

Introduction to Subtyping

Lecture 13

CS 565

3/23/09

Polymorphism

- Different varieties of polymorphism:
 - Parametric (ML)
 - type variables are abstract, and used to encode the fact that the same term can be used in many different contexts.
 - Subtype (OO)
 - a single term may have many types using the rule of subsumption that allows the type system to selectively “forget” information about the term’s behavior (and type)
 - Ad-hoc (overloading)
 - A polymorphic value exhibits different behaviors when “viewed” at different types. Overloading permits a single function symbol to be associated with many implementations.
 - Intensional polymorphism allows computation over types at runtime. (e.g., typecase or instanceof)
-

Motivation

Consider the rule for typing applications:

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash (e_1 e_2) : \tau_2}$$

This rule is very rigid. For example, the term:

$$(\lambda r:\{x:\text{Nat}\}. r.x) \{x = 0, y=1\}$$

is not well-typed. The type rule is too restrictive since we're simply passing an argument more precise than necessary.

Subsumption

- Some types are “better” than others in the sense that the value of one can always be safely used where a value of the other is expected.
 - Formalize this notion:
 - a subtyping relation between types: $\sigma <: \tau$
 - a subsumption relation that allows us to “forget” unnecessary information recorded in a type at the context where it is used.
-

Subsumption

$$\Gamma \vdash e : \sigma, \sigma <: \tau$$

$$\Gamma \vdash e : \tau$$

This rule tells us that if $\sigma <: \tau$, every element $v \in \sigma$ is also an element of τ .

Thus, if our subtype relation defined $\{x: \text{Nat}, y: \text{Bool}\} <: \{x: \text{Nat}\}$, then the subsumption rule would allow us to derive $\{x=0, y=\text{true}\} : \{x: \text{Nat}\}$

A Subtype Relation

Intuition: $\sigma <: \tau$ if an element of σ may be safely used wherever an element of τ is expected.

- σ is "better" than τ
- σ is a subset of τ
- σ is more informative/richer than τ .

$\sigma <: \sigma$ (Reflexive)

$\sigma <: \psi \quad \psi <: \tau$

 $\sigma <: \tau$ (Transitive)

We now consider subtyping for each form of type (functions, records, etc.)

Records

- “Width” subtyping:
 - forget fields “on the right”
 - $\{l_i : \tau_i \mid i \in 1 \dots n+k\} <: \{l_i : \tau_i \mid i \in 1 \dots n\}$
 - Intuition: $\{x:\text{Nat}\}$ is the type of all records that have at least one field numeric field x .
 - A record type with more fields is a subtype of a record with fewer fields.
 - The more fields a record has, the more constraints it imposes on its uses, and thus describes fewer values.
-

Records

Depth subtyping:

for each i $\sigma_i <: \tau_i$

$$\{ |i::\sigma_i^i \in 1\dots n \} <: \{ |i::\tau_i^i \in 1\dots n \}$$

- types of individual fields can vary provided all types in the record satisfy the subtype relation.
-

Example

To show that

$$\{x: \{a:\text{Nat}, b:\text{Nat}\}, y:\{m:\text{Nat}\}\} <: \{x:\{a:\text{Nat}\}, y:\{\}\}$$

$$\{a:\text{Nat}, b:\text{Nat}\} <: \{a:\text{Nat}\}$$

$$\{m:\text{Nat}\} <: \{\}$$

$$\{x: \{a:\text{Nat}, b:\text{Nat}\}, y:\{m:\text{Nat}\}\} <: \{x:\{a:\text{Nat}\}, y:\{\}\}$$

Where is width subtyping used? Where is depth subtyping used?

Example

To show that

$$\{x: \{a:\text{Nat}, b:\text{Nat}\}, y:\{m:\text{Nat}\}\} <: \{x:\{a:\text{Nat}\}, y:\{m:\text{Nat}\}\}$$

$$\{a:\text{Nat}, b:\text{Nat}\} <: \{a:\text{Nat}\} \quad \{m:\text{Nat}\} <: \{m:\text{Nat}\}$$

$$\{x: \{a:\text{Nat}, b:\text{Nat}\}, y:\{m:\text{Nat}\}\} <: \{x:\{a:\text{Nat}\}, y:\{m:\text{Nat}\}\}$$

Uses reflexive property of subtype relation.

