

CS 565

Programming Languages (graduate) Spring 2025

Week 9
Subtyping

Inclusion Polymorphism

2

- Polymorphism = allowing a single definition to be used with different types in different contexts.
 - Parametric polymorphism: behavior is the same for all instantiations
 - Ad-hoc polymorphism: behavior is based on the type of its arguments (overloading)
- One of the defining characteristics of Object-Oriented languages is their support for inclusion polymorphism
 - Inclusion polymorphism allows for code that needs *at least* a T in a context

A Calculus for Subtyping

3

- ★ Begin with a simple language for studying subtyping
- ★ Key extension is Records (labeled products)

$t ::= x \mid \lambda x:T.t \mid tt \mid n \in \mathbb{N} \mid i \in \mathbb{R} \mid t + t \mid \dots$

$\mid \langle l_1=t_1, \dots, l_n=t_n \rangle \Leftarrow \text{Records}$

$\mid t.l \Leftarrow \text{Projection}$

$v ::= \lambda x:T.t \mid n \in \mathbb{N} \mid i \in \mathbb{R} \mid \langle l_1=v_1, \dots, l_n=v_n \rangle$
can be empty $\langle \rangle$

$T ::= T \rightarrow T \mid \mathbb{R} \mid \mathbb{N}$

$\mid \langle l_1:T_1, \dots, l_n:T_n \rangle \Leftarrow \text{Record Types}$

$\mid \text{Top}$

A Calculus for Subtyping

4

- Begin with a simple language for studying subtyping
- Key extension is Records (labeled products)

$t ::= x \mid \lambda x:T.t \mid t \ t \mid \dots \mid \langle l_1=t_1, \dots, l_n=t_n \rangle \Leftarrow \text{Records}$

$\mid t.l \Leftarrow \text{Projection}$

$T ::= T \rightarrow T \mid \mathbb{R} \mid \dots \mid \langle l_1:T_1, \dots, l_n:T_n \rangle \Leftarrow \text{Record Types}$

$\text{Point} \equiv \langle x:\mathbb{R}, y:\mathbb{R} \rangle$

$\text{Dist} \equiv \lambda p:\text{Point}.\sqrt{p.x^2 + p.y^2}$

$\text{Dist } \langle x:2, y:2 \rangle \longrightarrow^* 2.82\dots$

$\text{ColorPoint} \equiv \langle x:\mathbb{R}, y:\mathbb{R}, R:\mathbb{N}, G:\mathbb{N}, B:\mathbb{N} \rangle$

Subsumption

5

★ Would like this to typecheck:

$\text{Dist } \langle x=2, y=2, R=0, G=140, B=255 \rangle$

★ Let's extend existing type system!

★ Key Idea: the subsumption rule:

$$\vdash \text{Dist} : \text{Point} \rightarrow \mathbb{R}$$
$$\frac{\Gamma \vdash t_1 : T_1 \quad T_1 <: T_2}{\Gamma \vdash t_1 : T_2} \text{TSUB}$$
$$\Gamma \vdash \langle x=2, y=2, R=0, G=140, B=255 \rangle : \text{ColorPoint}$$
$$\text{ColorPoint} <: \text{Point}$$
$$\Gamma \vdash \langle x=2, y=2, R=0, G=140, B=255 \rangle : \text{Point}$$
$$\vdash \text{Dist} : \text{Point} \rightarrow \mathbb{R}$$
$$\vdash \text{Dist } \langle x=2, y=2, R=0, G=140, B=255 \rangle : \mathbb{R}$$

Subsumption

6

Would like this to typecheck:

Dist $\langle x=2, y=2, R=0, G=140, B=255 \rangle$

$$\frac{\Gamma \vdash t_1 : T_1 \quad T_1 <: T_2}{\Gamma \vdash t_1 : T_2} \text{TSUB}$$

How to define $T_1 <: T_2$?

Substitutability: If $T_1 <: T_2$, then any value of type T_1 must be usable in every way a T_2 is.

The difficulty is ensuring this is safe (i.e. doesn't break type safety)!

Subtype Relation

7

What key properties should a substitutability relation satisfy?

- Should be transitive:

$$\frac{S <: T \quad T <: U}{S <: U} \quad \text{STRANS}$$

- And reflexive (not strictly necessary, but makes theory nicer)

$$\frac{}{S <: S} \quad \text{SREFL}$$

- How to relate concrete types?

$$\frac{}{\mathbb{N} <: \mathbb{R}} \quad \text{SNUM}$$

Subtyping Records

8

What sorts of pairs of record types satisfy substitutability?

