编程语言的设计原理
Design Principles of Programming Languages

Haiyan Zhao, Di Wang
赵海燕，王迪

Peking University, Spring Term 2023

Chap 26: Bounded Quantification

Polymorphism + Subtyping

Kernel and Full $F_{<:}$

Examples

Properties

Bounded Existential Types

Motivation

Limitation of Subtyping

```
f = λx:{a:Nat}. x;
▶ f : {a:Nat} → {a:Nat}

ra = {a=0};
f ra;
▶ {a=0} : {a:Nat}

rab = {a=0, b=true};
f rab;
▶ {a=0, b=true} : {a:Nat}
```

By passing `rab` through the identify function, we have lost the ability to access its `b` field!

Question

We have studied System F, which supports parametric polymorphism. Could it help?

Motivation

```
fpoly = λX. λx:X. x;
▶ f : ∀X. X → X

fpoly [{a:Nat, b:Bool}] rab;
▶ {a=0, b=true} : {a:Nat, b:Bool}
```

Limitation of Universal Quantification

```
f2 = λx:{a:Nat}. {orig=x, asucc=succ(x.a)};
▶ f2 : {a:Nat} → {orig:{a:Nat}, asucc:Nat}

f2 rab;
▶ {orig={a=0,b=true}, asucc=1} : {orig:{a:Nat}, asucc:Nat}

f2poly = λX. λx:X. {orig=x, asucc=succ(x.a)};
▶ Error: expected record type
```

Motivation

Solution: Bounded Quantification

We want to express in the type of f2 that it can take any record type R with a numeric a field as its argument.

$$R <: \{a : Nat\}$$

In the quantification, we introduce a **subtyping constraint** on the bounded variable X:

```
f2poly = λX<:{a:Nat}. λx:X. {orig=x, asucc=succ(x.a)};
▶ f2poly = ∀X<:{a:Nat}. X → {orig:X, asucc:Nat}

f2poly [{a:Nat,b:Bool}] rab;
▶ {orig={a=0,b=true}, asucc=1} : {orig:{a:Nat,b:Bool}, asucc:Nat}
```

System $F_{<:}$

Syntax, Evaluation, and Typing

Syntax

$$t ::= \ldots \mid \lambda X <: T.\, t \mid t\,[T] \qquad v ::= \ldots \mid \lambda X <: T.\, t$$

$$T ::= X \mid \mathsf{Top} \mid T \to T \mid \forall X <: T.T \qquad \Gamma ::= \varnothing \mid \Gamma, x : T \mid \Gamma, X <: T$$

Evaluation

$$\frac{}{(\lambda X <: T_{11}.\, t_{12})\,[T_2] \longrightarrow [X \mapsto T_2]t_{12}} \ \textsc{E-TappTabs}$$

Typing

$$\frac{\Gamma, X <: T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda X <: T_1.\, t_2 : \forall X <: T_1.T_2} \ \textsc{T-TAbs}$$

$$\frac{\Gamma \vdash t_1 : \forall X <: T_{11}.T_{12} \qquad \Gamma \vdash T_2 <: T_{11}}{\Gamma \vdash t_1\,[T_2] : [X \mapsto T_2]T_{12}} \ \textsc{T-TApp}$$

Subtyping (for Kernel $F_{<:}$)

Hypothetical Subtyping

$$\frac{}{\Gamma \vdash S <: S} \text{ S-Refl} \qquad \frac{\Gamma \vdash S <: U \qquad \Gamma \vdash U <: T}{\Gamma \vdash S <: T} \text{ S-Trans} \qquad \frac{}{\Gamma \vdash S <: \mathsf{Top}} \text{ S-Top} \qquad \frac{X <: T \in \Gamma}{\Gamma \vdash X <: T} \text{ S-TVar}$$

$$\frac{\Gamma \vdash T_1 <: S_1 \qquad \Gamma \vdash S_2 <: T_2}{\Gamma \vdash S_1 \to S_2 <: T_1 \to T_2} \text{ S-Arrow} \qquad \frac{\Gamma, X <: U_1 \vdash S_2 <: T_2}{\Gamma \vdash \forall X <: U_1 . S_2 <: \forall X <: U_1 . T_2} \text{ S-All}$$

