COMP3161/COMP9161

Datatypes and Type Safety Exercises

Liam O'Connor-Davis

September 14, 2018

1. **Safety and Liveness Properties**

   (a) [⋆] For each of the following properties, identify if it is a safety or a liveness property.

   i. When I come home, there must be beer in the fridge.
   ii. When I come home, I'll drop onto the couch and drink a beer.
   iii. I'll be home later.
   iv. When process p has executed line 5, then process q must execute line 17 again.
   v. When process p has executed line 5, then process q cannot execute line 17 again.
   vi. Process q cannot execute line 17 again unless process p has executed line 5.
   vii. Process p has to execute line 5 before q can execute line 17 again.

   (b) [⋆⋆⋆⋆] By considering a property as a set of behaviours (infinite sequences of states), show that if the state space Σ has at least two states, then any property can be expressed as the intersection of two liveness properties.
   Hint: It may be helpful to know that the union of a liveness property and any other property is also a liveness property (this result follows from the fact that liveness properties are dense sets).

2. **Type Safety**: Consider this very simple language with function application and two built-in functions:

$$\begin{array}{rcl} e & ::= & (\mathtt{App}\ e_1\ e_2) \\ & | & \mathsf{S} \\ & | & \mathsf{K} \end{array}$$

   The dynamic semantics evaluate the left hand side of applications as much as possible:

$$\frac{e_1 \mapsto e_1'}{e_1\ e_2 \mapsto e_1'\ e_2}$$

   The K function takes two arguments and returns the first one.

$$\frac{}{(\mathtt{App}\ (\mathtt{App}\ \mathsf{K}\ x)\ y) \mapsto x}$$

   The S function takes three arguments, applies the first argument to the third, and applies the result of that to the second argument applied to the third. More clearly:

$$\frac{}{(\mathtt{App}\ (\mathtt{App}\ (\mathtt{App}\ \mathsf{S}\ x)\ y)\ z) \mapsto (\mathtt{App}\ (\mathtt{App}\ x\ z)\ (\mathtt{App}\ y\ z))}$$

   (a) [⋆⋆] Define a set of typing rules for this language, where the set of types is described by:

$$\begin{array}{rcl} \tau & ::= & \tau_1 \to \tau_2 \\ & | & \iota \end{array}$$

   Note that $\to$ is right-associative, so $\tau_1 \to \tau_2 \to \tau_3$ means $\tau_1 \to (\tau_2 \to \tau_3)$.

(b) [⋆⋆⋆] In order to prove that your typing rules are type-safe, we must prove *progress* and *preservation*. For progress, we will define the set of final states as all states that have no successor:

$$F = \{s \mid \nexists s'.\ s \mapsto s'\}$$

This trivially satisfies progress, as progress states that all well-typed states either have a successor state or are final states.

Preservation, however, requires a nontrivial proof. Prove preservation for your typing rules with respect to the dynamic semantics of this language.

3. **Haskell Types**: Determine a MinHS type that is isomorphic to the following Haskell type declarations:

   (a) [⋆] `data MaybeInt = Just Int | Nothing`

   (b) [⋆] `data Nat = Zero | Suc Nat`

   (c) [⋆] `data IntTree = Tree Int IntTree IntTree | Leaf Int`

4. **Inhabitation**: Do the following MinHS types contain any (finite) values? If not, explain why. If so, give an example value.

   (a) [⋆] $\texttt{rec}\, t.\, \texttt{Int} + t$

   (b) [⋆] $\texttt{rec}\, t.\, \texttt{Int} \times t$

   (c) [⋆] $(\texttt{rec}\, t.\, \texttt{Int} \times t) + \texttt{Bool}$

5. **Encodings**: For each of the following sets, give a MinHS type that corresponds to it. Justify why your MinHS type is equivalent to the set, for example by providing a bijective function that, given a element of that set, gives the corresponding MinHS value of the corresponding type.

   (a) [⋆] The natural number set $\mathbb{N}$.

   (b) [⋆⋆] The set of integers $\mathbb{Z}$.

   (c) [⋆⋆] The set of rational numbers $\mathbb{Q}$.

   (d) [⋆⋆⋆] The set of (computable) real numbers $\mathbb{R}_{\text{TM}}$. It may be useful to assume a lazy semantics.

6. **Curry-Howard**: Give a term in typed λ-calculus that is a proof of the following propositions. If there is no such term, explain why.

   (a) [⋆] $A \Rightarrow A \vee B$

   (b) [⋆] $A \wedge B \Rightarrow A$

   (c) [⋆⋆] $P \vee P \Leftrightarrow P$
   Hint: Recall that $A \Leftrightarrow B$ is shorthand for $A \Rightarrow B \wedge B \Rightarrow A$.

   (d) [⋆⋆] $(A \wedge B \Rightarrow C) \Leftrightarrow (A \Rightarrow B \Rightarrow C)$

   (e) [⋆⋆] $P \vee (Q \wedge R) \Rightarrow (P \vee Q) \wedge (P \vee R)$

   (f) [⋆⋆] $P \Rightarrow \neg(\neg P)$
   Hint: Recall that $\neg A$ is shorthand for $A \Rightarrow \bot$.

   (g) [⋆⋆⋆] $\neg(\neg P) \Rightarrow P$

   (h) [⋆⋆⋆] $\neg(\neg(\neg P)) \Rightarrow \neg P$

   (i) [⋆⋆⋆] $(P \vee \neg P) \Rightarrow \neg(\neg P) \Rightarrow P$