- A subtype should be able to add fields at the end:

$$\langle x:\mathbb{R}, y:\mathbb{R}, R:\mathbb{N} \rangle <: \langle x:\mathbb{R}, y:\mathbb{R} \rangle$$

$$\langle l_1:T_1, \dots, l_n:T_n, \dots, l_{n+k}:T_{n+k} \rangle <: \langle l_1:T_1, \dots, l_n:T_n \rangle$$

SBWIDTH

- A subtype should be able to reorder fields:

$$\langle y:\mathbb{R}, x:\mathbb{R} \rangle <: \langle x:\mathbb{R}, y:\mathbb{R} \rangle$$

$$\langle k_1:T_1, \dots, k_n:T_n \rangle \text{ is a permutation of } \langle l_1:T_1, \dots, l_n:T_n \rangle$$

$$\langle l_1:T_1, \dots, l_n:T_n \rangle <: \langle k_1:T_1, \dots, k_n:T_n \rangle$$

SBPERM

Subtyping Records

9

What sorts of pairs of record types satisfy substitutability?

A subtype should be able to add fields at the end:

$$\frac{}{\langle l_1:T_1, \dots, l_n:T_n, \dots, l_{n+k}:T_{n+k} \rangle <: \langle l_1:T_1, \dots, l_n:T_n \rangle}$$

SBWIDTH

A subtype should be able to reorder fields:

$$\frac{\langle k_1:T_1, \dots, k_n:T_n \rangle \text{ is a permutation of } \langle l_1:T_1, \dots, l_n:T_n \rangle}{\langle l_1:T_1, \dots, l_n:T_n \rangle <: \langle k_1:T_1, \dots, k_n:T_n \rangle}$$

SBPERM

A subtype should be able to specialize field types:

$$\langle x:\mathbb{R}, y:\mathbb{R}, R:\mathbb{R} \rangle <: \langle x:\mathbb{R}, y:\mathbb{R}, R:\mathbb{N} \rangle$$

$$\frac{}{T_i <: S_i}$$

$$\frac{}{\langle l_1:T_1, \dots, l_n:T_n \rangle <: \langle l_1:S_1, \dots, l_n:S_n \rangle}$$

SBDEPTH

Subtyping Functions

10

★ Subtyping helps with our function application problem!

$$\begin{array}{c} \vdash \text{Dist } \langle x=2, y=2, R=0, G=140, B=255 \rangle : \mathbb{R} \quad \text{SUBWIDTH} \\ \hline \Gamma \vdash \langle x=2, y=2, R=0, G=140, B=255 \rangle : \text{ColorPoint} \quad \text{ColorPoint} <: \text{Point} \\ \hline \Gamma \vdash \langle x=2, y=2, R=0, G=140, B=255 \rangle : \text{Point} \quad \text{TSUB} \end{array}$$

Subtyping Functions

11

- Subtyping helps with our function application problem* ...
- What about this expression:

$$\text{DistMoved} \equiv \lambda t:\text{Point} \rightarrow \text{Point}. p:\text{Point}.$$
$$|\text{Dist } (t \text{ } p) - \text{Dist } p|$$
$$\text{Flip} \equiv \lambda p:\text{Point}. \langle x = -(p.x), y = -(p.y) \rangle$$
$$\vdash \text{DistMoved} : (\text{Point} \rightarrow \text{Point}) \rightarrow \text{Point} \rightarrow \mathbb{R}$$
$$\vdash \text{DistMoved Flip } \langle x=2, y=2 \rangle : \mathbb{R}$$
$$\text{DistMoved Flip } \langle x=2, y=2 \rangle \longrightarrow^* 0$$

Subtyping Functions

12

- Subtyping helps with our function application problem* ...
- What about this expression:

$\text{DistMoved} \equiv \lambda t:\text{Point} \rightarrow \text{Point}. p:\text{Point}.$

$\text{IDist } (t \text{ } p) - \text{Dist } p \text{ }$

$\text{FlipGreen} \equiv \lambda p:\text{Point}.$

$\langle x=-(p.x), y=-(p.y), R=0, G=255, B=0 \rangle$

$\vdash \text{DistMoved} : (\text{Point} \rightarrow \text{Point}) \rightarrow \text{Point} \rightarrow \mathbb{R}$

$\vdash \text{FlipGreen} : (\text{Point} \rightarrow \text{ColorPoint})$

$\not\vdash \text{DistMoved } \text{FlipGreen} \langle x=2, y=2 \rangle$

$\text{DistMoved } \text{FlipGreen} \langle x=2, y=2 \rangle \longrightarrow^* 0$

Subtyping Functions

13

- Subtyping helps with our function application problem* ...
- What about this expression:

$\text{DistMoved} \equiv \lambda t:\text{Point} \rightarrow \text{Point}. p:\text{Point}.$

