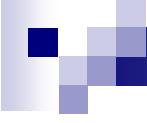


# Type Systems

Lecture 10 Dec. 22nd, 2004

Sebastian Maneth

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



# Today

# F-sub: kernel / full

1. System F-sub
2. Properties of F-sub
3. Algorithmic Typing for F-Sub
4. Algorithmic Subtyping for F-Sub
5. Joins and Meets

# 1. System F-Sub

## Bounded Quantification

```
f2poly = λx<:{a:Nat}. λx:x. {orig=x, asucc=succ(x.a)};
```

Has type       $\forall x <: \{a : \text{Nat}\}. x \rightarrow \{\text{orig}:x, \text{asucc}:\text{Nat}\}$

# 1. System F-Sub

## Bounded Quantification

`f2poly =  $\lambda x<:\{a:\text{Nat}\}. \lambda x:x. \{\text{orig}=x, \text{asucc}=\text{succ}(x.a)\};$`

Has type  $\forall x<:\{a:\text{Nat}\}. x \rightarrow \{\text{orig}:x, \text{asucc}:\text{Nat}\}$

---

How to derive it?

$\vdash \lambda x<:\{a:\text{Nat}\}. \lambda x:x. \{\text{orig}=x, \text{asucc}=\text{succ}(x.a)\} : \forall x<:\{a:\text{Nat}\}. x \rightarrow \{\text{orig}:x, \text{asucc}:\text{Nat}\}$

# 1. System F-Sub

## Bounded Quantification

type abstraction (remove  $\forall$ )

If bound satisfied, then term has specified type

$x : \{a : \text{Nat}\} \vdash$

$\lambda x : x. \{ \text{orig} = x, \text{asucc} = \text{succ}(x.a) \} : x \rightarrow \{ \text{orig} : x, \text{asucc} : \text{Nat} \}$

---

$\vdash \lambda x : \{a : \text{Nat}\}. \lambda x : x. \{ \text{orig} = x, \text{asucc} = \text{succ}(x.a) \} :$

$\forall x : \{a : \text{Nat}\}. x \rightarrow \{ \text{orig} : x, \text{asucc} : \text{Nat} \}$

# 1. System F-Sub

lambda abstraction (remove  $\rightarrow$ )

If argument type satisfied, then result term has specified result type.

$$\frac{\begin{array}{c} \text{X} : \{a : \text{Nat}\}, x : X \vdash \\ \{ \text{orig} = x, \text{asucc} = \text{succ}(x.a) \} : \{ \text{orig} : X, \text{asucc} : \text{Nat} \} \end{array}}{\text{X} : \{a : \text{Nat}\} \vdash \lambda x : X. \{ \text{orig} = x, \text{asucc} = \text{succ}(x.a) \} : X \rightarrow \{ \text{orig} : X, \text{asucc} : \text{Nat} \}}$$

---

$$\frac{\text{X} : \{a : \text{Nat}\} \vdash \lambda x : X. \{ \text{orig} = x, \text{asucc} = \text{succ}(x.a) \} : X \rightarrow \{ \text{orig} : X, \text{asucc} : \text{Nat} \}}{\vdash \lambda \text{X} : \{a : \text{Nat}\}. \lambda x : X. \{ \text{orig} = x, \text{asucc} = \text{succ}(x.a) \} : \forall \text{X} : \{a : \text{Nat}\}. X \rightarrow \{ \text{orig} : X, \text{asucc} : \text{Nat} \}}$$

# 1. System F-Sub

make record (remove { })

field terms have specified types

$\frac{}{x : \{a:\text{Nat}\}, x : x \vdash x : x}$

$\frac{}{x : \{a:\text{Nat}\}, x : x \vdash \text{succ}(x.a) : \text{Nat}}$

---

$\frac{}{x : \{a:\text{Nat}\}, x : x \vdash \{ \text{orig}=x, \text{asucc}=\text{succ}(x.a) \} : \{ \text{orig}:x, \text{asucc}:\text{Nat} \}}$

---

$\frac{}{\lambda x : x. \{ \text{orig}=x, \text{asucc}=\text{succ}(x.a) \} : x \rightarrow \{ \text{orig}:x, \text{asucc}:\text{Nat} \}}$

---

$\frac{}{\vdash \lambda x : \{a:\text{Nat}\}. \lambda x : x. \{ \text{orig}=x, \text{asucc}=\text{succ}(x.a) \} : \forall x : \{a:\text{Nat}\}. x \rightarrow \{ \text{orig}:x, \text{asucc}:\text{Nat} \}}$