Example

To show that

$$\{x: \{a:\text{Nat}, b:\text{Nat}\}, y:\{m:\text{Nat}\}\} <: \{x:\{a:\text{Nat}\}\}$$

<hr/>	<hr/>
$\{x:\{a:\text{Nat}, b:\text{Nat}\}, y:\{m:\text{Nat}\}\} <: \{x:\{a:\text{Nat}, b:\text{Nat}\}\}$	$\{a:\text{Nat}, b:\text{Nat}\} <: \{a:\text{Nat}\}$
<hr/>	<hr/>
	$\{x:\{a:\text{Nat}, b:\text{Nat}\}\} <: \{x:\{a:\text{Nat}\}\}$

$$\{x: \{a:\text{Nat}, b:\text{Nat}\}, y:\{m:\text{Nat}\}\} <: \{x:\{a:\text{Nat}\}\}$$

Uses transitive property of subtype relation.

Records

The permutation subtyping rule allows us to ignore the order of fields in records.

$\{k_j : \sigma_j^j \in 1 \dots n_j\}$ is a permutation of $\{l_i : \tau_i^i \in 1 \dots n_i\}$

$\{k_j : \sigma_j^j \in 1 \dots n_j\} <: \{l_i : \tau_i^i \in 1 \dots n_i\}$

Why is this safe?

Example: $\{c: \text{Bool}, d: \text{Nat}\} <: \{d: \text{Nat}, c: \text{Bool}\}$

Is the subtype relation anti-symmetric in the presence of permutations?

Variations

Not all languages adopt all these features.

Example:

In Java,

- A subclass may not change the argument or result types of a method of its superclass
 - no depth subtyping
 - Each class has just one superclass and each class member can be assigned a single index, adding new indices on the right
 - no permutation rule
 - permutations allowed only when considering interfaces
-

Arrow Types

$$\frac{\tau_1 \prec \sigma_1 \quad \sigma_2 \prec \tau_2}{\sigma_1 \rightarrow \sigma_2 \prec \tau_1 \rightarrow \tau_2}$$

The subtype relation is contravariant in the type of the argument, and covariant in the type of the result.

Intuition: if we have a function f of type $\sigma_1 \rightarrow \sigma_2$, then we know f accepts elements of any subtype $\tau_1 \prec \sigma_1$. Since f returns elements of type σ_2 , these results belong to any supertype τ_2 of σ_2 .

Arrow Types

It is safe to allow a function of type $\sigma_1 \rightarrow \sigma_2$ to be used in a context $\tau_1 \rightarrow \tau_2$ provided none of the arguments supplied will “surprise” it ($\tau_1 \prec \sigma_1$) and none of its results will “surprise” the context in which it is used ($\sigma_2 \prec \tau_2$)

Examples

- $b \rightarrow b \Leftarrow a \rightarrow b$
- $a \rightarrow a \Leftarrow a \rightarrow b$
- $(a \rightarrow b) \rightarrow a \Leftarrow (b \rightarrow a) \rightarrow b$

assuming $a \Leftarrow b$

Top

It is convenient to have a type that is the supertype of every type. Introduce a new type constant `Top` plus a rule that makes `Top` a maximal element in the subtype relation:

$$\sigma \prec: \text{Top}$$

Example revisited

$\lambda r: \{x:\text{Nat}\}. r.x) \{x = 0, y=1\}$

$f \equiv \lambda r: \{x:\text{Nat}\}. r.x$ $xy \equiv \{x = 0, y = 1\}$

$Rx \equiv \{x:\text{Nat}\}$ $Rxy \equiv \{x:\text{Nat}, y:\text{Nat}\}$

$$\frac{\frac{\frac{\frac{}{\vdash 0 : \text{Nat}}{} \quad \frac{}{\vdash 1 : \text{Nat}}{} }{\vdash xy : Rxy}}{\vdash f : Rx \rightarrow \text{Nat}} \quad \frac{}{Rxy <: Rx}}{\vdash f xy : \text{Nat}}}$$
