Semantics for (Hybrid) Programming

Renato Neves

Universidade do Minho

HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Table of Contents

Last Lectures

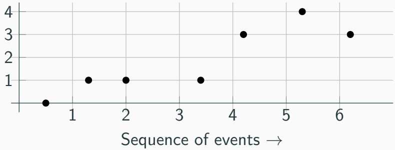
Explored a simple language (CCS) and its semantics

Used it to model and analyse communicating systems

Expanded our study to the timed setting, via UPPAAL

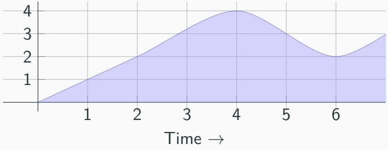
Used it to save us from zombies!

Going Beyond the Timed Setting



Described via classical methods of computation

+



Described via differential equations

Computational devices now interact with arbitrary physical processes (and not just time)

Which Language?

This time we explore a simple imperative language

No concurrency, no communication, and no higher-order func.

(languages with such features are still underdeveloped)

Perhaps some of you would like to improve them :-)

The Hybrid While-Language

Fix a stock of variables $X = \{\mathtt{x}_1, \dots, \mathtt{x}_n\}$. Then we have,

Linear Terms

$\mathtt{LTerm}(X) \ni \mathtt{r} \mid \mathtt{r} \cdot \mathtt{t} \mid \mathtt{x} \mid \mathtt{t} + \mathtt{s}$

real number

Atomic Programs

$\mathtt{At}(X) \ni \mathtt{x} := \mathtt{t} \mid \mathtt{x}'_1 = \mathtt{t}_1, \dots, \mathtt{x}'_n = \mathtt{t}_n \ \mathtt{for}\ \mathtt{t}$

"run" the system of differential equations for $\mathtt{t}$ seconds

Hybrid Programs

$\mathtt{Prog}(X) \ni \mathtt{a} \mid \mathtt{p} \,;\, \mathtt{q} \mid \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q} \mid \mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}$

Overview

First we tackle a while-language without differential equations and its semantics

Then we move to the hybrid case and see how the corresponding semantics helps the engineer to analyse hybrid programs

Throughout this journey, we will:

- write implementations in HASKELL
- do analyses in LINCE

Table of Contents

A Language of Linear Terms and its Semantics

Linear Terms

$\texttt{LTerm}(X) \ni \texttt{r} \mid \texttt{r} \cdot \texttt{t} \mid \texttt{x} \mid \texttt{t} + \texttt{s}$

Let $\sigma : X \to \mathbb{R}$ be an environment, i.e. a memory on which the program performs computations

The expression $\langle \texttt{t}, \sigma \rangle \Downarrow \texttt{r}$ tells that the linear expression $\texttt{t}$ outputs $\texttt{r}$ if the current memory is σ

$$\frac{}{\langle \texttt{x}, \sigma \rangle \Downarrow \sigma(\texttt{x})}\ (\text{var}) \qquad \frac{}{\langle \texttt{r}, \sigma \rangle \Downarrow \texttt{r}}\ (\text{con})$$

$$\frac{\langle \texttt{t}, \sigma \rangle \Downarrow \texttt{r}}{\langle \texttt{s} \cdot \texttt{t}, \sigma \rangle \Downarrow \texttt{s} \cdot \texttt{r}}\ (\text{scl}) \qquad \frac{\langle \texttt{t}_1, \sigma \rangle \Downarrow \texttt{r}_1 \qquad \langle \texttt{t}_2, \sigma \rangle \Downarrow \texttt{r}_2}{\langle \texttt{t}_1 + \texttt{t}_2, \sigma \rangle \Downarrow \texttt{r}_1 + \texttt{r}_2}\ (\text{add})$$

The Semantics at Work

The linear term $\mathtt{x} + 2 \cdot \mathtt{y}$ corresponds to the tree

$$\begin{tikzcd}
 & (+) \arrow[dl] \arrow[dr] & \\
\mathtt{x} & & (2\,\cdot) \arrow[d] \\
 & & \mathtt{y}
\end{tikzcd}$$