# 1. System F-Sub

what now?

$$\frac{\frac{\frac{x : \{a:\text{Nat}\}, x : X \vdash x : \{a:\text{Nat}\}}{x : \{a:\text{Nat}\}, x : X \vdash x.a : \text{Nat}}}{x : \{a:\text{Nat}\}, x : X \vdash \text{succ}(x.a) : \text{Nat}}}{\frac{x : \{a:\text{Nat}\}, x : X \vdash \{ \text{orig}=x, \text{asucc}=\text{succ}(x.a) \} : \{ \text{orig}:X, \text{asucc}:\text{Nat} \}}{\frac{x : \{a:\text{Nat}\} \vdash \lambda x : X. \{ \text{orig}=x, \text{asucc}=\text{succ}(x.a) \} : X \rightarrow \{ \text{orig}:X, \text{asucc}:\text{Nat} \}}{\vdash \lambda x : \{a:\text{Nat}\}. \lambda x : X. \{ \text{orig}=x, \text{asucc}=\text{succ}(x.a) \} : \forall x : \{a:\text{Nat}\}. x \rightarrow \{ \text{orig}:X, \text{asucc}:\text{Nat} \}}}$$

# 1. System F-Sub

what now? → subsumption!

term may be of any subtype

$$x : \{a:\text{Nat}\}, x : x \vdash x : x \quad x : \{a:\text{Nat}\}, x : x \vdash x : \{a:\text{Nat}\}$$

$$x : \{a:\text{Nat}\}, x : x \vdash x : \{a:\text{Nat}\}$$

ok

$$x : \{a:\text{Nat}\}, x : x \vdash x.a : \text{Nat}$$

$$x : \{a:\text{Nat}\}, x : x \vdash x : x$$

$$x : \{a:\text{Nat}\}, x : x \vdash \text{succ}(x.a) : \text{Nat}$$

$$x : \{a:\text{Nat}\}, x : x \vdash$$

$$\{\text{orig}=x, \text{asucc}=\text{succ}(x.a)\} : \{\text{orig}:x, \text{asucc}:\text{Nat}\}$$

$$x : \{a:\text{Nat}\} \vdash$$

$$\lambda x : x. \{\text{orig}=x, \text{asucc}=\text{succ}(x.a)\} : x \rightarrow \{\text{orig}:x, \text{asucc}:\text{Nat}\}$$

$$\vdash \lambda x : \{a:\text{Nat}\}. \lambda x : x. \{\text{orig}=x, \text{asucc}=\text{succ}(x.a)\} :$$

$$\forall x : \{a:\text{Nat}\}. x \rightarrow \{\text{orig}:x, \text{asucc}:\text{Nat}\}$$

# 1. System F-Sub

$$\frac{\begin{array}{c} \text{ok} \\ X <: \{a:\text{Nat}\}, x:X \vdash x:X \end{array}}{X <: \{a:\text{Nat}\}, x:X \vdash x:{a:\text{Nat}}}$$
$$\frac{\begin{array}{c} \text{ok} \\ X <: \{a:\text{Nat}\}, x:X \vdash x:{a:\text{Nat}} \end{array}}{X <: \{a:\text{Nat}\}, x:X \vdash x.a:\text{Nat}}$$
$$\frac{\begin{array}{c} \text{ok} \\ X <: \{a:\text{Nat}\}, x:X \vdash x:X \end{array}}{X <: \{a:\text{Nat}\}, x:X \vdash \text{succ}(x.a):\text{Nat}}$$
$$\frac{X <: \{a:\text{Nat}\}, x:X \vdash \{ \text{orig}=x, \text{asucc}=\text{succ}(x.a) \} : \{ \text{orig}:X, \text{asucc}:\text{Nat} \}}{\lambda x:X. \{ \text{orig}=x, \text{asucc}=\text{succ}(x.a) \} : X \rightarrow \{ \text{orig}:X, \text{asucc}:\text{Nat} \}}$$
$$\frac{\vdash \lambda x <: \{a:\text{Nat}\}. \lambda x:X. \{ \text{orig}=x, \text{asucc}=\text{succ}(x.a) \} : \forall X <: \{a:\text{Nat}\}. X \rightarrow \{ \text{orig}:X, \text{asucc}:\text{Nat} \}}{\quad}$$