Subsumption

System $F_{<:}$ has one structural typing rule:

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

System $F_{<:}$ is a Conservative Extension of System F

Bounded and Unbounded Quantification

$F_{<:}$ provides only bounded quantification, but it actually covers unbounded quantification of pure System F.

$$\forall X.T \quad \stackrel{\text{def}}{=} \quad \forall X <: \text{Top}.T$$

Scoping of Type Variables

Scoping

$\Gamma \vdash t : T$ indicates that free type variables in t and T should be in the domain of Γ.

What about free type variables appearing in the types **inside** Γ?

$$\begin{array}{lcll}
\Gamma_1 & = & X <: \mathsf{Top}, y : X \to \mathsf{Nat} & ✓ \\
\Gamma_2 & = & y : X \to \mathsf{Nat}, X <: \mathsf{Top} & ✗ \\
\Gamma_3 & = & X <: \{a : \mathsf{Nat}, b : X\} & ✗ \\
\Gamma_4 & = & X <: \{a : \mathsf{Nat}, b : Y\}, Y <: \{c : \mathsf{Bool}, d : X\} & ✗
\end{array}$$

Whenever we mention a type T in a context, the free variables of T should be bound in the portion of the context to the **left** of where type T appears.

Aside

We can introduce a well-formedness judgement for contexts:

$$\frac{}{\vdash \varnothing \text{ context}} \qquad \frac{\vdash \Gamma \text{ context} \qquad \Gamma \vdash T \text{ type}}{\vdash \Gamma, x : T \text{ context}} \qquad \frac{\vdash \Gamma \text{ context} \qquad \Gamma \vdash T \text{ type}}{\vdash \Gamma, X <: T \text{ context}}$$

Subtyping (for Full $F_{<:}$)

Observation

We can think of a universal quantifier as a sort of **arrow type** whose elements are functions from **types** to **terms**. The kernel-$F_{<:}$ rule (S-ALL) corresponds to something like

$$\frac{S_2 <: T_2}{U \to S_2 <: U \to T_2}$$

However, the standard subtyping rule for arrows is

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

"Full" Bounded Quantification

$$\frac{\Gamma \vdash T_1 <: S_1 \qquad \Gamma, X <: T_1 \vdash S_2 <: T_2}{\Gamma \vdash \forall X <: S_1 . S_2 <: \forall X <: T_1 . T_2} \quad \text{S-ALL}$$

Examples

Encoding Pairs

Remark

We have reviewed that in pure System F, we can encode pairs as follows:

```
Pair T1 T2 = ∀X. (T1→T2→X) → X;
pair = λX. λY. λx:X. λy:Y. (λR. λp:X→Y→R. p x y) as Pair X Y;
▶ pair : ∀X. ∀Y. X → Y → Pair X Y
fst = λX. λY. λp:Pair X Y. p [X] (λx:X. λy:Y. x);
▶ fst : ∀X. ∀Y. Pair X Y → X
snd = λX. λY. λp:Pair X Y. p [Y] (λx:X. λy:Y. y);
▶ snd : ∀X. ∀Y. Pair X Y → Y
```

Question (Exercise 26.3.1)

The encodings also work in System $F_{<:}$. Show that the subtyping rule for pairs follows directly from the encoding.

$$\frac{\Gamma \vdash S_1 <: T_1 \qquad \Gamma \vdash S_2 <: T_2}{\Gamma \vdash \texttt{Pair}\, S_1\, S_2 <: \texttt{Pair}\, T_1\, T_2}$$

