Type Systems

Lecture 5 Nov. 17th, 2004
Sebastian Maneth

http://lampwww.epfl.ch/teaching/typeSystems/2004

Today:

1. Subtyping (functions and records)
2. Algorithmic Subtyping
3. Joins and Meets

1. Subtyping (functions and records)

```
(λr:{x:Nat}. r.x) {x=0,y=1}
```

1. Subtyping (functions and records)

```
(λr:{x:Nat}. r.x) {x=0,y=1}
```

ill-typed! — this function can ONLY be applied to records of the type {x:Nat}

→ the type system is WAY too strict! too much slack!

Actually, the function can be applied to

ANY record that has **at least the field x:Nat!**

1. Subtyping (functions and records)

```
(λr:{x:Nat}. r.x) {x=0,y=1}
```

ill-typed! — this function can ONLY be applied to records of the type {x:Nat}

→ the type system is WAY too strict! too much slack!

Actually, the function can be applied to

ANY record that has **at least the field x:Nat!**

a **subtype** of {x:Nat}

Why "**sub**"?

{x:Nat,y:Nat} is a subtype, written **<:** of {x:Nat} because

# of records having this is LESS than # of records having this

1. Subtyping (functions and records)

{x:Nat,y:Nat} **<:** {x:Nat} ("is subtype of")

1. Subtyping (functions and records)

{x:Nat,y:Nat} <: {x:Nat} ("is subytpe of")

S T

→ If t satisfies S in some context, then it also satisfies T!!

1. Subtyping (functions and records)

{x:Nat,y:Nat} <: {x:Nat} ("is subytpe of")

S T

→ If t satisfies S in some context, then it also satisfies T!!

is of type … is of type

Rule of subsumtion: $\dfrac{\Gamma \vdash t : S \qquad S <: T}{\Gamma \vdash t : T}$

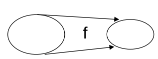
… = "move to a more general type (supertype)"

1. Subtyping (functions and records)

Rules for the Subtype Relation <:

reflexive, transitive: $S <: S$ $\qquad \dfrac{S <: U \qquad U <: T}{S <: T}$

"top type": $S <:$ Top



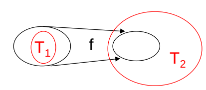
function $f : S_1 \rightarrow S_2$

1. Subtyping (functions and records)

Rules for the Subtype Relation <:

reflexive, transitive: $S <: S$ $\qquad \dfrac{S <: U \qquad U <: T}{S <: T}$

"top type": $S <:$ Top



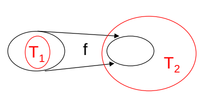
function $f : S_1 \rightarrow S_2$

1. Subtyping (functions and records)

Rules for the Subtype Relation <:

reflexive, transitive: $S <: S$ $\qquad \dfrac{S <: U \qquad U <: T}{S <: T}$

"top type": $S <:$ Top



function $f : S_1 \rightarrow S_2$

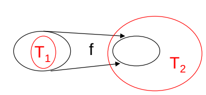
$$\frac{T_1 <: S_1 \qquad S_2 <: T_2}{S_1 \rightarrow S_2 \;<:\; T_1 \rightarrow T_2}$$

1. Subtyping (functions and records)

Rules for the Subtype Relation <:

reflexive, transitive: $S <: S$ $\qquad \dfrac{S <: U \qquad U <: T}{S <: T}$

"top type": $S <:$ Top



function $f : S_1 \rightarrow S_2$

contravariant in argument → $\dfrac{T_1 <: S_1 \qquad S_2 <: T_2}{S_1 \rightarrow S_2 \;<:\; T_1 \rightarrow T_2}$ ← covariant in result

1. Subtyping (functions and records)

Rules for the Subtype Relation <:

reflexive, transitive: $S <: S$ $\qquad \dfrac{S <: U \quad U <: T}{S <: T}$ $\qquad S <: Top$

$$\dfrac{T_1 <: S_1 \quad S_2 <: T_2}{S_1 \to S_2 <: T_1 \to T_2}$$

records:

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

(forget fields)

$$\dfrac{\text{for each } i \quad S_i <: T_i}{\{l_1{:}S_1, \ldots, l_n{:}S_n\} <: \{l_1{:}T_1, \ldots, l_n{:}T_n\}}$$

(subtype inside of record)

$$\dfrac{\{k_1{:}S_1, \ldots, k_n{:}S_n\} \text{ is permutation of } \{l_1{:}T_1, \ldots, l_n{:}T_n\}}{\{k_1{:}S_1, \ldots, k_n{:}S_n\} <: \{l_1{:}T_1, \ldots, l_n{:}T_n\}}$$

(don't care about order)

1. Subtyping (functions and records)

`(λr:{x:Nat}. r.x) {x=0,y=1}` ill-typed! (in simply lambda)

Can we type this now, using subtyping?

