CIS 500

Software Foundations

Fall 2003

19 November (continued)

Meets and Joins

Adding Booleans

Suppose we want to add booleans and conditionals to the language we have been discussing.

For the declarative presentation of the system, we just add in the appropriate syntactic forms, evaluation rules, and typing rules.

$$\Gamma \vdash \texttt{true} : \texttt{Bool} \qquad \text{(T-TRUE)}$$

$$\Gamma \vdash \texttt{false} : \texttt{Bool} \qquad \text{(T-FALSE)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{Bool} \qquad \Gamma \vdash \texttt{t}_2 : \texttt{T} \qquad \Gamma \vdash \texttt{t}_3 : \texttt{T}}{\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{T}} \qquad \text{(T-IF)}$$

A Problem with Conditional Expressions

For the algorithmic presentation of the system, however, we encounter a little difficulty.

What is the minimal type of

```
if true then {x=true,y=false} else {x=true,z=true}
```

?

The Algorithmic Conditional Rule

More generally, we can use subsumption to give an expression

`if t₁ then t₂ else t₃`

any type that is a possible type of both t_2 and t_3.

So the minimal type of the conditional is the least common supertype (or join) of the minimal type of t_2 and the minimal type of t_3.

$$\frac{\Gamma \mapsto t_1 : \mathtt{Bool} \qquad \Gamma \mapsto t_2 : T_2 \qquad \Gamma \mapsto t_3 : T_3}{\Gamma \mapsto \mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 : T_2 \vee T_3} \quad \text{(T-IF)}$$

The Algorithmic Conditional Rule

More generally, we can use subsumption to give an expression

`if t₁ then t₂ else t₃`

any type that is a possible type of both t_2 and t_3.

So the minimal type of the conditional is the least common supertype (or join) of the minimal type of t_2 and the minimal type of t_3.

$$\frac{\Gamma \mapsto t_1 : \mathtt{Bool} \qquad \Gamma \mapsto t_2 : T_2 \qquad \Gamma \mapsto t_3 : T_3}{\Gamma \mapsto \mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 : T_2 \vee T_3} \quad \text{(T-IF)}$$

Does such a type exist for every T_2 and T_3??

Existence of Joins

Theorem: For every pair of types S and T, there is a type J such that

1. S <: J
2. T <: J
3. If K is a type such that S <: K and T <: K, then J <: K.

I.e., J is the smallest type that is a supertype of both S and T.

Examples

What are the joins of the following pairs of types?

1. `{x:Bool,y:Bool}` and `{y:Bool,z:Bool}`?
2. `{x:Bool}` and `{y:Bool}`?
3. `{x:{a:Bool,b:Bool}}` and `{x:{b:Bool,c:Bool}, y:Bool}`?
4. `{}` and `Bool`?
5. `{x:{}}` and `{x:Bool}`?
6. `Top→{x:Bool}` and `Top→{y:Bool}`?
7. `{x:Bool}→Top` and `{y:Bool}→Top`?

Meets

To calculate joins of arrow types, we also need to be able to calculate meets (greatest lower bounds)!

Unlike joins, meets do not necessarily exist.

E.g., `Bool→Bool` and `{}` have no common subtypes, so they certainly don't have a greatest one!

However...

Existence of Meets

Theorem: For every pair of types `S` and `T`, if there is any type `N` such that `N <: S` and `N <: T`, then there is a type `M` such that

1. `M <: S`
2. `M <: T`
3. If `O` is a type such that `O <: S` and `O <: T`, then `O <: M`.

I.e., `M` (when it exists) is the largest type that is a subtype of both `S` and `T`.

Jargon: In the simply typed lambda calculus with subtyping, records, and booleans...

- The subtype relationhas joins
- The subtype relation has <u>bounded</u> meets

Examples

What are the meets of the following pairs of types?