Encoding Pairs

$$
\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, X <: \mathsf{Top} \vdash S_1 <: T_1 \qquad \dfrac{\Gamma, X <: \mathsf{Top} \vdash S_2 <: T_2 \qquad \dfrac{}{\Gamma, X <: \mathsf{Top} \vdash X <: X}}{\Gamma, X <: \mathsf{Top} \vdash T_2 \to X <: S_2 \to X}}{\Gamma, X <: \mathsf{Top} \vdash T_1 \to T_2 \to X <: S_1 \to S_2 \to X} \qquad \dfrac{}{\Gamma, X <: \mathsf{Top} \vdash X <: X}}{\Gamma, X <: \mathsf{Top} \vdash (S_1 \to S_2 \to X) \to X <: (T_1 \to T_2 \to X) \to X}}{\Gamma \vdash \forall X.(S_1 \to S_2 \to X) \to X <: \forall X.(T_1 \to T_2 \to X) \to X}\ \textsc{S-All}}{\Gamma \vdash \mathtt{Pair}\, S_1\, S_2 <: \mathtt{Pair}\, T_1\, T_2}
$$

Lemma (Weakening, 26.4.2(4))

If $\Gamma \vdash S <: T$ and $\Gamma, X <: U$ is well formed, then $\Gamma, X <: U \vdash S <: T$.

Encoding Tuples

Definition

For each $n \geqslant 0$ and types T_1 through T_n, let

$$\{T_i{}^{i \in 1 \ldots n}\} \stackrel{\text{def}}{=} \mathtt{Pair}\ T_1\ (\mathtt{Pair}\ T_2\ \ldots\ (\mathtt{Pair}\ T_n\ \mathtt{Top})\ \ldots)$$

In particular, $\{\} \stackrel{\text{def}}{=} \mathtt{Top}$. Then for terms t_1 through t_n, let

$$\{t_i{}^{i \in 1 \ldots n}\} \stackrel{\text{def}}{=} \mathtt{pair}\ t_1\ (\mathtt{pair}\ t_2\ \ldots\ (\mathtt{pair}\ t_n\ \mathtt{top})\ \ldots)$$

The projection $t.n$ is

$$\mathtt{fst}\ (\underbrace{\mathtt{snd}\ (\mathtt{snd}\ \ldots\ (\mathtt{snd}}_{n-1 \text{ times}}\ t)\ \ldots))$$

Proposition

The following rules follow directly from the encoding of tuples:

$$\frac{\forall i \in 1 \ldots n : \Gamma \vdash S_i <: T_i}{\Gamma \vdash \{S_i{}^{i \in 1 \ldots n+k}\} <: \{T_i{}^{i \in 1 \ldots n}\}} \qquad \frac{\forall i \in 1 \ldots n : \Gamma \vdash t_i : T_i}{\Gamma \vdash \{t_i{}^{i \in 1 \ldots n}\} : \{T_i{}^{i \in 1 \ldots n}\}} \qquad \frac{\Gamma \vdash t : \{T_i{}^{i \in 1 \ldots n}\}}{\Gamma \vdash t.i : T_i}$$

Church Encodings with Subtyping

Remark

Recall that in System F, numbers can be encoded by

$$\texttt{CNat} = \forall \texttt{X}.\ (\texttt{X}\to\texttt{X}) \to \texttt{X} \to \texttt{X}$$

This can be read as:

- "Tell me an arbitrary result type T;
- give me an **'induction function'** on T and a **'base element'** of T; and
- I'll give you another element of T formed by iterating the induction function n times over the base element."

Definition

We generalize `CNat` by adding two bounded quantifiers:

$$\texttt{SNat} = \forall \texttt{X}.\ \forall \texttt{S<:X}.\ \forall \texttt{Z<:X}.\ (\texttt{X}\to\texttt{S}) \to \texttt{Z} \to \texttt{X}$$

The "induction function" maps from the whole set X into the subset S and the "base element" is from the subset Z.
In other words, it distinguishes the base case and the induction case at the type level.