`⊢ (λr:{x:Nat}. r.x) {x=0,y=1}`

1. Subtyping (functions and records)

`(λr:{x:Nat}. r.x) {x=0,y=1}` ill-typed! (in simply lambda)

Can we type this now, using subtyping?

$$\dfrac{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x) : \{x{:}Nat\} \to Nat \qquad \vdash \{x{=}0, y{=}1\} : \{x{:}Nat\}}{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x)\ \{x{=}0, y{=}1\}}$$

1. Subtyping (functions and records)

`(λr:{x:Nat}. r.x) {x=0,y=1}` ill-typed! (in simply lambda)

Can we type this now, using subtyping?

$$\dfrac{\vdash \{x{=}0, y{=}1\} : \{x{:}Nat, y{:}Nat\} \qquad \{x{:}Nat, y{:}Nat\} <: \{x{:}Nat\}}{}$$

not derivable before
NOW: use *subsumption* !

$$\dfrac{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x) : \{x{:}Nat\} \to Nat \qquad \vdash \{x{=}0, y{=}1\} : \{x{:}Nat\}}{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x)\ \{x{=}0, y{=}1\}}$$

1. Subtyping (functions and records)

`(λr:{x:Nat}. r.x) {x=0,y=1}` ill-typed! (in simply lambda)

Can we type this now, using subtyping?

OK! (forget fields rule)

$$\dfrac{\vdash \{x{=}0, y{=}1\} : \{x{:}Nat, y{:}Nat\} \qquad \dfrac{}{\{x{:}Nat, y{:}Nat\} <: \{x{:}Nat\}}}{}$$

not derivable before
NOW: use *subsumption* !

$$\dfrac{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x) : \{x{:}Nat\} \to Nat \qquad \vdash \{x{=}0, y{=}1\} : \{x{:}Nat\}}{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x)\ \{x{=}0, y{=}1\}}$$

1. Subtyping (functions and records)

`(λr:{x:Nat}. r.x) {x=0,y=1}` ill-typed! (in simply lambda)

Can we type this now, using subtyping?

OK! (forget fields rule)

$$\dfrac{\dfrac{\vdash 0 : Nat \qquad \vdash 1 : Nat}{\vdash \{x{=}0, y{=}1\} : \{x{:}Nat, y{:}Nat\}} \qquad \dfrac{}{\{x{:}Nat, y{:}Nat\} <: \{x{:}Nat\}}}{}$$

not derivable before
NOW: use *subsumption* !

$$\dfrac{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x) : \{x{:}Nat\} \to Nat \qquad \vdash \{x{=}0, y{=}1\} : \{x{:}Nat\}}{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x)\ \{x{=}0, y{=}1\}}$$

1. Subtyping (functions and records)

`(λr:{x:Nat}. r.x) {x=0,y=1}` ill-typed! (in simply lambda)

Can we type this now, using subtyping?

$$\dfrac{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x) : \{x{:}Nat\} \rightarrow Nat \qquad \dfrac{\text{OK!}}{\vdash \{x{=}0, y{=}1\} : \{x{:}Nat\}}}{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x)\ \{x{=}0, y{=}1\}}$$

1. Subtyping (functions and records)

`(λr:{x:Nat}. r.x) {x=0,y=1}` ill-typed! (in simply lambda)

Can we type this now, using subtyping?

$$\dfrac{\dfrac{r{:}\{x{:}Nat\} \vdash r.x : Nat}{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x) : \{x{:}Nat\} \rightarrow Nat} \qquad \dfrac{\text{OK!}}{\vdash \{x{=}0, y{=}1\} : \{x{:}Nat\}}}{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x)\ \{x{=}0, y{=}1\}}$$

1. Subtyping (functions and records)

`(λr:{x:Nat}. r.x) {x=0,y=1}` ill-typed! (in simply lambda)

Can we type this now, using subtyping?

$$\dfrac{\dfrac{\dfrac{r{:}\{x{:}Nat\} \vdash r : \{x{:}Nat\}}{r{:}\{x{:}Nat\} \vdash r.x : Nat}}{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x) : \{x{:}Nat\} \rightarrow Nat} \qquad \dfrac{\text{OK!}}{\vdash \{x{=}0, y{=}1\} : \{x{:}Nat\}}}{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x)\ \{x{=}0, y{=}1\}}$$

1. Subtyping (functions and records)

`(λr:{x:Nat}. r.x) {x=0,y=1}` ill-typed! (in simply lambda)

Can we type this now, using subtyping?

