COMP 3610 Tutorial 4

24 August, 2023

Exercise 1

1. Extend IMP with a new operator "/" for integer division. This operator has a special case for when the divisor is 0. In this case, it should raise an exception. Pick one of the two exception semantics with try-catch-blocks from Section 9 to model this. Give the extensions to the grammar, typing rules, and operational semantics.
2. Write a program that uses the division operator within a try-block. Show how it type-checks.

Exercise 2

In Section 10, we proposed and dismissed the following two possible rules for subtyping between reference types:

$$\frac{T <: T'}{T\ \mathbf{ref} <: T'\ \mathbf{ref}} \qquad \frac{T' <: T}{T\ \mathbf{ref} <: T'\ \mathbf{ref}}$$

For each of them, write a program that would type-check if we used the respective rule, but that would go wrong if you run it. Show a bad state it would step to, and explain what is wrong.

Exercise 3

For each of the following subtypings, either show the proof tree or give a program that would type-check but also go wrong (similar to Exercise 2) if that subtyping would hold. Assume nat <: int.

1. $\{\} \to \{p : \text{int}\} <: \{q : \text{bool}\} \to \{p : \text{int}\}$
2. $\{\} \to \{p : \text{int}\} <: \{q : \text{bool}\} \to \{p : \text{int}, q : \text{bool}\}$
3. $\{q : \text{bool}\} \to \{p : \text{int}\} <: \{\} \to \{p : \text{int}\}$
4. $\{q : \text{bool}\} \to \{p : \text{int}\}\ \mathbf{ref} <: \{\} \to \{p : \text{int}\}$
5. $(\{q : \text{bool}\} \to \{p : \text{int}\}) \to \text{nat} <: (\{\} \to \{p : \text{int}\}) \to \text{int}$
6. $(\{q : \text{bool}\} \to \{p : \text{int}\}\ \mathbf{ref}) \to \text{nat} <: (\{\} \to \{p : \text{int}\}\ \mathbf{ref}) \to \text{int}$

Exercise 4

Prove the following statement: For all $\Gamma, E, E', T, T', T'', x$, if $x \notin dom(\Gamma)$, $\Gamma \vdash E : T$, $T <: T''$, and $\Gamma, x : T'' \vdash E' : T'$, then $\Gamma \vdash \{E/x\}E' : T''$

Bonus Exercise

So far, when we "built" a typing proof or tried to find a step in the operational semantics, we could simply derive every part by just searching for applicable rules and doing pattern matching. The `s-trans` rule for subtyping prevents this, because it requires us to come up with the middle type T'. In our system, the only reason we need this rule explicitly is because we split up record subtyping into three rules. Suppose that instead of `s-rcd1`, `s-rcd2`, and `s-rcd3`, we used the following rule:

$$\textsc{s-rcd}\ \frac{\forall 1 \leq i \leq m. \exists 1 \leq j \leq n. lab_j = lab'_i \wedge T_j <: T_i}{\{lab_1 : T_1, ..., lab_n : T_n\} <: \{lab'_1 : T'_1, ..., lab'_m : T'_m\}}$$

Consider a variant of $<:$ that not use the rules `s-trans`, `s-rcd1`, `s-rcd2`, and `s-rcd3`, but instead possibly `s-rcd`. Let us call this variant $<:^A$, while the original version without `s-rcd` is still called $<:$.

1. Prove (by rule induction) that for any T_1, T_2, and T_3, if $T_1 <:^A T_2$ and $T_2 <:^A T_3$, then $T_1 <:^A T_3$.
2. Prove (by rule induction) that for any T and T', if $T <: T'$, then $T <:^A T'$.