CSE526: Principles of Programming Languages

Scott Stoller

hw5: shared-variable concurrency, lambda calculus

version: 5pm,5mar2004

due: 11mar2004

Answer

16th March 2004

1 Problem 1 (10pt)

1.1 Problem

Consider adding a new command CU (”conditional update”) to the programming language of chapter 8. The syntax of CU is:

$$< comm >::= CU(< var >, < var >, < intexp >, < var >)$$

Informally, the semantics of CU(x,x0,e,flag) is that it executes the following two atomic steps.

(1) evaluate the expression e to an integer i

(2) if $x = x_0$ then $(x := i; flag := 1)$ else $flag := 0$

Each of these two steps executes atomically. Transitions of other threads may occur between the steps.

Extend the transition semantics of section 8.1 with transition rule(s) for CU. Hint: One issue is where to store the value k between the two steps. One possible approach is to augment the variety of configurations by introducing an additional kind of command.

1.2 Answer without crit

$$\begin{array}{rcl} < comm > & ::= & CU(< var >, < var >, < intexp >, < var >) \\ & & |CU2(< var >, < var >, < intcfm >, < var >) \end{array}$$

$$\frac{}{< CU(x, x_0, e, flag), \sigma > \longrightarrow < CU2(x, x_0, i, flag), \sigma >} \quad where\ i = [[e]]_{intexp}\sigma$$

$$\frac{}{< CU2(x, x_0, i, flag), \sigma > \longrightarrow [\sigma]x : i|flag : 1]} \quad if\ \sigma x = \sigma x_0$$

$$\frac{}{< CU2(x, x_0, i, flag), \sigma > \longrightarrow [\sigma|flag : 0]} \quad if\ \sigma x \neq \sigma x_0$$

1.3 Answer with crit

$$< comm >::= CU(< var >, < var >, < intexp >, < var >)$$

$$\frac{}{\begin{array}{c} < CU(x, x_0, e, flag), \sigma > \longrightarrow \\ < \textbf{newvar}\ k := 0\ \textbf{in}\ (\textbf{crit}\ (k := e); \textbf{crit}\ (\textbf{if}\ x = x_0\ \textbf{then}\ (x := k; flag := 1)\ \textbf{else}\ flag := 0)), \sigma > \end{array}}$$

2 Problem 2

2.1 10.1 (a) (10pt)

$$
\begin{array}{ll}
 & (\lambda d.dd)(\lambda f.\lambda x.f(fx)) \\
\rightarrow & ((\lambda f.\lambda x.f(fx))(\lambda f.\lambda x.f(fx)) \\
\rightarrow & \lambda x.(\lambda f.\lambda x.f(fx))((\lambda f.\lambda x.f(fx))x) \\
\rightarrow & \lambda x.\lambda x_1.(\lambda f.\lambda x.f(fx))x((\lambda f.\lambda x.f(fx))x\ x_1) \\
\rightarrow & \lambda x.\lambda x_1.(\lambda x2.x(x\ x2))((\lambda f.\lambda x.f(fx))x\ x_1) \\
\rightarrow & \lambda x.\lambda x_1.x(x((\lambda f.\lambda x.f(fx))x\ x_1)) \\
\rightarrow & \lambda x.\lambda x_1.x(x((\lambda x5.x(x\ x5))x_1)) \\
\rightarrow & \lambda x.\lambda x_1.x(x(x(x\ x_1))) \\
\rightarrow & \lambda x.\lambda x_1.x(x(x(x\ x_1)))
\end{array}
$$

2.2 10.1 (b) (10pt)

The reduction sequence would stop at the first canonical form:

$$\lambda x.(\lambda f.\lambda x.f(fx))((\lambda f.\lambda x.f(fx))x)$$

2.3 10.1 (e) (10pt)

$$
\begin{array}{c}
(\lambda d.dd)(\lambda f.\lambda x.f(fx)) \\
\lambda d.dd \Rightarrow_E \lambda d.dd \\
\lambda f.\lambda x.f(fx) \Rightarrow_E \lambda f.\lambda x.f(fx) \\
(\lambda f.\lambda x.f(fx))(\lambda f.\lambda x.f(fx)) \\
\lambda f.\lambda x.f(fx) \Rightarrow_E \lambda f.\lambda x.f(fx) \\
\lambda f.\lambda x.f(fx) \Rightarrow_E \lambda f.\lambda x.f(fx) \\
\lambda x.(\lambda f.\lambda x.f(fx))((\lambda f.\lambda x.f(fx))x) \Rightarrow_E \lambda x.(\lambda f.\lambda x.f(fx))((\lambda f.\lambda x.f(fx))x) \\
\Rightarrow_E \lambda x.((\lambda f.\lambda x.f(fx))(\lambda f.\lambda x.f(fx))x) \\
\Rightarrow_E \lambda x.((\lambda f.\lambda x.f(fx))(\lambda f.\lambda x.f(fx))x)
\end{array}
$$