# 1. System F-Sub

**new typing rule**

type abstraction (remove  $\forall$ )

If bound satisfied, then term has specified type

$$\frac{\Gamma, X : B \vdash t : T}{\Gamma \vdash \lambda X : B. t : \forall X : B. T}$$

# 1. System F-Sub

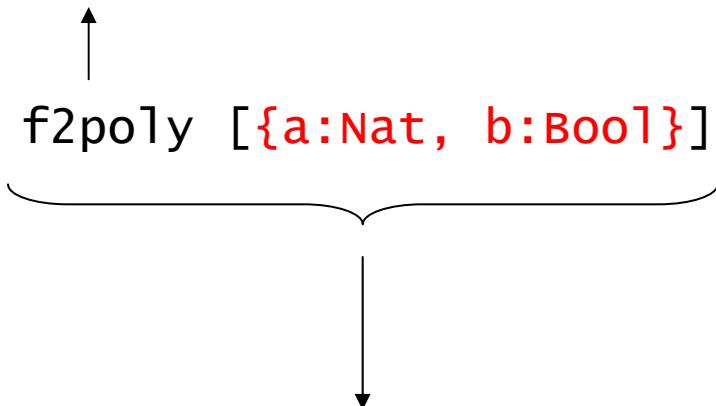
As in System F: to **apply** a polymorphic function,  
we have to **supply a concrete SUBtype C**.

```
( f2poly [{a:Nat, b:Bool}] ) {a=5, b=true}
```

# 1. System F-Sub

As in System F: to **apply** a polymorphic function,  
we have to **supply a concrete SUBtype C.**  
 $C <: \{a:\text{Nat}\}$

$\forall x <: \{a:\text{Nat}\}. x \rightarrow \{\text{orig}:x, \text{asucc}:\text{Nat}\}$



$\{a:\text{Nat}, b:\text{Bool}\} \rightarrow \{\text{orig}:\{a:\text{Nat}, b:\text{Bool}\}, \text{asucc}:\text{Nat}\}$

# 1. System F-Sub

How to derive it?

---

```
↑ f2poly [{a:Nat, b:Bool}]:  
          {a:Nat, b:Bool} → {orig:{a:Nat,b:Bool}, asucc:Nat}
```

# 1. System F-Sub

How to derive it?

Applying  $[x/\{a:\text{Nat}, b:\text{Bool}\}]$  must give specified type



$\vdash \text{f2poly} : \forall x <: \{a:\text{Nat}\}. x \rightarrow \{\text{orig}:x, \text{asucc}:\text{Nat}\}$        $\vdash \{a:\text{Nat}, b:\text{Bool}\} <: \{a:\text{Nat}\}$

---

$\vdash \text{f2poly } [\{a:\text{Nat}, b:\text{Bool}\}]:$   
 $\{a:\text{Nat}, b:\text{Bool}\} \rightarrow \{\text{orig}:\{a:\text{Nat}, b:\text{Bool}\}, \text{asucc}:\text{Nat}\}$

# 1. System F-Sub

(record subtyping)

ok

ok

$\vdash f2poly : \forall X <: \{a:\text{Nat}\}. X \rightarrow \{\text{orig}:X, \text{asucc}:\text{Nat}\}$        $\vdash \{a:\text{Nat}, b:\text{Bool}\} <: \{a:\text{Nat}\}$

---

$\vdash f2poly [\{a:\text{Nat}, b:\text{Bool}\}] :$   
 $\{a:\text{Nat}, b:\text{Bool}\} \rightarrow \{\text{orig}:\{a:\text{Nat}, b:\text{Bool}\}, \text{asucc}:\text{Nat}\}$

# 1. System F-Sub

**new typing rule**

type application (remove [ ])

$t$  polymorphic with bound  $x <: B$        $C$  subtype of  $B$

$$\frac{\Gamma \vdash t : \forall x <: B. T \quad \Gamma \vdash C <: B}{\Gamma \vdash t[c] : [x/c]T}$$

# 1. System F-Sub

## Typing Rules

- Usual Lambda-rules (T-VAR, T-ABS, T-APP)
- type abstraction
- type application (uses subtyping on bounds!)
- subsumption (uses subtyping!)

