Type checker for System F

Mudathir Mahgoub

May 2, 2018

1 Project Problem

This project is a type checker for annotated simple typed lambda calculus and system F. It supports subtyping for simple types. In general, a type checker would decide whether $\Gamma \vdash t : T$ is derivable: can the term t be assigned the type T under the typing context Γ?

Section 2 describes the software and its architecture. Section 3 describes the rules used for simple types and provides some examples. It also explains how subsumption rule can be delayed until variable terms are generated, and provides a proof for correctness. Section 3 describes the rules used for system F and provides some examples.

2 Software description

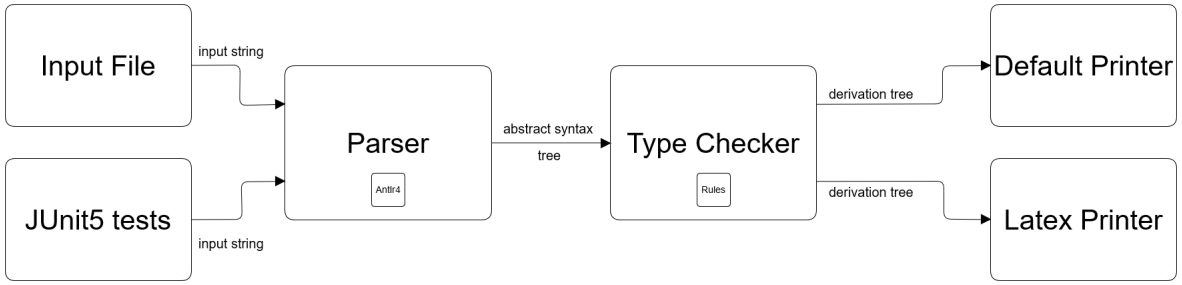


Figure 1: Project architecture.

The project is implemented using Java and the executable is a jar file (TypeChecker.jar). The program receives as an input a text file containing subtypes definitions and a judgment to be checked. For testing JUnit5 was used to test the program directly without files. Below is an example of an input:

Listing 1: test.txt

```
SubBase(bool, int);
. |- \lambda x. \lambda y. (x y)[bool]: (int ->T) -> (bool -> T);
```

Here is the default output.

Listing 2: java -jar TypeChecker.jar -i test.txt

```
Yes
                                    SubBase(bool, int)
-------------------(var)           -------------------(subBase)
x: bool ? x : bool                      bool <: int
-----------------------------------------(subsumption)
x: bool ? x : int
```

3 Simple types

4 Rules

1. $$\frac{\Gamma(x) = T}{\Gamma \vdash x : T}\ \text{var}$$

2. $$\frac{\Gamma \vdash t_1 : T_1 \to T_2 \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash (t_1 t_2)[T_1] : T_2}\ \text{app}$$

3. $$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \lambda x.t : T_1 \to T_2}\ \lambda$$

4. $$\frac{\Gamma \vdash t : T_1 \qquad T_1 <: T_2}{\Gamma \vdash t : T_2}\ \text{subsumption}$$

5. $$\frac{}{T <: T}\ \text{reflexive}$$

6. $$\frac{SubBase(b_1, b_2)}{b_1 <: b_2}\ \text{subBase}$$

7. $$\frac{T_1' <: T_1 \qquad T_2 <: T_2'}{T_1 \to T_2 <: T2_1 \to T_2'}\ \text{arrow}$$

8. $$\frac{T_1 <: T_2 \qquad T_2 <: T_3}{T_1 <: T_3}\ \text{transitive}$$

5 Variable terms

5.1 Valid

$$\frac{}{x : T \vdash x : T}\ \text{var}$$

5.2 Invalid

$$\frac{}{\cdot \vdash x : T}\ \text{invalid var}$$

6 Lambda & application terms

$$\frac{\dfrac{\dfrac{\dfrac{}{x : (T1 \to T2), y : T1 \vdash x : (T1 \to T2)}\ \text{var} \qquad \dfrac{}{x : (T1 \to T2), y : T1 \vdash y : T1}\ \text{var}}{x : (T1 \to T2), y : T1 \vdash (x\ y)[T1] : T2}\ \text{app}}{x : (T1 \to T2) \vdash \lambda y.(x\ y)[T1] : (T1 \to T2)}\ \lambda}{\cdot \vdash \lambda x.\lambda y.(x\ y)[T1] : ((T1 \to T2) \to (T1 \to T2))}\ \lambda$$

7 Direct Subtyping

7.1 Valid

$$\frac{\dfrac{}{x : bool \vdash x : bool}\ \text{var} \qquad \dfrac{\text{SubBase}(bool, int)}{bool <: int}\ \text{subBase}}{x : bool \vdash x : int}\ \text{subsumption}$$

