Principles of Programming Languages 2022 Final Exam Key

1 FbAb

a

(* ... Fb rules ... *)

$$\textsc{Abort}\ \frac{}{\texttt{Abort} \Rightarrow \texttt{Abort}}$$

$$\textsc{Abort And Left}\ \frac{e_1 \Rightarrow \texttt{Abort}}{e_1\,\texttt{And}\,e_2 \Rightarrow \texttt{Abort}} \quad \textsc{Abort And Right}\ \frac{e_1 \Rightarrow v \qquad e_2 \Rightarrow \texttt{Abort}}{e_1\,\texttt{And}\,e_2 \Rightarrow \texttt{Abort}}$$

$$\textsc{Abort Or Left}\ \frac{e_1 \Rightarrow \texttt{Abort}}{e_1\,\texttt{Or}\,e_2 \Rightarrow \texttt{Abort}} \quad \textsc{Abort Or Right}\ \frac{e_1 \Rightarrow v \qquad e_2 \Rightarrow \texttt{Abort}}{e_1\,\texttt{Or}\,e_2 \Rightarrow \texttt{Abort}}$$

$$\textsc{Abort Plus Left}\ \frac{e_1 \Rightarrow \texttt{Abort}}{e_1 + e_2 \Rightarrow \texttt{Abort}} \quad \textsc{Abort Plus Right}\ \frac{e_1 \Rightarrow v_1 \qquad e_2 \Rightarrow \texttt{Abort}}{e_1 + e_2 \Rightarrow \texttt{Abort}}$$

$$\textsc{Abort Minus Left}\ \frac{e_1 \Rightarrow \texttt{Abort}}{e_1 - e_2 \Rightarrow \texttt{Abort}} \quad \textsc{Abort Minus Right}\ \frac{e_1 \Rightarrow v_1 \qquad e_2 \Rightarrow \texttt{Abort}}{e_1 - e_2 \Rightarrow \texttt{Abort}}$$

$$\textsc{Abort Equals Left}\ \frac{e_1 \Rightarrow \texttt{Abort}}{e_1 = e_2 \Rightarrow \texttt{Abort}} \quad \textsc{Abort Equals Right}\ \frac{e_1 \Rightarrow v_1 \qquad e_2 \Rightarrow \texttt{Abort}}{e_1 = e_2 \Rightarrow \texttt{Abort}}$$

$$\textsc{Abort If Cond}\ \frac{e_1 \Rightarrow \texttt{Abort}}{\texttt{If}\,e_1\,\texttt{Then}\,e_2\,\texttt{Else}\,e_3 \Rightarrow \texttt{Abort}} \qquad \textsc{Abort Application Left}\ \frac{e_1 \Rightarrow \texttt{Abort}}{e_1\ e_2 \Rightarrow \texttt{Abort}}$$

$$\textsc{Abort Application Right}\ \frac{e_1 \Rightarrow \texttt{Function}\,x\,\texttt{->}\,e \qquad e_2 \Rightarrow \texttt{Abort}}{e_1\ e_2 \Rightarrow \texttt{Abort}}$$

$$\textsc{Abort Let}\ \frac{e_1 \Rightarrow \texttt{Abort}}{\texttt{Let}\,x = e_1\,\texttt{In}\,e_2 \Rightarrow \texttt{Abort}}$$

Note that we don't need to specify additional rules for If True and If False because the existing rules suffice to abort the evaluation.

b

Yes, because `Abort` alters the flow of the execution and can introduce nondeterminism to programs expecting an `Abort` (or not). This is very similar to the case of exceptions.

c

$e \cong e'$ iff for all contexts C such that $C[e]$ and $C[e']$ are both closed expressions, $C[e] \Rightarrow v$ for some v iff $C[e'] \Rightarrow v'$ for some v' or $C[e] \Rightarrow \texttt{Abort}$ iff $C[e'] \Rightarrow \texttt{Abort}$.

d

The pure functional law no longer holds. Consider the following counterexample:

$$\texttt{Let}\ a = (0\ 0)\ \texttt{In}\ \texttt{Let}\ b = \texttt{Abort}\ \texttt{In}\ e \not\cong \texttt{Let}\ b = \texttt{Abort}\ \texttt{In}\ \texttt{Let}\ a = (0\ 0)\ \texttt{In}\ e$$