# 1. System F-Sub

## Subtyping Rules

→ reflexivity, transitivity, Top (evth. is  $<:\text{Top}$ )

→ type variables:

$$\frac{x <: T \in \Gamma}{\Gamma \vdash x <: T}$$

→ function (S-ARROW)

(covariant on result,  
contravariant on argument)

# 1. System F-Sub

**new subtyping rule**      (for quantified types)

---

$$\Gamma \vdash \forall X : B_1 . T_1 \quad <: \quad \forall X : B_2 . T_2$$

# 1. System F-Sub

**new subtyping rule** (for quantified types)

which bound??

$\Gamma, x : B \vdash T_1 <: T_2$

---

$\Gamma \vdash \forall x : B_1 . T_1 <: \forall x : B_2 . T_2$

# 1. System F-Sub

**new subtyping rule** (for quantified types)

Must have **SAME bound B**

“the *kernel* rule”

$$\frac{\Gamma, x : B \vdash T_1 <: T_2}{\Gamma \vdash \forall x : B. T_1 <: \forall x : B. T_2}$$

→ with this simple rule: **Kernel F-Sub**

# 1. System F-Sub

“the *kernel* rule”

$$\frac{\Gamma, X <: B \vdash T_1 <: T_2}{\Gamma \vdash \forall X <: B. T_1 <: \forall X <: B. T_2}$$

$$\forall X <: \{a: \text{Nat}\}. \{a: \text{Nat}, b: \text{Bool}\} <: \forall X <: \{a: \text{Nat}\}. \{a: \text{Nat}\}$$

If expected type is  $\forall X <: \{a: \text{Nat}\}. \{a: \text{Nat}\}$   
then also fu. of type  $\forall X <: \{a: \text{Nat}\}. \{a: \text{Nat}, b: \text{Bool}\}$  is OK!

# 1. System F-Sub

“the *kernel rule*”

$$\frac{\Gamma, X <: B \vdash T_1 <: T_2}{\Gamma \vdash \forall X <: B. T_1 <: \forall X <: B. T_2}$$

$$\forall X <: \{a: \text{Nat}\}. \{a: \text{Nat}, b: \text{Bool}\} <: \forall X <: \{a: \text{Nat}\}. \{a: \text{Nat}\}$$

If expected type is  $\forall X <: \{a: \text{Nat}\}. \{a: \text{Nat}\}$   
then also fu. of type  $\forall X <: \{a: \text{Nat}\}. \{a: \text{Nat}, b: \text{Bool}\}$  is OK!

→ Will only be instantiated by subtypes  $X$  of  $\{a: \text{Nat}\}$

THUS,  $\forall X <: \text{Top}$  is OK too, because it asks for less! (for the least, actually..)

$\forall X <: \{a: \text{Nat}, d: \text{Nat}\}$  NOT OK, because we only know  $X$  will be  $<: \{a: \text{Nat}\}$

# 1. System F-Sub

Full F-Sub

$$\frac{\Gamma \vdash B_2 <: B_1 \quad \Gamma, X <: B_2 \vdash T_1 <: T_2}{\Gamma \vdash \forall X <: B_1 . T_1 <: \forall X <: B_2 . T_2}$$

- covariant on result
- contravariant on bounds

## 2. Properties of F-Sub

### Preservation

If  $t \vdash t:T$  and  $t \rightarrow t'$ , then  $\Gamma \vdash t':T$ .

### Progress

If  $t$  is a closed and well-typed term, then either  
 $t$  is a value, or  
 $t \rightarrow t'$  for some term  $t'$ .

Proofs: Induction on the structure of terms.

→ Use canonical forms lemma:

If  $v$  is closed value of type  $T_1 \rightarrow T_2$ , then  $v = \lambda x:S_1 . t_2$

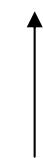
If  $v$  is closed value of type  $\forall X<:T_1 . T_2$ , then  $v = \lambda X<:T_1 . t_2$ .

### 3. Algorithmic Typing for F-Sub

Idea of type checking algorithm for simply typed lambda-calculus w. subtyping:

→ calculate the *minimal type* of terms

Apply same idea for F-Sub!



Now application, and then  
subsumption, and all is OK!

$$x : \text{Nat} \rightarrow \text{Nat}, y : x \vdash y 5 : \text{Nat}$$

---

$$x : \text{Nat} \rightarrow \text{Nat} \vdash \lambda y : x. y 5 : x \rightarrow \text{Nat}$$

---

$$\vdash \lambda x : \text{Nat} \rightarrow \text{Nat}. \lambda y : x. y 5 : \forall x. x : \text{Nat} \rightarrow \text{Nat}. x \rightarrow \text{Nat}$$

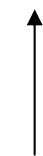
### 3. Algorithmic Typing for F-Sub

Idea of type checking algorithm for simply typed lambda-calculus w. subtyping:

→ calculate the *minimal type* of terms

Apply same idea for F-Sub!

