_1 <: \mathtt{S}_1 \qquad \mathtt{S}_2 <: \mathtt{T}_2}{\mathtt{S}_1 \to \mathtt{S}_2 <: \mathtt{T}_1 \to \mathtt{T}_2} \qquad \text{(S-Arrow)}$$

$$\mathtt{S} <: \mathtt{Top} \qquad \text{(S-Top)}$$

Issues

For a given subtyping statement, there are multiple rules that could be used last in a derivation.

1. S-Rcd-Width, S-Rcd-Depth, and S-Rcd-Perm overlap with each other
2. S-Refl and S-Trans overlap with everything

Step 1: simplify record subtyping

Idea: combine all three record subtyping rules into one "macro rule" that captures all of their effects

$$\frac{\{\mathtt{l}_i\ ^{i\in 1..n}\} \subseteq \{\mathtt{k}_j\ ^{j\in 1..m}\} \qquad \mathtt{k}_j = \mathtt{l}_i \text{ implies } \mathtt{S}_j <: \mathtt{T}_i}{\{\mathtt{k}_j{:}\mathtt{S}_j\ ^{j\in 1..m}\} <: \{\mathtt{l}_i{:}\mathtt{T}_i\ ^{i\in 1..n}\}} \qquad \text{(S-Rcd)}$$

Simpler subtype relation

$$\mathtt{S} <: \mathtt{S} \qquad \text{(S-Refl)}$$

$$\frac{\mathtt{S} <: \mathtt{U} \qquad \mathtt{U} <: \mathtt{T}}{\mathtt{S} <: \mathtt{T}} \qquad \text{(S-Trans)}$$

$$\frac{\{\mathtt{l}_i\ ^{i\in 1..n}\} \subseteq \{\mathtt{k}_j\ ^{j\in 1..m}\} \qquad \mathtt{k}_j = \mathtt{l}_i \text{ implies } \mathtt{S}_j <: \mathtt{T}_i}{\{\mathtt{k}_j{:}\mathtt{S}_j\ ^{j\in 1..m}\} <: \{\mathtt{l}_i{:}\mathtt{T}_i\ ^{i\in 1..n}\}} \qquad \text{(S-Rcd)}$$

$$\frac{\mathtt{T}_1 <: \mathtt{S}_1 \qquad \mathtt{S}_2 <: \mathtt{T}_2}{\mathtt{S}_1{\to}\mathtt{S}_2 <: \mathtt{T}_1{\to}\mathtt{T}_2} \qquad \text{(S-Arrow)}$$

$$\mathtt{S} <: \mathtt{Top} \qquad \text{(S-Top)}$$

Step 2: Get rid of reflexivity

Observation: S-Refl is unnecessary.

Lemma: $\mathtt{S} <: \mathtt{S}$ can be derived for every type without using S-Refl.

Even simpler subtype relation

$$\frac{\mathtt{S} <: \mathtt{U} \qquad \mathtt{U} <: \mathtt{T}}{\mathtt{S} <: \mathtt{T}} \qquad \text{(S-Trans)}$$

$$\frac{\{\mathtt{l}_i\ ^{i\in 1..n}\} \subseteq \{\mathtt{k}_j\ ^{j\in 1..m}\} \qquad \mathtt{k}_j = \mathtt{l}_i \text{ implies } \mathtt{S}_j <: \mathtt{T}_i}{\{\mathtt{k}_j{:}\mathtt{S}_j\ ^{j\in 1..m}\} <: \{\mathtt{l}_i{:}\mathtt{T}_i\ ^{i\in 1..n}\}} \qquad \text{(S-Rcd)}$$

$$\frac{\mathtt{T}_1 <: \mathtt{S}_1 \qquad \mathtt{S}_2 <: \mathtt{T}_2}{\mathtt{S}_1{\to}\mathtt{S}_2 <: \mathtt{T}_1{\to}\mathtt{T}_2} \qquad \text{(S-Arrow)}$$

$$\mathtt{S} <: \mathtt{Top} \qquad \text{(S-Top)}$$

Step 3: Get rid of transitivity

Observation: S-Trans is unnecessary.

Lemma: If $\mathtt{S} <: \mathtt{T}$ can be derived, then there is a derivation that does not use S-Trans.

"Algorithmic" subtype relation

$$\mapsto S <: Top \qquad \text{(SA-Top)}$$

$$\frac{\mapsto T_1 <: S_1 \qquad \mapsto S_2 <: T_2}{\mapsto S_1 \to S_2 <: T_1 \to T_2} \qquad \text{(SA-Arrow)}$$

$$\frac{\{l_i\ ^{i \in 1..n}\} \subseteq \{k_j\ ^{j \in 1..m}\} \qquad \text{for each } k_j = l_i,\ \mapsto S_j <: T_i}{\mapsto \{k_j{:}S_j\ ^{j \in 1..m}\} <: \{l_i{:}T_i\ ^{i \in 1..n}\}} \qquad \text{(SA-Rcd)}$$

Soundness and completeness

Theorem: $S <: T$ iff $\mapsto S <: T$.