7.2 Invalid

$$\dfrac{\dfrac{}{x : int \vdash x : int}\ \text{var} \qquad \dfrac{\dfrac{\bot}{int <: bool}\ \text{invalid}}{x : int \vdash x : bool}\ \text{subsumption}$$

8 Transitive Subtyping

$$\dfrac{\dfrac{}{x : bool \vdash x : bool}\ \text{var} \qquad \dfrac{\dfrac{\dfrac{\text{SubBase}(bool, int)}{bool <: int}\ \text{subBase} \qquad \dfrac{\text{SubBase}(int, double)}{int <: double}\ \text{subBase}}{bool <: double}\ \text{transitive}}{x : bool \vdash x : double}\ \text{subsumption}$$

$$\dfrac{\dfrac{}{x : bool \vdash x : bool}\ \text{var} \qquad \dfrac{\dfrac{\dfrac{\text{SubBase}(bool, int)}{bool <: int}\ \text{subBase} \qquad \dfrac{\text{SubBase}(int, quotient)}{int <: quotient}\ \text{subBase}}{bool <: quotient}\ \text{transitive} \qquad \dfrac{\text{SubBase}(quotient, double)}{quotient <: double}\ \text{subBase}}{bool <: double}\ \text{transitive}}{x : bool \vdash x : double}\ \text{subsumption}$$

9 Arrow Types

9.1 Valid

$$\dfrac{\dfrac{}{x : (int \to bool) \vdash x : (int \to bool)}\ \text{var} \qquad \dfrac{\dfrac{\dfrac{\text{SubBase}(bool, int)}{bool <: int}\ \text{subBase} \qquad \dfrac{\text{SubBase}(bool, int)}{bool <: int}\ \text{subBase}}{(int \to bool) <: (bool \to int)}\ \text{arrow}}{x : (int \to bool) \vdash x : (bool \to int)}\ \text{subsumption}$$

9.1.1 Reflexive

$$\dfrac{\dfrac{}{x : (int \to bool) \vdash x : (int \to bool)}\ \text{var} \qquad \dfrac{\dfrac{\dfrac{}{int <: int}\ \text{reflexive} \qquad \dfrac{\text{SubBase}(bool, int)}{bool <: int}\ \text{subBase}}{(int \to bool) <: (int \to int)}\ \text{arrow}}{x : (int \to bool) \vdash x : (int \to int)}\ \text{subsumption}$$

9.1.2 Nested

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : (int \to T), y : bool \vdash x : (int \to T)}\ \text{var} \qquad \dfrac{\dfrac{\dfrac{\text{SubBase}(bool, int)}{bool <: int}\ \text{subBase} \qquad \dfrac{}{T <: T}\ \text{reflexive}}{(int \to T) <: (bool \to T)}\ \text{arrow}}{x : (int \to T), y : bool \vdash x : (bool \to T)}\ \text{subsumption} \qquad \dfrac{}{x : (int \to T), y : bool \vdash y : bool}\ \text{var}}{x : (int \to T), y : bool \vdash (x\ y)[bool] : T}\ \text{app}}{x : (int \to T) \vdash \lambda y.(x\ y)[bool] : (bool \to T)}\ \lambda}{\cdot \vdash \lambda x.\lambda y.(x\ y)[bool] : ((int \to T) \to (bool \to T))}\ \lambda$$

9.2 Invalid

$$\dfrac{\dfrac{}{x : (bool \to bool) \vdash x : (bool \to bool)}\ \text{var} \qquad \dfrac{\dfrac{\dfrac{\bot}{int <: bool}\ \text{invalid} \qquad \dfrac{\text{SubBase}(bool, int)}{bool <: int}\ \text{subBase}}{(bool \to bool) <: (int \to int)}\ \text{arrow}}{x : (bool \to bool) \vdash x : (int \to int)}\ \text{subsumption}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : (bool \to T), y : bool \vdash x : (bool \to T)}\ \text{var} \qquad \dfrac{\dfrac{\dfrac{\bot}{int <: bool}\ \text{invalid} \qquad \dfrac{}{T <: T}\ \text{reflexive}}{(bool \to T) <: (int \to T)}\ \text{arrow}}{x : (bool \to T), y : bool \vdash x : (int \to T)}\ \text{subsumption} \qquad \dfrac{}{x : (bool \to T), y : bool \vdash y : bool}\ \text{var}}{x : (bool \to T), y : bool \vdash y : int}\ \text{app} \qquad \dfrac{\text{SubBase}(bool, int)}{bool <: int}\ \text{subBase}}{x : (bool \to T), y : bool \vdash (x\ y)[int] : T}\ \text{subsumption}}{x : (bool \to T) \vdash \lambda y.(x\ y)[int] : (bool \to T)}\ \lambda}{\cdot \vdash \lambda x.\lambda y.(x\ y)[int] : ((bool \to T) \to (bool \to T))}\ \lambda$$