→ subsumption is NOT syntax-directed!! Can we do without??



Now application, and then  
subsumption, and all is OK!

$$x : \text{Nat} \rightarrow \text{Nat}, y : x \vdash y 5 : \text{Nat}$$

---

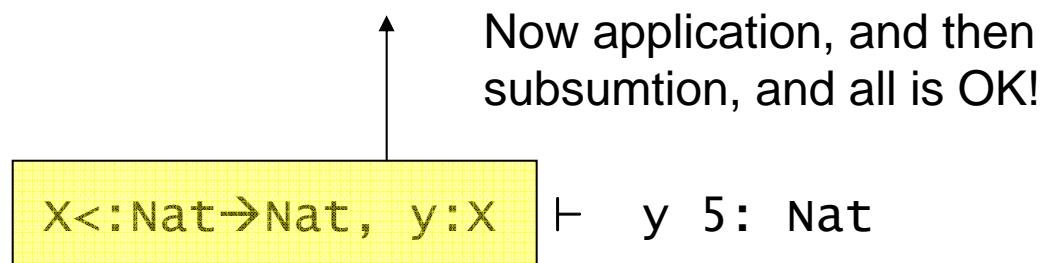
$$x : \text{Nat} \rightarrow \text{Nat} \vdash \lambda y : x. y 5 : x \rightarrow \text{Nat}$$

---

$$\vdash \lambda x : \text{Nat} \rightarrow \text{Nat}. \lambda y : x. y 5 : \forall x. x : \text{Nat} \rightarrow \text{Nat}. x \rightarrow \text{Nat}$$

### 3. Algorithmic Typing for F-Sub

→ subsumption is NOT syntax-directed!! Can we do without??



→  $y$  must be arrow type!

→ smallest (non-variable) arrow type, that is a supertype of  $X$

$\Gamma \vdash x \uparrow \text{Nat} \rightarrow \text{Nat}$

“ $x$  exposes to  $\text{Nat} \rightarrow \text{Nat}$  under  $\Gamma$ ”

### 3. Algorithmic Typing for F-Sub

**Exposure:**

$$x : T \in \Gamma \quad \Gamma \vdash T \uparrow T'$$

---

$$\Gamma \vdash x \uparrow T'$$

T is not a type variable

---

$$\Gamma \vdash T \uparrow T$$

Example:  $\Gamma = x : \text{Nat}, y : \text{Nat} \rightarrow \text{Nat}, z : y, w : z$

Then  $\Gamma \vdash z \uparrow \text{Nat} \rightarrow \text{Nat}$  and  $\Gamma \vdash w \uparrow \text{Nat} \rightarrow \text{Nat}$

**Lemma.** If  $\Gamma \vdash S \uparrow T$ , then

- (1)  $\Gamma \vdash S : T$
- (2) If  $\Gamma \vdash S : U$  and  $U$  is not a variable, then  $\Gamma \vdash T : U$ .

### 3. Algorithmic Typing for F-Sub

New rule for application, includes argument subsumtion:

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : T_1 \quad \Gamma \vdash T_1 \uparrow (D \rightarrow E) \\ \Gamma \vdash t_2 : T_2 \quad \Gamma \vdash T_2 <: D \end{array}}{\Gamma \vdash t_1 t_2 : E} \text{ TA-APP}$$

In Example:

$$\frac{\begin{array}{c} \Gamma \vdash y : x \quad \Gamma \vdash x \uparrow (\text{Nat} \rightarrow \text{Nat}) \\ \Gamma \vdash 5 : \text{Nat} \quad \Gamma \vdash \text{Nat} <: \text{Nat} \end{array}}{\underbrace{x <: \text{Nat} \rightarrow \text{Nat}, y : x \vdash y 5 : \text{Nat}}_{\Gamma}} \text{ TA-APP}$$

### 3. Algorithmic Typing for F-Sub

New rule for type application, includes argument subsumption:

Old:

$$\frac{\Gamma \vdash t : \forall X <: B. T \quad \Gamma \vdash C <: B}{\Gamma \vdash t [C] : [x/C]T}$$

