Subtyping variance and its application

SIG of Mathematics, SAST of NJUPT

June 5, 2024

Untyped λ-calculus

term	t
abstraction	$\lambda x.t$
application	$t_1\ t_2$

Example 1

$$
\begin{aligned}
& (\lambda x.x)\ ((\lambda x.x)\ (\lambda z.\,(\lambda x.x)\ z)) \\
\rightarrow\ & (\lambda x.x)\ ((\lambda x.x)\ (\lambda z.z)) \\
\rightarrow\ & (\lambda x.x)\ (\lambda z.z) \\
\rightarrow\ & (\lambda z.z) \\
\rightarrow\ & (\lambda x.x)
\end{aligned}
$$

Simply typed λ-calculus ($\lambda_{\to}$)

term	t
abstraction	$\lambda x : T.t$
application	$t_1\ t_2$
type of functions	$T_1 \to T_2$
typing context	Γ

Example 2

$$\frac{\dfrac{\Gamma, x : T_1 \vdash t_1 : T_2}{\Gamma \vdash \lambda x : T_1.t_1 : T_1 \to T_2} \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash (\lambda x : T_1.t_1)\ \ t_2 : T_2}$$

Simply typed λ-calculus with subtyping ($\lambda_{<:}$)

term	t	type of functions	$T_1 \to T_2$
abstraction	$\lambda x : T.t$	typing context	Γ
application	$t_1\ t_2$	subtyping	$S <: T$

Properties of subtyping

reflexive $S <: S$

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

top and bottom $\mathsf{Bot} <: S <: \mathsf{Top}$

Subtyping variance

covariant $$\frac{S <: T}{\text{list of } S <: \text{list of } T}$$

contravariant $$\frac{S <: T}{T \to U <: S \to U}$$

invariant $$\frac{S <: T}{S \to S \text{ is neither subtype nor supertype of } T \to T}$$

Application: lifetime in Rust's type system

A piece of Rust code

```
{
    let foo: &u8 = &1;
    {
        let bar = &foo;
        println!("bar = {}", **bar);
    }
}
```

Application: lifetime in Rust's type system

Pseudocode with explicit lifetime

```
'b: {
    let foo: &'b u8 = &1;
    'a: {
        let bar: &'a &'b u8 = &foo;
        println!("bar = {}", **bar);
    }
}
```

$$\text{Bot} = \texttt{\&'static T} <: \texttt{\&'b T} <: \texttt{\&'a T}$$

Soundness hole in Rust's type system

Oops!

```
static UNIT: &'static &'static () = &&();

fn foo<'a, 'b, T>(_: &'a &'b (), v: &'b T) -> &'a T { v }

fn bad<'a, T>(x: &'a T) -> &'static T {
    let f: fn(_, &'a T) -> &'static T = foo;
    f(UNIT, x)
}
```

Soundness hole in Rust's type system

What compiler thought to be correct

```
fn foo<'a, 'b, T>(_: &'a &'b (), v: &'b T) -> &'a T

fn foo<'a, 'b, T>(_: &'a &'static (), v: &'b T) -> &'a T

fn foo<'b, T>(_: &'static &'static (), v: &'b T) -> &'static T
```

Soundness hole in Rust's type system

Manual solution

```
fn foo<'a, 'b, T>(_: &'a &'b (), v: &'b T) -> &'a T
    where 'b: 'a
```

Thanks!