Church Encodings with Subtyping

Type Refinements

```
SNat = ∀X. ∀S<:X. ∀Z<:X. (X→S) → Z → X;

SZero = ∀X. ∀S<:X. ∀Z<:X. (X→S) → Z → Z;
szero = λX. λS<:X. λZ<:X. λs:(X→S). λz:Z. z;
▶ szero : SZero

SPos = ∀X. ∀S<:X. ∀Z<:X. (X→S) → Z → S;
ssucc = λn:SNat.
          λX. λS<:X. λZ<:X. λs:(X→S). λz:Z. s (n [X] [S] [Z] s z);
▶ ssucc : SNat → SPos
```

Question (Homework: Exercise 26.3.5)

Generalize the type `CBool` of Church booleans to a type `SBool` and two subtypes `STrue` and `SFalse`. Write a function `notft : SFalse → STrue` and a similar one `nottf : STrue → SFalse`.

Properties

Preservation

Theorem (Preservation, 26.4.13)

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

Lemma (Substitution preserves typing, 26.4.6)

If $\Gamma, x : Q, \Delta \vdash t : T$ and $\Gamma \vdash q : Q$, then $\Gamma, \Delta \vdash [x \mapsto q]t : T$.

Lemma (Type substitution preserves typing, 26.4.9)

If $\Gamma, X <: Q, \Delta \vdash t : T$ and $\Gamma \vdash P <: Q$, then $\Gamma, [X \mapsto P]\Delta \vdash [X \mapsto P]t : [X \mapsto P]T$.

Lemma (Type substitution preserves subtyping, 26.4.8)

If $\Gamma, X <: Q, \Delta \vdash S <: T$ and $\Gamma \vdash P <: Q$, then $\Gamma, [X \mapsto P]\Delta \vdash [X \mapsto P]S <: [X \mapsto P]T$.

Progress

Theorem (Progress, 26.4.15)

If t is a closed, well-typed $F_{<:}$-term, then either t is a value or else there is some t' with $t \longrightarrow t'$.

Lemma (Canonical Forms, 26.4.14)

- If v is a closed value of type $T_1 \to T_2$, then v has the form $\lambda x{:}S_1 .\, t_2$.
- If v is a closed value of type $\forall X <: T_1 . T_2$, then v has the form $\lambda X <: T_1 .\, t_2$.

Bounded Existential Types

Bounded Existential Quantification (Kernel Variant)

New Syntactic Forms

$$T ::= \ldots \mid \{\exists X <: T, T\}$$

New Typing Rules

T-PACK

$$\frac{\Gamma \vdash t_2 : [X \mapsto U]T_2 \qquad \Gamma \vdash U <: T_1}{\Gamma \vdash \{{}^*U, t_2\} \text{ as } \{\exists X <: T_1, T_2\} : \{\exists X <: T_1, T_2\}}$$

T-UNPACK

$$\frac{\Gamma \vdash t_1 : \{\exists X <: T_{11}, T_{12}\} \qquad \Gamma, X <: T_{11}, x : T_{12} \vdash t_2 : T_2}{\Gamma \vdash \texttt{let}\ \{X, x\} = t_1\ \texttt{in}\ t_2 : T_2}$$

New Subtyping Rules

$$\frac{\Gamma, X <: U \vdash S_2 <: T_2}{\Gamma \vdash \{\exists X <: U, S_2\} <: \{\exists X <: U, T_2\}} \text{ T-SOME}$$

An Example

```
counterADT =
   {*Nat, {new = 1, get = λi:Nat. i, inc = λi:Nat. succ(i)}}
   as {∃Counter<:Nat,
        {new: Counter, get: Counter→Nat, inc: Counter→Counter}};
▶ counterADT : {∃Counter<:Nat,
                  {new:Counter,get:Counter→Nat,inc:Counter→Counter}}

let {Counter,counter} = counterADT in
counter.get (counter.inc (counter.inc counter.new));
▶ 3 : Nat

let {Counter,counter} = counterADT in
succ (succ (counter.inc counter.new));
▶ 4 : Nat

let {Counter,counter} = counterADT in
counter.inc 3;
▶ Error: parameter type mismatch
```

Homework

Question (Exercise 26.3.5)

Generalize the type `CBool` of Church booleans to a type `SBool` and two subtypes `STrue` and `SFalse`. Write a function `notft : SFalse → STrue` and a similar one `nottf : STrue → SFalse`.