2 Mutable list

a

```
type 'a list = Mt | Cons of 'a ref * 'a list
```

b

```
type 'a linked_list = Mt | Cons of 'a * ('a linked_list ref)
```

c

```
let append l1 l2 =
  let rec append_aux l l1 l2 =
    match l1 with
    | Mt -> l2
    | Cons (_, xs) ->
      match !xs with
      | Mt -> xs := l2; l
      | _ -> append_aux l !xs l2
  in
  append_aux l1 l1 l2
```

3 AFbV

a

$$\textsc{Create}\ \frac{e_1 \overset{S}{\Longrightarrow} v_1 \qquad e_2 \overset{S'}{\Longrightarrow} v_2 \qquad v_1\ a\ v_2 \overset{S''}{\Longrightarrow} v_3}{\texttt{Create}\ (e_1, e_2) \xrightarrow{S \cup S' \cup S'' \cup \{\langle a, v_3 \rangle\}} a, \text{for } a \text{ a fresh actor name}}$$

b

This self-notice is necessary because when a message sender needs a reply from the receiver, it needs to know its own address before sending it to the receiver. This also applies when an actor creates another actor; the one created knowing who created it is necessary for practical collaborative tasks.

Further, knowing its own address enables an actor to send messages to itself, which can be useful in certain scenarios, like yielding control to other tasks while performing some intensive computation.

4 EFbS

a

(* ... EFb rules ... *)

$$\textsc{Ref}\ \frac{\Gamma \vdash e : \tau \backslash E}{\Gamma \vdash \texttt{Ref}\, e : \tau\, \texttt{Ref}\, \backslash E} \qquad \textsc{Get}\ \frac{\Gamma \vdash e : \tau \backslash E}{\Gamma \vdash !e : \alpha \backslash E \cup \{\tau = \alpha\, \texttt{Ref}\}}$$

$$\textsc{Set}\ \frac{\Gamma \vdash e : \tau \backslash E \qquad \Gamma \vdash e' : \tau' \backslash E'}{\Gamma \vdash e := e' : \alpha \backslash E \cup E' \cup \{\tau = \alpha\, \texttt{Ref}\, , \tau' = \alpha\}}$$

b

(* ... EFb set closure algorithm ... *)

For each equation of the form $\alpha\, \texttt{Ref} = \alpha'\, \texttt{Ref}$, add $\alpha = \alpha'$ to the set.

c

(* ... EFb consistency checks ... *)

No immediate inconsistencies like $\alpha\, \texttt{Ref} = \mathit{Int}$, $\alpha\, \texttt{Ref} = \mathit{Bool}$, or $\alpha\, \texttt{Ref} = \tau\, \texttt{->}\, \tau'$.

Note that we don't need a clause for checking cases of $\alpha\, \texttt{Ref} = \alpha'\, \texttt{Ref}$ where $\alpha \neq \alpha'$ because $\alpha = \alpha'$ would have been added to the set when performing the closure and checked via existing clauses.

5 ANF

We prove $f\ (x+5) - 2 \cong \texttt{Let}\, x_1 = x + 5\, \texttt{In}\, \texttt{Let}\, x_2 = f\ x_1\, \texttt{In}\, \texttt{Let}\, x_3 = x_2 - 2\, \texttt{In}\, x_3$ via the following:

By operation execution ordering, we have

$$f\ (x+5) - 2 \cong \texttt{Let}\, x_1 = f\ (x+5)\, \texttt{In}\, \texttt{Let}\, x_2 = 2\, \texttt{In}\, x_1 - x_2 \tag{1}$$

By the pure functional law, we have

$$\begin{aligned} &\texttt{Let}\, x_1 = f\ (x+5)\, \texttt{In}\, \texttt{Let}\, x_2 = 2\, \texttt{In}\, x_1 - x_2 \\ \cong\ &\texttt{Let}\, x_2 = 2\, \texttt{In}\, \texttt{Let}\, x_1 = f\ (x+5)\, \texttt{In}\, x_1 - x_2 \end{aligned} \tag{2}$$

By Let-β, we have

$$\begin{aligned}&\texttt{Let}\,x_2 = 2\,\texttt{In}\,\texttt{Let}\,x_1 = f\ (x+5)\,\texttt{In}\,x_1 - x_2\\ &\cong (\texttt{Let}\,x_1 = f\ (x+5)\,\texttt{In}\,x_1 - x_2)[2/x_2]\\ &= \texttt{Let}\,x_1 = f\ (x+5)\,\texttt{In}\,x_1 - 2\end{aligned} \tag{3}$$

By Transitivity on (1), (2), and (3), we have

$$f\ (x+5) - 2 \cong \texttt{Let}\,x_1 = f\ (x+5)\,\texttt{In}\,x_1 - 2 \tag{4}$$