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{OK!}}{r{:}\{x{:}Nat\} \in r{:}\{x{:}Nat\}}}{r{:}\{x{:}Nat\} \vdash r : \{x{:}Nat\}}}{r{:}\{x{:}Nat\} \vdash r.x : Nat}}{\vdash (\lambda r{:}\{x{:}Nat\}.\ r.x) : \{x{:}Nat\} \rightarrow Nat} \qquad \dfrac{\text{OK!}}{\vdash \{x{=}0, y{=}1\} : \{x{:}Nat\}}$$

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

2. Algorithmic Subtyping

Problematic rules for implementing subtyping:

$$\dfrac{S <: U \qquad U <: T}{S <: T}$$

$$S <: S$$

$$\dfrac{\Gamma \vdash t : S \qquad S <: T}{\Gamma \vdash t : T}$$

Are NOT syntax-directed!!

→ When to apply them??

2. Algorithmic Subtyping

Problematic rules for implementing subtyping:

$$\dfrac{S <: U \qquad U <: T}{S <: T}$$

← needed to merge subtyping derivation of records

$$S <: S$$

to see this, find a derivation for

`{x:{a:Nat,b:Nat},y:{m:Nat}}` <: `{x:{a:Nat}}`

$$\dfrac{\Gamma \vdash t : S \qquad S <: T}{\Gamma \vdash t : T}$$

2. Algorithmic Subtyping

Problematic rules for implementing subtyping:

$$\frac{S <: U \qquad U <: T}{S <: T}$$ ← needed to merge subtyping derivation of records

$S <: S$

$$\frac{\Gamma \vdash t : S \qquad S <: T}{\Gamma \vdash t : T}$$

new:

$$\frac{\{k_1{:}S_1, \dots, k_n{:}S_n\} \subseteq \{l_1{:}T_1, \dots, l_m{:}T_m\} \quad k_j = l_i \text{ implies } S_j <: T_i}{\{k_1{:}S_1, \dots, k_n{:}S_n\} <: \{l_1{:}T_1, \dots, l_m{:}T_m\}}$$

2. Algorithmic Subtyping

Problematic rules for implementing subtyping:

$$\frac{S <: U \qquad U <: T}{S <: T}$$ ← needed to merge subtyping derivation of records

$S <: S$

$$\frac{\Gamma \vdash t : S \qquad S <: T}{\Gamma \vdash t : T}$$

↑ ONLY needed for fu. application!

new:

$$\frac{\{k_1{:}S_1, \dots, k_n{:}S_n\} \subseteq \{l_1{:}T_1, \dots, l_m{:}T_m\} \quad k_j = l_i \text{ implies } S_j <: T_i}{\{k_1{:}S_1, \dots, k_n{:}S_n\} <: \{l_1{:}T_1, \dots, l_m{:}T_m\}}$$

2. Algorithmic Subtyping

Problematic rules for implementing subtyping:

$$\frac{S <: U \qquad U <: T}{S <: T}$$ ← needed to merge subtyping derivation of records

$S <: S$

$$\frac{\Gamma \vdash t : S \qquad S <: T}{\Gamma \vdash t : T}$$

↑ ONLY needed for fu. application!

new:

$$\frac{\{k_1{:}S_1, \dots, k_n{:}S_n\} \subseteq \{l_1{:}T_1, \dots, l_m{:}T_m\} \quad k_j = l_i \text{ implies } S_j <: T_i}{\{k_1{:}S_1, \dots, k_n{:}S_n\} <: \{l_1{:}T_1, \dots, l_m{:}T_m\}}$$

new:

$$\frac{\Gamma \vdash t_1{:}T_1 \to T_2 \qquad \Gamma \vdash t_2 : R \qquad R <: T_1}{\Gamma \vdash t_1\ t_2 : T_2}$$