9.3 Delaying applying the subsumption rule

The sumbumption rule can be delayed until the term in the judgment is a just a variable. This simplifies the code since there is only one rule to be applied for application and λ abstraction terms. Here is the correctness proof.

$$\frac{\Gamma \vdash t : T \qquad T <: T'}{\Gamma \vdash t : T'} \text{ subsumption}$$

Proof:
By induction hypothesis on the structure of the term.

1. Case $t = x$: trivial since t is a variable.

$$\frac{\Gamma \vdash x : T \qquad T <: T'}{\Gamma \vdash x : T'} \text{ subsumption}$$

2. Case $t = t_1 t_2$: assume $T_2 <: T_2'$ and the following derivation:

$$\frac{\dfrac{\Gamma \vdash t_1 : T_1 \to T_2 \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1 t_2[T_1] : T_2} \text{ app} \qquad T_2 <: T_2'}{\Gamma \vdash t_1 t_2[T_1] : T_2'} \text{ subsumption}$$

We can get a different derivation tree where the sumbsumption rule is delayed:

$$\frac{\dfrac{\Gamma \vdash t_1 : T_1 \to T_2'}{\Gamma \vdash t_1 t_2[T_1] : T_2} \text{ subsumption} \qquad \dfrac{\dfrac{}{T_1 <: T_1} \text{ reflexive} \qquad T_2 <: T_2'}{T_1 \to T_2 <: T_1 \to T_2'} \text{ arrow} \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1 t_2[T_1] : T_2'} \text{ app}$$

By the induction hypothesis, the subsumption rule can be delayed until variable terms are generated in the derivation of $\Gamma \vdash t_1 : T_1 \to T_2'$ and $\Gamma \vdash t_2 : T_1$.

3. Case $t = \lambda x.t'$: assume $T_1 \to T_2 <: T_1' \to T_2'$ and the following derivation:

$$\frac{\dfrac{\dfrac{\Gamma, x : T_1 \vdash t' : T_2}{\Gamma \vdash \lambda x.t' : T_1 \to T_2} \lambda \qquad \dfrac{T_1' <: T_1 \qquad T_2 <: T_2'}{T_1 \to T_2 <: T_1' \to T_2'} \text{ arrow}}{\Gamma \vdash \lambda x.t' : T_1' \to T_2'} \text{ subsumption}$$

Alternatively we can derive:

$$\frac{\dfrac{\Gamma, x : T_1' \vdash t' : T_2 \qquad T_2 <: T_2'}{\Gamma, x : T_1' \vdash t' : T_2'} \text{ subsumption}}{\Gamma \vdash \lambda x.t' : T_1' \to T_2'} \lambda$$

Since $T_1' <: T_1$ then by the subsumption rule:

$$\frac{\dfrac{}{\Gamma, x : T_1' \vdash x : T_1'} \text{ var} \qquad T_1' <: T_1}{\Gamma, x : T_1' \vdash x : T_1} \text{ subsumption}$$

Therefore if $\Gamma, x : T_1 \vdash t' : T_2$ then $\Gamma, x : T_1' \vdash t' : T_2$ which concludes the proof.

Note:

If $\Gamma, x : T_1' \vdash t' : T_2$, it is not always true that $\Gamma, x : T_1 \vdash t' : T_2$ where $T_1' <: T_1$. A counter example would be

$$x : bool \vdash x : int \nRightarrow x : double \vdash x : int, \quad bool <: int <: double$$

However

$$x : int \vdash x : double \Rightarrow x : bool \vdash x : double, \quad bool <: int <: double$$

Therefore in the following example, using the subsumption rule first would fail and slow the type checker because it needs to backtrack and check the λ rule. However using the subsumption rule would prove the type checking and is faster.

$$\dfrac{\dfrac{\dfrac{x : double \vdash x : int}{\Gamma \vdash \lambda x.x : double \to int}\,\lambda \qquad \dfrac{\dfrac{SubBase(bool, double)}{bool <: double}\,\text{subBase} \qquad \dfrac{}{int <: int}\,\text{reflexive}}{double \to int <: bool \to int}\,\text{arrow}}{\Gamma \vdash \lambda x.x : bool \to int}\,\text{subsumption}$$

