Assignment 3
Nontermination

15-814: Types and Programming Languages
Frank Pfenning

Due Tuesday, October 1, 2019

Task 1 (L6.2, 15 points) Consider adding a new expression $\bot$ to our call-by-value language (with functions and Booleans) with the following evaluation and typing rules:

$$\frac{}{\bot \mapsto \bot}\ \mathsf{step/bot} \qquad \frac{}{\Gamma \vdash \bot : \tau}\ \mathsf{bot}$$

We do not change our notion of value, that is, $\bot$ is not a value.

1. Does preservation (Theorem L6.2) still hold? If not, provide a counterexample. If yes, show how the proof has to be modified to account for the new form of expression.

2. Does the canonical forms theorem (L6.4) still hold? If not, provide a counterexample. If yes, show how the proof has to be modified to account for the new form of expression.

3. Does progress (Theorem L6.3) still hold? If not, provide a counterexample. If yes, show how the proof has to be modified to account for the new form of expression.

Once we have nonterminating computation, we sometimes compare expressions using *Kleene equality*: e_1 and e_2 are Kleene equal ($e_1 \simeq e_2$) if they evaluate to the same value, or they both diverge (do not compute to a value). Since we assume we cannot observe functions, we can further restrict this definition: For $\cdot \vdash e_1 : \mathsf{bool}$ and $\cdot \vdash e_2 : \mathsf{bool}$ we write $e_1 \simeq e_2$ iff for all values v, $e_1 \mapsto^* v$ iff $e_2 \mapsto^* v$.

4. Give an example of two closed terms e_1 and e_2 of type bool such that $e_1 \simeq e_2$ but not $e_1 =_\beta e_2$, or indicate that no such example exists (no proof needed in either case).

Task 2 (L6.3, 15 points) In our call-by-value language with functions, Booleans, and $\bot$ (see Task 1) consider the following specification of *or*, sometimes called "short-circuit or":

$$\begin{array}{lcl} \mathit{or}\ \mathsf{true}\ e & \simeq & \mathsf{true} \\ \mathit{or}\ \mathsf{false}\ e & \simeq & e \end{array}$$

where $e_1 \simeq e_2$ is Kleene equality from Task 1.

- We cannot define a *function* $\mathit{or} : \mathsf{bool} \to (\mathsf{bool} \to \mathsf{bool})$ with this behavior. Prove that it is indeed impossible.

- Show how to translate an expression *or* $e_1 \;\; e_2$ into our language so that it satisfies the specification, and verify the given equalities by calculation.

Task 3 (L6.4, 30 points) In our call-by-value language with functions, Booleans, and $\bot$ (see Task 1) consider the following specification of *por*, sometimes called "parallel or":

$$\begin{array}{lcl} \textit{por}\ \mathsf{true}\ e & \simeq & \mathsf{true} \\ \textit{por}\ e\ \mathsf{true} & \simeq & \mathsf{true} \\ \textit{por}\ \mathsf{false}\ \mathsf{false} & \simeq & \mathsf{false} \end{array}$$

where $e_1 \simeq e_2$ is Kleene equality as in Tasks 1 and 2.

1. We cannot define a *function* $\textit{por} : \mathsf{bool} \to (\mathsf{bool} \to \mathsf{bool})$ in our language with this behavior. Prove that it is indeed impossible.

2. We also cannot translate expressions *por* $e_1 \; e_2$ into our language so that the result satisfies the given properties (which you do not need to prove). Instead consider adding a new primitive form of expression $\mathsf{por}\ e_1\ e_2$ to our language.

   (a) Give one or more typing rules for $\mathsf{por}\ e_1\ e_2$.

   (b) Provide one or more evaluation rules for $\mathsf{por}\ e_1\ e_2$ so that it satisfies the given specification and, furthermore, such that preservation, canonical forms, and progress continue to hold.

   (c) Show the new case(s) in the preservation theorem.

   (d) Show the new case(s) in the progress theorem.

   (e) Do your rules satisfy single-step determinacy (see Exercise L6.1)? If not, provide a counterexample. If yes, just indicate that it is the case (you do not need to prove it).