2. Algorithmic Subtyping

Implementing subtyping:

$$\frac{S <: U \qquad U <: T}{S <: T}$$

$S <: S$

$$\frac{\Gamma \vdash t : S \qquad S <: T}{\Gamma \vdash t : T}$$

$$\frac{T_1 <: S_1 \qquad S_2 <: T_2}{S_1 \to S_2 <: T_1 \to T_2}$$

$$\frac{\{k_1{:}S_1, \dots, k_n{:}S_n\} \subseteq \{l_1{:}T_1, \dots, l_m{:}T_m\} \quad k_j = l_i \text{ implies } S_j <: T_i}{\{k_1{:}S_1, \dots, k_n{:}S_n\} <: \{l_1{:}T_1, \dots, l_m{:}T_m\}}$$

$$\frac{\Gamma \vdash t_1{:}T_1 \to T_2 \qquad \Gamma \vdash t_2 : R \qquad R <: T_1}{\Gamma \vdash t_1\ t_2 : T_2}$$

2. Algorithmic Subtyping

Implementing subtyping:

$$\frac{S <: U \qquad U <: T}{S <: T}$$

$S <: S$

$$\frac{\Gamma \vdash t : S \qquad S <: T}{\Gamma \vdash t : T}$$

$$\frac{T_1 <: S_1 \qquad S_2 <: T_2}{S_1 \to S_2 <: T_1 \to T_2}$$

$$\frac{\{k_1{:}S_1, \dots, k_n{:}S_n\} \subseteq \{l_1{:}T_1, \dots, l_m{:}T_m\} \quad k_j = l_i \text{ implies } S_j <: T_i}{\{k_1{:}S_1, \dots, k_n{:}S_n\} <: \{l_1{:}T_1, \dots, l_m{:}T_m\}}$$

$$\frac{\Gamma \vdash t_1{:}T_1 \to T_2 \qquad \Gamma \vdash t_2 : R \qquad R <: T_1}{\Gamma \vdash t_1\ t_2 : T_2}$$

AND: $S <: S$ for every base type S.

2. Algorithmic Subtyping

Call the rules used for implementation "algorithmic typing" ($\Vdash$)

→ are these rules *sound* and *complete* w.r.t. the previous rules ($\vdash$)??

2. Algorithmic Subtyping

Call the rules used for implementation "algorithmic typing" ($\Vdash$)

→ are these rules *sound* and *complete* w.r.t. the previous rules ($\vdash$)??

Soundness: If $\Gamma \Vdash t : T$, then $\Gamma \vdash t : T$

YES, prove this by straightforward induction!

2. Algorithmic Subtyping

Call the rules used for implementation "algorithmic typing" ($\Vdash$)

→ are these rules *sound* and *complete* w.r.t. the previous rules ($\vdash$)??

Soundness: If $\Gamma \Vdash t : T$, then $\Gamma \vdash t : T$

YES, prove this by straightforward induction!

Completeness: If $\Gamma \vdash t : T$, then $\Gamma \Vdash t : T$

2. Algorithmic Subtyping

Call the rules used for implementation "algorithmic typing" ($\Vdash$)

→ are these rules *sound* and *complete* w.r.t. the previous rules ($\vdash$)??

Soundness: If $\Gamma \Vdash t : T$, then $\Gamma \vdash t : T$

YES, prove this by straightforward induction!

Completeness: If $\Gamma \vdash t : T$, then $\Gamma \Vdash t : T$

NO, $\vdash$ `{a=0}:{a:Nat,b:Nat}`,

but **not** true for $\Vdash$

2. Algorithmic Subtyping

Call the rules used for implementation "algorithmic typing" ($\Vdash$)

→ are these rules *sound* and *complete* w.r.t. the previous rules ($\vdash$)??

Soundness: If $\Gamma \Vdash t : T$, then $\Gamma \vdash t : T$

YES, prove this by straightforward induction!

Completeness: If $\Gamma \vdash t : T$, then $\Gamma \Vdash t : T$

2. Algorithmic Subtyping

Call the rules used for implementation "algorithmic typing" ($\Vdash$)

→ are these rules *sound* and *complete* w.r.t. the previous rules ($\vdash$)??

Soundness: If $\Gamma \Vdash t : T$, then $\Gamma \vdash t : T$

YES, prove this by straightforward induction!

Completeness: If $\Gamma \vdash t : T$, then $\Gamma \Vdash t : S$ for some $S <: T$

= "*Minimal Typing Theorem*"

→ Can you prove that S is actually **minimal**??

3. Joins and Meets

How to type if-then-else, in the presence of subsumtion?