Proof: ...

Terminology:

- The algorithmic presentation of subtyping is sound with respect to the original if $\mapsto S <: T$ implies $S <: T$.
  (Everything validated by the algorithm is actually true.)
- The algorithmic presentation of subtyping is complete with respect to the original if $S <: T$ implies $\mapsto S <: T$.
  (Everything true is validated by the algorithm.)

Subtyping Algorithm (pseudo-code)

The algorithmic rules can be translated directly into code:

subtype(S,T) = if $T = Top$, then true
  else if $S = S_1 \to S_2$ and $T = T_1 \to T_2$
    then subtype(T_1, S_1) $\wedge$ subtype(S_2, T_2)
  else if $S = \{k_j{:}S_j\ ^{j \in 1..m}\}$ and $T = \{l_i{:}T_i\ ^{i \in 1..n}\}$
    then $\{l_i\ ^{i \in 1..n}\} \subseteq \{k_j\ ^{j \in 1..m}\}$
      $\wedge$ for all i there is some $j \in 1..m$ with $k_j = l_i$
        and subtype(S_j, T_i)
  else false.

Decision Procedures

A decision procedure for a relation $R \subseteq U$ is a total function p from U to $\{true, false\}$ such that $p(u) = true$ iff $u \in R$.

Decision Procedures

A decision procedure for a relation $R \subseteq \mathcal{U}$ is a total function p from $\mathcal{U}$ to $\{true, false\}$ such that $p(u) = true$ iff $u \in R$.

Is our subtype function a decision procedure?

Decision Procedures

A decision procedure for a relation $R \subseteq \mathcal{U}$ is a total function p from $\mathcal{U}$ to $\{true, false\}$ such that $p(u) = true$ iff $u \in R$.

Is our subtype function a decision procedure?

Since subtype is just an implementation of the algorithmic subtyping rules, we have

1. if $subtype(S, T) = true$, then $\mapsto S <: T$
   (hence, by soundness of the algorithmic rules, $S <: T$)
2. if $subtype(S, T) = false$, then not $\mapsto S <: T$
   (hence, by completeness of the algorithmic rules, not $S <: T$)

Decision Procedures

A decision procedure for a relation $R \subseteq \mathcal{U}$ is a total function p from $\mathcal{U}$ to $\{true, false\}$ such that $p(u) = true$ iff $u \in R$.

Is our subtype function a decision procedure?

Since subtype is just an implementation of the algorithmic subtyping rules, we have

1. if $subtype(S, T) = true$, then $\mapsto S <: T$
   (hence, by soundness of the algorithmic rules, $S <: T$)
2. if $subtype(S, T) = false$, then not $\mapsto S <: T$
   (hence, by completeness of the algorithmic rules, not $S <: T$)

Q: What's missing?

Decision Procedures

A decision procedure for a relation $R \subseteq \mathcal{U}$ is a total function p from $\mathcal{U}$ to $\{true, false\}$ such that $p(u) = true$ iff $u \in R$.

Is our subtype function a decision procedure?

Since subtype is just an implementation of the algorithmic subtyping rules, we have

1. if $subtype(S, T) = true$, then $\mapsto S <: T$
   (hence, by soundness of the algorithmic rules, $S <: T$)
2. if $subtype(S, T) = false$, then not $\mapsto S <: T$
   (hence, by completeness of the algorithmic rules, not $S <: T$)

Q: What's missing?

A: How do we know that subtype is a total function?

Decision Procedures

A decision procedure for a relation $R \subseteq U$ is a total function p from U to $\{true, false\}$ such that $p(u) = true$ iff $u \in R$.

Is our subtype function a decision procedure?

Since subtype is just an implementation of the algorithmic subtyping rules, we have

1. if $\mathsf{subtype}(S, T) = \mathsf{true}$, then $\mapsto S <: T$

   (hence, by soundness of the algorithmic rules, $S <: T$)
2. if $\mathsf{subtype}(S, T) = \mathsf{false}$, then not $\mapsto S <: T$

   (hence, by completeness of the algorithmic rules, not $S <: T$)

Q: What's missing?

A: How do we know that subtype is a total function?

Prove it!

Metatheory of Typing

Issue

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

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

Where is this rule really needed?

Issue

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

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

Where is this rule really needed?

For applications. E.g., the term

`(λr:{x:Nat}. r.x) {x=0,y=1}`

is not typable without using subsumption.

Issue

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

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

Where is this rule really needed?

For applications. E.g., the term

`(λr:{x:Nat}. r.x) {x=0,y=1}`

is *not* typable without using subsumption.

Where else??

Issue

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

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

Where is this rule really needed?

For applications. E.g., the term

`(λr:{x:Nat}. r.x) {x=0,y=1}`

is *not* typable without using subsumption.

Where else??

Nowhere else! Uses of subsumption to help typecheck applications are the *only* interesting ones.

Example

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

Example

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

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

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

Example

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

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

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

Example

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

becomes

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

Example

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

Example

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

becomes

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