New:

$$\frac{\Gamma \vdash t : T_1 \quad \Gamma \vdash T_1 \uparrow \forall X <: B. T \quad \Gamma \vdash C <: B}{\Gamma \vdash t [C] : [x/C]T} \text{ TA-TAPP}$$

### 3. Algorithmic Typing for F-Sub

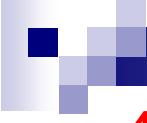
$\vdash = \underbrace{\text{T-VAR}, \text{T-ABS}, \text{T-APP}, \text{T-TABS}, \text{T-TAPP}, \text{T-SUB}}$   
as in simply typed

$\vdash = \text{T-VAR}, \text{T-ABS}, \text{TA-APP}, \text{T-TABS}, \text{TA-TAPP}$

**Theorem.** (correctness of minimal typing)

- (1) If  $\Gamma \vdash t:T$ , then  $\Gamma \vdash t:T$  (soundness)
- (2) If  $\Gamma \vdash t:T$ , then  $\Gamma \vdash t:M$  with  $\Gamma \vdash M <: T$ . (completeness)

**Corollary.** The relation  $\vdash$  is decidable, given a decision procedure for the subtype relation.



## 4. Algorithmic *Subtyping* for F-Sub

### Subtyping Rules

- reflexivity, transitivity, Top (evth. is <:Top)
- type variables
- function (S-ARROW)
- quantified types (kernel / full)

## 4. Algorithmic *Subtyping* for F-Sub

### Problematic Subtyping Rules

- reflexivity, transitivity, Top
- type variables
- function (S-ARROW)
- quantified types (kernel / full)

## 4. Algorithmic Subtyping for F-Sub

### Problematic Subtyping Rules

- reflexivity, transitivity, Top
- type variables
- function (S-ARROW)
- quantified types (kernel / full)

Reflexivity is ONLY needed for variables:  $\Gamma \vdash X <: X$  not derivable without it!

→ Replace reflexivity by new rule:  $\Gamma \vdash X <: X.$

## 4. Algorithmic *Subtyping* for F-Sub

### Problematic Subtyping Rules

- $\Gamma \vdash X <: X$ ,      **transitivity**, Top
- type variables
- function (S-ARROW)
- quantified types (kernel / full)

Transitivity:  $\Gamma = W <: \text{Top}, V <: W, U <: V, X <: U$

$\Gamma \vdash Z <: W$  can ONLY be proved using transitivity.

$$\frac{\begin{array}{c} X <: U \in \Gamma \\ \vdots \\ \Gamma \vdash X <: U \quad \Gamma \vdash U <: W \end{array}}{\Gamma \vdash X <: W}$$

## 4. Algorithmic Subtyping for F-Sub

### Problematic Subtyping Rules

- $\Gamma \vdash X <: X$ ,      **transitivity**, Top
- type variables
- function (S-ARROW)
- quantified types (kernel / full)

Transitivity:  $\Gamma = T <: \text{Top}, V <: T, U <: V, X <: U$

$\Gamma \vdash Z <: W$  can ONLY be proved using transitivity.

$$\frac{\begin{array}{c} X <: U \in \Gamma \\ \hline \Gamma \vdash X <: U \end{array} \quad : \quad \begin{array}{c} \Gamma \vdash U <: T \end{array}}{\Gamma \vdash X <: T}$$

ONLY essential use of  
transitivity!

## 4. Algorithmic Subtyping for F-Sub

### Algorithmic Subtyping Rules

- $\Gamma \vdash X <: X$ , Top
- function (S-ARROW)
- quantified types (kernel / full)

→ Replace **transitivity**, and **type vars** by **new rule**:

