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 query a velocity sensor every n seconds. So let us consider the following simple, imperative programming language:

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

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

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

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

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

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

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

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

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

Exercise 1. Prove that $\mathtt{wait_n}(\mathtt{wait_m}(\mathtt{p})) \sim \mathtt{wait_{n+m}}(\mathtt{p})$.

Exercise 2. Implement in `Haskell` the while-language described above and its semantics. Suggestion: use the code developed in previous lectures.

What to submit: A `.pdf` file containing the solution to the first exercise and a also `.hs` file containing the code that you developed (properly commented!) for the second exercise. Please send a corresponding `.zip` archive by email (`nevrenato@di.uminho.pt`) with the name “`cpp2122-N.zip`”, where “`N`” is your student number. The subject of the email should be “`cpp2122 N TPC-2`”.