Cyber-Physical Programming

TPC-2

Renato Neves

nevrenato@di.uminho.pt

It is often necessary to incorporate *wait calls* in whatever programming language we are working with. For example, we might wish to perform an action every n seconds such as reading from a velocity sensor. So let us consider the following simple, imperative programming language:

$$\texttt{Prog}(X) \ni \texttt{x} := \texttt{t} \mid \texttt{wait}_\texttt{n}(\texttt{p}) \mid \texttt{p} \,;\, \texttt{q} \mid \texttt{if}\ \texttt{b}\ \texttt{then}\ \texttt{p}\ \texttt{else}\ \texttt{q} \mid \texttt{while}\ \texttt{b}\ \texttt{do}\ \{\ \texttt{p}\ \}$$

Note that $\texttt{t}$ is a *linear term* (defined in the previous lectures) and $\texttt{n}$ (in the program construct $\texttt{wait}_\texttt{n}$) is a natural number. The program $\texttt{wait}_\texttt{n}(\texttt{p})$ reads as "wait n seconds and then run program $\texttt{p}$". For such a language we take a semantics $\langle \texttt{p}, \sigma \rangle \Downarrow n, \sigma'$ which informs not only of the output of $\texttt{p}$ (i.e. σ') but also its execution time (i.e. n). Specifically, we adopt the following semantic rules:

$$\frac{\langle \texttt{t}, \sigma \rangle \Downarrow \texttt{r}}{\langle \texttt{x} := \texttt{t}, \sigma \rangle \Downarrow 0, \sigma[\texttt{r}/\texttt{x}]}\ (\text{asg}) \qquad \frac{\langle \texttt{p}, \sigma \rangle \Downarrow n, \sigma'}{\langle \texttt{wait}_\texttt{m}(\texttt{p}), \sigma \rangle \Downarrow m + n, \sigma'}\ (\text{wait})$$

$$\frac{\langle \texttt{p}, \sigma \rangle \Downarrow n, \sigma' \qquad \langle \texttt{q}, \sigma' \rangle \Downarrow m, \sigma''}{\langle \texttt{p} \,;\, \texttt{q}, \sigma \rangle \Downarrow n + m, \sigma''}\ (\text{seq})$$

$$\frac{\langle \texttt{b}, \sigma \rangle \Downarrow \texttt{tt} \qquad \langle \texttt{p}, \sigma \rangle \Downarrow n, \sigma'}{\langle \texttt{if}\ \texttt{b}\ \texttt{then}\ \texttt{p}\ \texttt{else}\ \texttt{q}, \sigma \rangle \Downarrow n, \sigma'}\ (\text{if}_1) \qquad \frac{\langle \texttt{b}, \sigma \rangle \Downarrow \texttt{ff} \qquad \langle \texttt{q}, \sigma \rangle \Downarrow n, \sigma'}{\langle \texttt{if}\ \texttt{b}\ \texttt{then}\ \texttt{p}\ \texttt{else}\ \texttt{q}, \sigma \rangle \Downarrow n, \sigma'}\ (\text{if}_2)$$

$$\frac{\langle \texttt{b}, \sigma \rangle \Downarrow \texttt{tt} \qquad \langle \texttt{p}, \sigma \rangle \Downarrow n, \sigma' \qquad \langle \texttt{while}\ \texttt{b}\ \texttt{do}\ \{\ \texttt{p}\ \}, \sigma' \rangle \Downarrow m, \sigma''}{\langle \texttt{while}\ \texttt{b}\ \texttt{do}\ \{\ \texttt{p}\ \}, \sigma \rangle \Downarrow n + m, \sigma''}\ (\text{wh}_1)$$

$$\frac{\langle \texttt{b}, \sigma \rangle \Downarrow \texttt{ff}}{\langle \texttt{while}\ \texttt{b}\ \texttt{do}\ \{\ \texttt{p}\ \}, \sigma \rangle \Downarrow 0, \sigma}\ (\text{wh}_2)$$

We can then define a natural notion of *equivalence* for our programs: we say that two programs $\texttt{p}$ and $\texttt{q}$ are equivalent (in symbols, $\texttt{p} \sim \texttt{q}$) if for all environments σ we have,

$$\langle \texttt{p}, \sigma \rangle \Downarrow n, \sigma' \text{ iff } \langle \texttt{q}, \sigma \rangle \Downarrow n, \sigma'$$

Exercise 1. Implement in `Haskell` the while-language described above and its semantics. Suggestion: use the code developed in the previous lectures.

Exercise 2 (***)**.** Prove that $\texttt{wait}_\texttt{n}(\texttt{wait}_\texttt{m}(\texttt{p})) \sim \texttt{wait}_{\texttt{n+m}}(\texttt{p})$. Can you think of (and prove) other interesting equivalences? Note that the more equivalences a compiler knows the more ways it has of performing optimisations.

What to submit: An `.hs` file containing the code that you developed (properly commented!) for the first exercise. A `.pdf` file containing the solution to the second exercise. Please send a corresponding `.zip` archive by email (`nevrenato@di.uminho.pt`) with the name "`cpp2324-N.zip`", where "`N`" is your student number. The subject of the email should be "`cpp2324 N TPC-2`".