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x : bool \vdash x : bool}\,\text{var} \qquad \dfrac{SubBase(bool, int)}{bool <: int}\,\text{suBase}}{\Gamma, x : bool \vdash x : int}\,\text{subsumption}}{\Gamma \vdash \lambda x.x : bool \to int}\,\lambda$$

10 System F

10.1 Variables

10.1.1 Valid

$$\dfrac{}{x : \forall X.X \vdash x : \forall X.X}\,\text{var}$$

$$\dfrac{}{x : \forall X.X \vdash x : \forall Y.Y}\,\text{var}$$

$$\dfrac{}{x : \forall X.(X \to X) \vdash x : \forall Y.(Y \to Y)}\,\text{var}$$

Y is free in the context and the type:

$$\dfrac{}{x : \forall X.(X \to Y) \vdash x : \forall Z.(Z \to Y)}\,\text{var}$$

$$\dfrac{\dfrac{}{x : \forall X.X \vdash x : \forall X1.X1}\,\text{var}}{x : \forall X.X \vdash x[Y] : Y}\,\text{elimination}$$

$$\dfrac{\dfrac{}{x : \forall X.X \vdash x : \forall X1.X1}\,\text{var}}{x : \forall X.X \vdash x[(Y \to Y)] : (Y \to Y)}\,\text{elimination}$$

$$\dfrac{\dfrac{\dfrac{}{x : \forall X.X \vdash x : \forall X1.X1}\,\text{var}}{x : \forall X.X \vdash x[[(\forall X.X \to \forall X.X)]][(\forall X.X \to \forall X.X)] : (\forall X.X \to \forall X.X)}\,\text{elimination} \qquad \dfrac{}{x : \forall X.X \vdash x : \forall X.X}\,\text{var}}{x : \forall X.X \vdash (x[[(\forall X.X \to \forall X.X)]]\ x)[\forall X.X] : \forall X.X}\,\text{app}$$

10.1.2 Invalid

Y is free in the context:

$$\frac{}{x : \forall X.(X \to Y) \vdash x : \forall Y.(Y \to Y)}\ \text{invalid var}$$

Y is free in the context, and Z is free in the type:

$$\frac{}{x : \forall X.(X \to Y) \vdash x : \forall Y.(Y \to Z)}\ \text{invalid var}$$

$$\dfrac{\dfrac{}{x : \forall X.X \vdash x : \forall X1.X1}\ \text{var}}{x : \forall X.X \vdash x[Y] : Y}\ \text{elimination}$$

10.2 Numbers

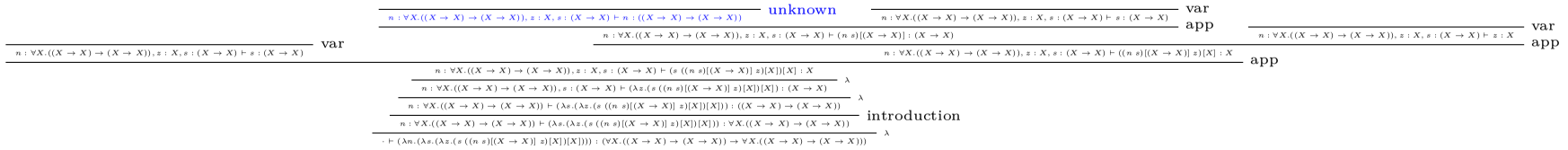
Zero

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z : X, s : (X \to X) \vdash z : X}\ \text{var}}{s : (X \to X) \vdash (\lambda z.z) : (X \to X)}\ \lambda}{\cdot \vdash (\lambda s.(\lambda z.z)) : ((X \to X) \to (X \to X))}\ \lambda}{\cdot \vdash (\lambda s.(\lambda z.z)) : \forall X.((X \to X) \to (X \to X))}\ \text{introduction}$$

Zero with free variable X

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y : X, z : X_2, s : (X_2 \to X_2) \vdash z : X_2}\ \text{var}}{y : X, s : (X_2 \to X_2) \vdash (\lambda z.z) : (X_2 \to X_2)}\ \lambda}{y : X \vdash (\lambda s.(\lambda z.z)) : ((X_2 \to X_2) \to (X_2 \to X_2))}\ \lambda}{y : X \vdash (\lambda s.(\lambda z.z)) : \forall X_2.((X_2 \to X_2) \to (X_2 \to X_2))}\ \text{introduction}}{y : X \vdash (\lambda s.(\lambda z.z)) : \forall X.((X \to X) \to (X \to X))}\ \text{renaming}$$

Successor missing annotation



Successor well annotated