Again, by operation execution ordering, we have

$$f\ (x+5) \cong \texttt{Let}\,x_1 = f\,\texttt{In}\,\texttt{Let}\,x_2 = x+5\,\texttt{In}\,x_1\ x_2 \tag{5}$$

By Let-β, we have

$$\begin{aligned}&\texttt{Let}\,x_1 = f\,\texttt{In}\,\texttt{Let}\,x_2 = x+5\,\texttt{In}\,x_1\ x_2\\ &\cong (\texttt{Let}\,x_2 = x+5\,\texttt{In}\,x_1\ x_2)[f/x_1]\\ &= \texttt{Let}\,x_2 = x+5\,\texttt{In}\,f\ x_2\end{aligned} \tag{6}$$

By Transitivity on (5) and (6), we have

$$f\ (x+5) \cong \texttt{Let}\,x_2 = x+5\,\texttt{In}\,f\ x_2 \tag{7}$$

By Congruence in (7), we have

$$\begin{aligned}&\texttt{Let}\,x_1 = f\ (x+5)\,\texttt{In}\,x_1 - 2\\ &\cong \texttt{Let}\,x_1 = (\texttt{Let}\,x_2 = x+5\,\texttt{In}\,f\ x_2)\,\texttt{In}\,x_1 - 2\end{aligned} \tag{8}$$

By Associativity, we have

$$\begin{aligned}&\texttt{Let}\,x_1 = (\texttt{Let}\,x_2 = x+5\,\texttt{In}\,f\ x_2)\,\texttt{In}\,x_1 - 2\\ &\cong \texttt{Let}\,x_2 = x+5\,\texttt{In}\,\texttt{Let}\,x_1 = f\ x_2\,\texttt{In}\,x_1 - 2\end{aligned} \tag{9}$$

By renaming variables via Let-α (steps omitted), we have

$$\begin{aligned}&\texttt{Let}\,x_2 = x+5\,\texttt{In}\,\texttt{Let}\,x_1 = f\ x_2\,\texttt{In}\,x_1 - 2\\ &\cong \texttt{Let}\,x_1 = x+5\,\texttt{In}\,\texttt{Let}\,x_2 = f\ x_1\,\texttt{In}\,x_2 - 2\end{aligned} \tag{10}$$

By Return, we have

$$\texttt{Let}\, x_3 = x_2 - 2\, \texttt{In}\, x_3 \cong x_2 - 2 \tag{11}$$

By Symmetry on (11), we have

$$x_2 - 2 \cong \texttt{Let}\, x_3 = x_2 - 2\, \texttt{In}\, x_3 \tag{12}$$

By Congruence in (10), we have

$$\begin{gathered}\texttt{Let}\, x_1 = x + 5\, \texttt{In}\, \texttt{Let}\, x_2 = f\ x_1\, \texttt{In}\, x_2 - 2 \\ \cong \texttt{Let}\, x_1 = x + 5\, \texttt{In}\, \texttt{Let}\, x_2 = f\ x_1\, \texttt{In}\, \texttt{Let}\, x_3 = x_2 - 2\, \texttt{In}\, x_3\end{gathered}$$

By Transitivity on (4) through (12), we have proven

$$f\ (x + 5) - 2 \cong \texttt{Let}\, x_1 = x + 5\, \texttt{In}\, \texttt{Let}\, x_2 = f\ x_1\, \texttt{In}\, \texttt{Let}\, x_3 = x_2 - 2\, \texttt{In}\, x_3$$

6 STFbR++

a

(* ... STFbR rules ... *)

$$\textsc{Sub-Record-Int}\ \frac{}{\vdash \{l_1 : \tau_1; ...; ln : \tau_n\} <: \texttt{Int}}$$

b

$$\frac{\dfrac{}{\emptyset \vdash 4 : \texttt{Int}} \quad \dfrac{\dfrac{\dfrac{}{\emptyset \vdash 5 : \texttt{Int}} \quad \dfrac{}{\emptyset \vdash \texttt{True} : \texttt{Bool}}}{\emptyset \vdash \{a=5; b=\texttt{True}\} : \{a : \texttt{Int}; b : \texttt{Bool}\}} \quad \dfrac{}{\vdash \{a : \texttt{Int}; b : \texttt{Bool}\} <: \texttt{Int}}}{\{a=5; b=\texttt{True}\} : \texttt{Int}}}{\emptyset \vdash 4 + \{a = 5; b = \texttt{True}\} : \texttt{Int}}$$