## 9 Semantics of Programming Languages (PMS)

Consider the concurrent imperative language L with syntax and conventional operational semantics as below.

$$\textit{statement},\ s ::= \ \mathbf{skip} \mid x := e; s \mid \mathbf{let}\, r = x\, \mathbf{in}\, s \mid \mathbf{let}\, r = op(e_1, ..., e_n)\, \mathbf{in}\, s \mid \mathbf{if}\ (e_1 = e_2)\ s\ \mathbf{else}\ s'$$

$$\textit{expression},\ e ::= \ r \mid v$$

$$\textit{process},\ p ::= \ \textit{tid}{:}s \mid p|p'$$

$$\textit{label},\ l ::= \ \mathbf{W}x{=}v \mid \mathbf{R}x{=}v \mid \tau \mid \textit{tid}{:}l \mid \mathbf{L}x$$

Here $x$ and $r$ range over shared and thread-local variables, $op$ over built-in operators, $v$ over values $0, 1, \ldots$, $\textit{tid}$ over thread ids $a, b, \ldots$. Let $m$ range over memory states, functions from shared variables to values. In the **let**s, $r$ binds in $s$.

$\boxed{s \xrightarrow{l} s'}$

$$\frac{}{x := v; s \xrightarrow{\mathbf{W}x=v} s}\ \text{WR} \qquad \frac{}{\mathbf{let}\, r = x\, \mathbf{in}\, s \xrightarrow{\mathbf{R}x=v} \{v/r\}s}\ \text{RD} \qquad \frac{}{\mathbf{if}\ (v = v)\ s\ \mathbf{else}\ s' \xrightarrow{\tau} s}\ \text{IF1}$$

$$\frac{v \neq v'}{\mathbf{if}\ (v = v')\ s\ \mathbf{else}\ s' \xrightarrow{\tau} s'}\ \text{IF2} \qquad \frac{v = [\![op]\!](v_1, \ldots, v_n)}{\mathbf{let}\, r = op(v_1, ..., v_n)\, \mathbf{in}\, s \xrightarrow{\tau} \{v/r\}s}\ \text{OP}$$

$\boxed{p \xrightarrow{l} p'}$

$$\frac{s \xrightarrow{l} s'}{\textit{tid}{:}s \xrightarrow{\textit{tid}:l} \textit{tid}{:}s'}\ \text{THREAD} \qquad \frac{p_1 \xrightarrow{l} p_1'}{p_1|p_2 \xrightarrow{l} p_1'|p_2}\ \text{PAR1} \qquad \frac{p_2 \xrightarrow{l} p_2'}{p_1|p_2 \xrightarrow{l} p_1|p_2'}\ \text{PAR2}$$

$\boxed{p, m \xrightarrow{l} p', m'}$

$$\frac{p \xrightarrow{\textit{tid}:\mathbf{W}x=v} p'}{p, m \xrightarrow{\textit{tid}:\mathbf{W}x=v} p', m \oplus \{x \mapsto v\}}\ \textsc{Swr} \qquad \frac{m(x) = v \quad p \xrightarrow{\textit{tid}:\mathbf{R}x=v} p'}{p, m \xrightarrow{\textit{tid}:\mathbf{R}x=v} p', m}\ \textsc{Srd} \qquad \frac{p \xrightarrow{\textit{tid}:\tau} p'}{p, m \xrightarrow{\textit{tid}:\tau} p', m}\ \textsc{Stau}$$

Say $p, m$ has a *data race* if there is a sequence of transitions $p, m \xrightarrow{l_1} \ldots \xrightarrow{l_n} \xrightarrow{l} \xrightarrow{l'}$ where $l$ and $l'$ *conflict*: they are reads or writes to the same location, at least one is a write, and they are by different threads.

**[continued ...]**

(*a*) Give a $p$ for which $p, m_0$ has a data race. [1 mark]

(*b*) A *vector clock* $c$ is a function from thread ids to natural numbers, identifying the $c(tid)$'th transition of each thread $tid$. Modify the semantics above to add a vector clock $c$ to each process thread ($tid_c{:}s$), each process label ($tid_c{:}l$), and each memory location (with each $m(x)$ now being a pair $v_c$ of a value and vector clock). In your semantics each vector clock should be computed so as to record the latest transition number of all threads that have causally affected that point. Explain your semantics, perhaps with some simple examples. [11 marks]

(*c*) Suppose that $p, m \xrightarrow{l} \xrightarrow{l_1} \ldots \xrightarrow{l_n} \xrightarrow{l'}$ in your vector-clock semantics, where $l$ and $l'$ conflict but are separated by $l_1, \ldots, l_n$. To implement a dynamic race detector, we would like to find conditions on $l_1, \ldots, l_n$ under which there is some other execution with $l$ and $l'$ adjacent: $p, m \xrightarrow{\hat{l}_1} \ldots \xrightarrow{\hat{l}_{\hat{n}}} \xrightarrow{\bar{l}} \xrightarrow{\bar{l}'}$ (where $\bar{l}$ and $\bar{l}'$ are like $l$ and $l'$ but perhaps with different vector clocks). Give such a condition, as liberal as you can, and explain why it has that property. [8 marks]