Homework # 5
due March 25, 13:00

Please turn in §2 on paper and §3 electronically. The skeleton file `stlc.slf.SKEL` is available on the course web page.

1 Reading

Please read Chapter 9 in the textbook.

2 Discussion

The first attempts to define type rules often fail. Consider the following dubious type rules:

$$\text{T-VarA}\ \frac{}{x : T} \qquad \text{T-AbsA}\ \frac{t : T}{\lambda x.t : T \to T} \qquad \text{T-AppA}\ \frac{t_1 : T \to T' \qquad t_2 : T}{(t_1\, t_2) : T'}$$

... additional rules for booleans and "`if`" ...

1. Can one prove progress for closed terms for this type system? Explain! If not, give a counter-example.

   > Progress still holds for closed terms. The only values given function type are indeed functions, so an application of a value to a value will always succeed (for one further step of evaluation).

2. What about preservation?

   > The term $\lambda x.xx$ can be given the type `Bool` $\to$ `Bool`:
   >
   > $$\dfrac{\dfrac{\dfrac{}{x : \texttt{Bool} \to \texttt{Bool}}\text{T-VarA} \quad \dfrac{}{x : \texttt{Bool}}\text{T-VarA}}{xx : \texttt{Bool}}\text{T-AppA}}{\lambda x.(xx) : \texttt{Bool} \to \texttt{Bool}}\text{T-AbsA}$$
   >
   > Then we can type $(\lambda x.(xx))$`true` : `Bool`, but after one step of execution, we have "`true true`" which cannot be typed.

3. A type system can be sound but not useful. Are there simple, reasonable program that this type system rejects? Explain with examples.

   > The term $(\lambda x.\lambda y.x)$ `true false` should be capable of receiving type `Bool`, requiring the function to receive type `Bool` $\to$ `Bool` $\to$ `Bool`, but the type system can only type functions which have the same input and output types, and so the term cannot be typed although it is simple and reasonable.

3 Proofs

Complete the proof of the type safety of the simply-typed lambda calculus (STLC) using the skeleton provided. In particular:

3.5.7′ Prove non-evaluation of values for STLC.

9.3.1 Not needed: SASyLF's `inversion` suffices.

9.3.2 Prove that if $\Gamma \vdash t\, t : T$ then we have a contradiction.

9.3.3 Done for you.

9.3.4 Done for you.

9.3.5 Progress: You need to prove the application case.

9.3.6 Not needed: SASyLF's `exchange` suffices.

9.3.7 Not needed: SASyLF's `weakening` suffices.

9.3.8 Substitution

9.3.9 Preservation