$\quad | \text{Dist } (t \text{ } p) - \text{Dist } p |$

$\text{FlipGreen} \equiv \lambda p:\text{Point}.$

$\quad \langle x=-(p.x), y=-(p.y), R=0, G=255, B=0 \rangle$

$$\frac{S_2 <: T_2}{T_1 \rightarrow S_2 <: T_1 \rightarrow T_2} \quad \text{SB-ARROW}_2$$

$\text{DistMoved FlipGreen } \langle x=2, y=2 \rangle$

$\vdash \text{DistMoved FlipGreen } \langle x=2, y=2 \rangle : \mathbb{R}$

Subtyping Functions

14

- Subtyping helps with our function application problem* ...
- What about this expression:

$\text{DistMoved} \equiv \lambda t:\text{Point} \rightarrow \text{Point}. p:\text{Point}.$
 $\quad | \text{Dist } (t \text{ } p) - \text{Dist } p |$

$\text{FlipGreener} \equiv \lambda p:\text{ColorPoint}.$
 $\quad \langle x=-(p.x), y=-(p.y), R=p.R, G=255, B=p.B \rangle$

$\text{DistMoved } \text{FlipGreener } \langle x=2, y=2 \rangle$

Subtyping Functions

15

- Subtyping helps with our function application problem*...
- What about this expression:

$\text{DistMoved} \equiv \lambda t:\text{Point} \rightarrow \text{Point}. p:\text{Point}.$

$\text{IDist } (t \text{ } p) - \text{Dist } p \text{ }$

$\text{FlipGreener} \equiv \lambda p:\text{ColorPoint}.$

$\langle x=-(p.x), y=-(p.y), R=p.R, G=255, B=p.B \rangle$

$$\frac{T_1 <: S_1 \quad S_2 <: T_2}{S_1 \rightarrow S_2 <: T_1 \rightarrow T_2} \quad \text{SB-ARROW}$$

✗ $\text{DistMoved } \text{FlipGreener } \langle x=2, y=2 \rangle$

Subtyping Functions

16

- Subtyping helps with our function application problem*!
- What about this expression:

$\text{DistMovedC} \equiv \lambda t:\text{ColorPoint} \rightarrow \text{Point}. p:\text{ColorPoint}.$

$\text{IDist } (t \ p) - \text{Dist } p \mid$

$\text{FlipGreener} \equiv \lambda p:\text{ColorPoint}.$

$\langle x=-(p.x), y=-(p.y), R=p.R, G=255, B=p.B \rangle$

$$\frac{T_1 <: S_1 \quad S_2 <: T_2}{S_1 \rightarrow S_2 <: T_1 \rightarrow T_2} \quad \text{SB-ARROW}$$

$\text{DistMovedC Flip } \langle x=2, y=2, R=0, G=0, B=0 \rangle$

$\text{DistMovedC FlipGreen } \langle x=2, y=2, R=0, G=0, B=0 \rangle$

$\text{DistMovedC FlipGreener } \langle x=2, y=2, R=0, G=0, B=0 \rangle$

Variance

17

Variance is a property on the arguments of type constructors like function types $(A \rightarrow B)$, tuples $(A \times B)$, and record types

$F(A)$ is **covariant** over A if $A <: A'$ implies that $F(A) <: F(A')$

$F(B)$ is **contravariant** over B if $B' <: B$ implies that $F(B) <: F(B')$

$F(T)$ is **invariant** over T otherwise

$$\frac{S_1 <: T_1 \quad S_2 <: T_2}{S_1 \times S_2 <: T_1 \times T_2} \quad \text{SB-TUPLE}$$

$$\frac{T_1 <: S_1 \quad S_2 <: T_2}{S_1 \rightarrow S_2 <: T_1 \rightarrow T_2} \quad \text{SB-ARROW}$$

Classes Vs. Types

18

- A class provides implementations of an object's behavior
Subclassing inherits behavior and changes it via extension and overriding
- A class also defines an interface (type) for its object's type
A **subtype** is substitutable in terms of its field/method types

Inheritance Vs. Subtyping

19

- Consider three datatypes:
 - Queues:** FIFO insertion and deletion
 - Stacks:** LIFO insertion and deletion
 - Dequeues:** insert/delete from rear/front
- A **dequeue** implementation provides both a stack and a queue:
 - Stack:** only expose `insert_front` and `delete_front` operations
 - Queue:** only expose `insert_rear` and `delete_front` operations

Neither a **Stack** nor a **Queue** is a **Dequeue** though!

Inheritance Vs. Subtyping

20

- These are separate concepts!
 - Java confuses them by making subclasses be subtypes
 - Inheritance is a mechanism for code reuse
 - Subtyping is a mechanism for checking substitutability