1. `{x:Bool,y:Bool}` and `{y:Bool,z:Bool}`?
2. `{x:Bool}` and `{y:Bool}`?
3. `{x:{a:Bool,b:Bool}}` and `{x:{b:Bool,c:Bool}, y:Bool}`?
4. `{}` and `Bool`?
5. `{x:{}}` and `{x:Bool}`?
6. `Top→{x:Bool}` and `Top→{y:Bool}`?
7. `{x:Bool}→Top` and `{y:Bool}→Top`?

Calculating Joins

$$
\mathtt{S} \vee \mathtt{T} \;=\; \begin{cases}
\mathtt{Bool} & \text{if } \mathtt{S} = \mathtt{T} = \mathtt{Bool} \\
\mathtt{M_1}\to\mathtt{J_2} & \text{if } \mathtt{S} = \mathtt{S_1}\to\mathtt{S_2} \quad \mathtt{T} = \mathtt{T_1}\to\mathtt{T_2} \\
 & \quad \mathtt{S_1} \wedge \mathtt{T_1} = \mathtt{M_1} \quad \mathtt{S_2} \vee \mathtt{T_2} = \mathtt{J_2} \\
\{\mathtt{j_l}\!:\!\mathtt{J_l}\ ^{l\in 1..q}\} & \text{if } \mathtt{S} = \{\mathtt{k_j}\!:\!\mathtt{S_j}\ ^{j\in 1..m}\} \\
 & \quad \mathtt{T} = \{\mathtt{l_i}\!:\!\mathtt{T_i}\ ^{i\in 1..n}\} \\
 & \quad \{\mathtt{j_l}\ ^{l\in 1..q}\} = \{\mathtt{k_j}\ ^{j\in 1..m}\} \cap \{\mathtt{l_i}\ ^{i\in 1..n}\} \\
 & \quad \mathtt{S_j} \vee \mathtt{T_i} = \mathtt{J_l} \quad \text{for each } \mathtt{j_l} = \mathtt{k_j} = \mathtt{l_i} \\
\mathtt{Top} & \text{otherwise}
\end{cases}
$$

Calculating Meets

$$
\mathtt{S} \wedge \mathtt{T} \;=\; \begin{cases}
\mathtt{S} & \text{if } \mathtt{T} = \mathtt{Top} \\
\mathtt{T} & \text{if } \mathtt{S} = \mathtt{Top} \\
\mathtt{Bool} & \text{if } \mathtt{S} = \mathtt{T} = \mathtt{Bool} \\
\mathtt{J_1 \to M_2} & \text{if } \mathtt{S} = \mathtt{S_1 \to S_2} \quad \mathtt{T} = \mathtt{T_1 \to T_2} \\
 & \quad \mathtt{S_1} \vee \mathtt{T_1} = \mathtt{J_1} \quad \mathtt{S_2} \wedge \mathtt{T_2} = \mathtt{M_2} \\
\{\mathtt{m_l} : \mathtt{M_l}\ ^{l \in 1..q}\} & \text{if } \mathtt{S} = \{\mathtt{k_j} : \mathtt{S_j}\ ^{j \in 1..m}\} \\
 & \quad \mathtt{T} = \{\mathtt{l_i} : \mathtt{T_i}\ ^{i \in 1..n}\} \\
 & \quad \{\mathtt{m_l}\ ^{l \in 1..q}\} = \{\mathtt{k_j}\ ^{j \in 1..m}\} \cup \{\mathtt{l_i}\ ^{i \in 1..n}\} \\
 & \quad \mathtt{S_j} \wedge \mathtt{T_i} = \mathtt{M_l} \qquad \text{for each } \mathtt{m_l} = \mathtt{k_j} = \mathtt{l_i} \\
 & \quad \mathtt{M_l} = \mathtt{S_j} \qquad \text{if } \mathtt{m_l} = \mathtt{k_j} \textit{ occurs only in } \mathtt{S} \\
 & \quad \mathtt{M_l} = \mathtt{T_i} \qquad \text{if } \mathtt{m_l} = \mathtt{l_i} \textit{ occurs only in } \mathtt{T} \\
\mathit{fail} & \textit{otherwise}
\end{cases}
$$