$X <: U \in \Gamma$

$\Gamma \vdash U <: T$

---

$\Gamma \vdash X <: T$

## 4. Algorithmic Subtyping for F-Sub

### Algorithmic Subtyping Rules (kernel)

$$\Gamma \vdash S <: \text{Top}$$

$$\Gamma \vdash x <: x$$

$$x <: u \in \Gamma \quad \Gamma \vdash u <: t$$

$$\Gamma \vdash x <: t$$

$$\Gamma \vdash T_1 <: S_1 \quad \Gamma \vdash S_2 <: T_2$$

$$\Gamma \vdash S_1 \rightarrow S_2 <: T_1 \rightarrow T_2$$

$$\Gamma, x <: b \vdash T_1 <: T_2$$

$$\Gamma \vdash \forall x <: b. T_1 <: \forall x <: b. T_2$$

## 4. Algorithmic Subtyping for F-Sub

### Algorithmic Subtyping Rules (kernel)

$$\Gamma \vdash S <: \text{Top}$$

$$\Gamma \vdash x <: x$$

$$x <: u \in \Gamma \quad \Gamma \vdash u <: t$$

$$\Gamma \vdash x <: t$$

$$\Gamma \vdash T_1 <: S_1 \quad \Gamma \vdash S_2 <: T_2$$

$$\Gamma \vdash S_1 \rightarrow S_2 <: T_1 \rightarrow T_2$$

$$\Gamma, x <: b \vdash T_1 <: T_2$$

$$\Gamma \vdash \forall x <: b. T_1 <: \forall x <: b. T_2$$

**Theorem.**  $\Gamma \vdash S <: T$  if and only if  $\Gamma \vdash S <: T$ .

**Theorem.** The subtyping algorithm terminates on all inputs.

→ Subtyping in kernel F-sub is decidable.

## 4. Algorithmic Subtyping for F-Sub

### Algorithmic Subtyping Rules (FULL)

$$\Gamma \vdash S <: \text{Top}$$

$$\Gamma \vdash X <: X$$

$$\Gamma \vdash T_1 <: S_1 \quad \Gamma \vdash S_2 <: T_2$$

$$\Gamma \vdash S_1 \rightarrow S_2 <: T_1 \rightarrow T_2$$

$$X <: U \in \Gamma \quad \Gamma \vdash U <: T$$

$$\Gamma \vdash X <: T$$

$$\frac{\Gamma, X <: B \vdash T_1 <: T_2}{\Gamma \vdash \forall X <: B . T_1 <: \forall X <: B . T_2}$$

$$\Gamma \vdash B_2 <: B_1 \quad \Gamma, X <: B_2 \vdash T_1 <: T_2$$

$$\Gamma \vdash \forall X <: B_1 . T_1 <: \forall X <: B_2 . T_2$$

## 4. Algorithmic Subtyping for F-Sub

Algorithmic Subtyping Rules (FULL) **do not terminate!!**

→ Construct a **vicious circle**:

Define  $\neg S = \forall X <: S . X$

Then,  $\Gamma \vdash \neg S < : \neg T$  iff  $T <: S$ . (Contravariance on bounds..)

$T := \forall X <: \text{Top} . \neg(\forall Y <: X . \neg Y)$

Try to show that

$$x_0 <: T \quad \vdash \quad x_0 <: \forall x_1 <: x_0 . \neg x_1$$

## 4. Algorithmic Subtyping for F-Sub

Algorithmic Subtyping Rules (FULL) **do not terminate!!**

→ Construct a **vicious circle**:

Define  $\neg S = \forall X <: S . X$

Then,  $\Gamma \vdash \neg S < : \neg T$  iff  $T <: S$ . (Contravariance on bounds..)

$T := \forall X <: \text{Top} . \neg(\forall Y <: X . \neg Y)$

Try to show that

$$\begin{array}{lll} x_0 <: T & \vdash & x_0 <: \forall x_1 <: x_0 . \neg x_1 \\ x_0 <: T & \vdash & \forall x_1 <: \text{Top} . \neg(\forall x_2 <: x_1 . \neg x_2) <: \forall x_1 <: x_0 . \neg x_1 \end{array}$$

## 4. Algorithmic Subtyping for F-Sub

Algorithmic Subtyping Rules (FULL) **do not terminate!!**

→ Construct a **vicious circle**:

Define  $\neg S = \forall X <: S . X$

Then,  $\Gamma \vdash \neg S < : \neg T$  iff  $T <: S$ . (Contravariance on bounds..)

$$T := \forall X <: \text{Top} . \neg(\forall Y <: X . \neg Y)$$

Try to show that

$$\begin{array}{lll} x_0 <: T & \vdash & x_0 <: \forall x_1 <: x_0 . \neg x_1 \\ x_0 <: T & \vdash & \forall x_1 <: \text{Top} . \neg(\forall x_2 <: x_1 . \neg x_2) <: \forall x_1 <: x_0 . \neg x_1 \\ x_0 <: T, x_1 <: x_0 & \vdash & \neg(\forall x_2 <: x_1 . \neg x_2) <: \neg x_1 \end{array}$$