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

What is the type of this term?

3. Joins and Meets

How to type if-then-else, in the presence of subsumtion?

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

What is the type of this term?

→ maybe `{x:Bool}`?

3. Joins and Meets

How to type if-then-else, in the presence of subsumtion?

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

What is the type of this term?

→ maybe `{x:Bool}`?

or `{x:Top}`?

3. Joins and Meets

How to type if-then-else, in the presence of subsumtion?

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

What is the type of this term?

→ maybe `{x:Bool}`?

<:

or `{x:Top}`?

<:

or `{}`?

<:

or `Top`?

3. Joins and Meets

How to type if-then-else, in the presence of subsumtion?

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

What is the type of this term?

→ `{x:Bool}` — take the *least* (most precise) common supertype of S and T

<:

or `{x:Top}`?

<:

or `{}`?

<:

or `Top`?

3. Joins and Meets

How to type if-then-else, in the presence of subsumtion?

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

What is the type of this term?

→ `{x:Bool}` — take the *least* (most precise) common supertype of S and T

<:

or `{x:Top}`?

<:

or `{}`?

<:

or `Top`?

= "the *join* of S and T"

=: $S \vee T$

3. Joins and Meets

How to type if-then-else, in the presence of subsumtion?

$$\frac{t_1 : \text{Bool} \quad t_2 : T_2 \quad t_3 : T_3 \quad T = T_2 \vee T_3}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T}$$

3. Joins and Meets

How to type if-then-else, in the presence of subsumtion?

$$\frac{t_1 : Bool \quad t_2 : T_2 \quad t_3 : T_3 \quad T = T_2 \vee T_3}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 : T}$$

$S \vee T$:= `Bool`, if $S = T =$ `Bool`

`{j1:J1,…,jq:Jq}` if $S = \{k_1{:}S_1,\ldots,k_m{:}S_m\}$, $T = \{l_1{:}T_1,\ldots,l_n{:}T_n\}$, $\{j_1{:}J_1,\ldots,j_q{:}J_q\} = S \cap T$, and $J_u = S_v \vee T_w$ for $j_u = k_v = l_w$

3. Joins and Meets

How to type if-then-else, in the presence of subsumtion?

$$\frac{t_1 : Bool \quad t_2 : T_2 \quad t_3 : T_3 \quad T = T_2 \vee T_3}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 : T}$$

$A \wedge C$:=
meet of A and C
= greatest common subtype

$S \vee T$:= `Bool`, if $S = T =$ `Bool`

`{j1:J1,…,jq:Jq}` if $S = \{k_1{:}S_1,\ldots,k_m{:}S_m\}$, $T = \{l_1{:}T_1,\ldots,l_n{:}T_n\}$, $\{j_1{:}J_1,\ldots,j_q{:}J_q\} = S \cap T$, and $J_u = S_v \vee T_w$ for $j_u = k_v = l_w$

$E \rightarrow F$, if $S = A \rightarrow B$ and $T = C \rightarrow D$ and $E = A \wedge C$ and $F = B \vee D$

3. Joins and Meets

How to type if-then-else, in the presence of subsumtion?

$$\frac{t_1 : Bool \quad t_2 : T_2 \quad t_3 : T_3 \quad T = T_2 \vee T_3}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 : T}$$

$A \wedge C$:=
meet of A and C
= greatest common subtype

$S \vee T$:= `Bool`, if $S = T =$ `Bool`

`{j1:J1,…,jq:Jq}` if $S = \{k_1{:}S_1,\ldots,k_m{:}S_m\}$, $T = \{l_1{:}T_1,\ldots,l_n{:}T_n\}$, $\{j_1{:}J_1,\ldots,j_q{:}J_q\} = S \cap T$, and $J_u = S_v \vee T_w$ for $j_u = k_v = l_w$

$E \rightarrow F$, if $S = A \rightarrow B$ and $T = C \rightarrow D$ and $E = A \wedge C$ and $F = B \vee D$

Top, otherwise.

3. Joins and Meets

How to type if-then-else, in the presence of subsumtion?

$$\frac{t_1 : Bool \quad t_2 : T_2 \quad t_3 : T_3 \quad T = T_2 \vee T_3}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 : T}$$

$A \wedge C$:=
meet of A and C
= greatest common subtype

$S \wedge T$:=

define this in a similar way as the join!

IN LAB TODAY:

Implement subtyping for our language.

First, leave if-then-else untouched. Then, add joins and meets.