## 4. Algorithmic Subtyping for F-Sub

Algorithmic Subtyping Rules (FULL) **do not terminate!!**

→ Construct a **vicious circle**:

Define  $\neg S = \forall X <: S . X$

Then,  $\Gamma \vdash \neg S < : \neg T$  iff  $T <: S$ . (Contravariance on bounds..)

$$T := \forall X <: \text{Top} . \neg(\forall Y <: X . \neg Y)$$

Try to show that

$$\begin{array}{lll} x_0 <: T & \vdash & x_0 <: \forall x_1 <: x_0 . \neg x_1 \\ x_0 <: T & \vdash & \forall x_1 <: \text{Top} . \neg(\forall x_2 <: x_1 . \neg x_2) <: \forall x_1 <: x_0 . \neg x_1 \\ x_0 <: T, x_1 <: x_0 & \vdash & \neg(\forall x_2 <: x_1 . \neg x_2) <: \neg x_1 \\ x_0 <: T, x_1 <: x_0 & \vdash & x_1 <: \forall x_2 <: x_1 . \neg x_2 . \end{array}$$

## 4. Algorithmic Subtyping for F-Sub

Algorithmic Subtyping Rules (FULL) do not terminate!!

→ Construct a vicious circle:

Define  $\neg S = \forall X <: S . X$

Then,  $\Gamma \vdash \neg S < : \neg T$  iff  $T <: S$ . (Contravariance on bounds..)

$T := \forall X <: \text{Top} . \neg(\forall Y <: X . \neg Y)$

Try to show that

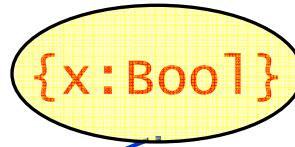
$$\begin{array}{lll} x_0 <: T & \vdash & x_0 <: \forall x_1 <: x_0 . \neg x_1 \\ x_0 <: T & \vdash & \forall x_1 <: \text{Top} . \neg(\forall x_2 <: x_1 . \neg x_2) \\ x_0 <: T, x_1 <: x_0 & \vdash & x_1 <: \forall x_1 <: x_0 . \neg x_1 \\ x_0 <: T, x_1 <: x_0 & \vdash & \neg(\forall x_2 <: x_1 . \neg x_2) \\ & & \vdash \forall x_2 <: x_1 . \neg x_2 \end{array}$$

## 5. Joins and Meets

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

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

What is the type of this term?

-  take the *least* (most precise) common supertype of S and T  
or  $\{x:\text{Top}\}?$
- <: = “the *join* of S and T”
- or  $\{\}?$  =:  $S \vee T$
- <:
- or  $\text{Top}?$

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

## 5. Joins and Meets

$\Gamma \vdash S \vee T :=$	$T$	if $\Gamma \vdash S <: T$
	$S$	if $\Gamma \vdash T <: S$
	$J$	if $S=X, X <: U \in \Gamma$ , and $\Gamma \vdash U \vee T = J$
	$J$	if $T=X, X <: U \in \Gamma$ , and $\Gamma \vdash S \vee U = J$
$M \rightarrow J$		if $S=S_1 \rightarrow S_2, T=T_1 \rightarrow T_2,$ $\Gamma \vdash S_1 \wedge T_1 = M$ , and $\Gamma \vdash S_1 \vee T_2 = J$
$\forall X <: U. J_2$		if $S = \forall X <: U. S_2$ $T = \forall X <: U. T_2$ $\Gamma, X <: U \vdash S_2 \vee T_2 = J_2$
Top		otherwise

## 5. Joins and Meets

In kernel F-sub:

- every pair  $S, T$  has (effectively) a **join**
- if  $S$  and  $T$  have at least one subtype in common, then they have (effectively) a **meet**

In full F-sub: NO / NO

# Summary

formalism	comput. power	type checking	type reconstr.
Simply typed lambda calculus	normalizing	lin.time	poly.time
Simply typed lambda calculus + subtyping	normalizing	lin.time	poly.time
Featherweight Java	r.e. compl.	lin.time	
Let-Polymorphism / Prenex-Polymorphism	normalizing	lin.time	EXPTIME
System F	normalizing	lin.time	UNDEC.
System F + subtyping (kernel)	normalizing	poly.time	UNDEC.
System F + subtyping (full)	normalizing	UNDEC.	UNDEC.