Consider an environment σ such that $\sigma(\mathtt{x}) = 3$ and $\sigma(\mathtt{y}) = 4$. We can then build the following derivation tree:

$$\dfrac{\langle \mathtt{x}, \sigma \rangle \Downarrow 3 \qquad\qquad \dfrac{\langle \mathtt{y}, \sigma \rangle \Downarrow 4}{\langle 2 \cdot \mathtt{y}, \sigma \rangle \Downarrow 8}}{\langle \mathtt{x} + 2 \cdot \mathtt{y}, \sigma \rangle \Downarrow 11}$$

Exercises

- $\langle 2 \cdot \mathtt{x} + 2 \cdot \mathtt{y}, \sigma \rangle \Downarrow$?
- $\langle 3 \cdot (2 \cdot \mathtt{x}) + 2 \cdot (\mathtt{y} + \mathtt{z}), \sigma \rangle \Downarrow$?

Exercises

- $\langle 2 \cdot \mathtt{x} + 2 \cdot \mathtt{y}, \sigma \rangle \Downarrow$?
- $\langle 3 \cdot (2 \cdot \mathtt{x}) + 2 \cdot (\mathtt{y} + \mathtt{z}), \sigma \rangle \Downarrow$?

Boring computations? If so why not implement the semantics in HASKELL?

Equivalence of Linear Terms

The previous semantics yields the following notion of equivalence:
$\mathtt{t} \sim \mathtt{s}$ if for all environments σ

$$\langle \mathtt{t}, \sigma \rangle \Downarrow \mathtt{r} \text{ iff } \langle \mathtt{s}, \sigma \rangle \Downarrow \mathtt{r}$$

Examples of equivalent terms:

- $\mathtt{r} \cdot (\mathtt{x} + \mathtt{y}) \sim \mathtt{r} \cdot \mathtt{x} + \mathtt{r} \cdot \mathtt{y}$
- $0 \cdot \mathtt{x} \sim 0$
- $(\mathtt{r} \cdot \mathtt{s}) \cdot \mathtt{x} \sim \mathtt{r} \cdot (\mathtt{s} \cdot \mathtt{x})$?

Table of Contents

A Language of Boolean Terms and its Semantics

Boolean Terms

$\mathtt{BTerm}(X) \ni \mathtt{t}_1 \leq \mathtt{t}_2 \mid \mathtt{b} \wedge \mathtt{c} \mid \neg \mathtt{b}$

A Language of Boolean Terms and its Semantics

Boolean Terms

$\mathtt{BTerm}(X) \ni \mathtt{t}_1 \leq \mathtt{t}_2 \mid \mathtt{b} \wedge \mathtt{c} \mid \neg \mathtt{b}$

The expression $\langle \mathtt{b}, \sigma \rangle \Downarrow \mathtt{v}$ says that the Boolean term $\mathtt{b}$ outputs $\mathtt{v}$ if the current memory is σ

$$\frac{\langle \mathtt{t}_1, \sigma \rangle \Downarrow \mathtt{r}_1 \qquad \langle \mathtt{t}_2, \sigma \rangle \Downarrow \mathtt{r}_2 \qquad \mathtt{r}_1 \leq \mathtt{r}_2}{\langle \mathtt{t}_1 \leq \mathtt{t}_2, \sigma \rangle \Downarrow \mathtt{tt}} \text{ (leq)}$$

$$\frac{\langle \mathtt{t}_1, \sigma \rangle \Downarrow \mathtt{r}_1 \qquad \langle \mathtt{t}_2, \sigma \rangle \Downarrow \mathtt{r}_2 \qquad \mathtt{r}_1 \not\leq \mathtt{r}_2}{\langle \mathtt{t}_1 \leq \mathtt{t}_2, \sigma \rangle \Downarrow \mathtt{ff}} \text{ (gtr)}$$

$$\frac{\langle \mathtt{b}, \sigma \rangle \Downarrow \mathtt{v}}{\langle \neg \mathtt{b}, \sigma \rangle \Downarrow \neg \mathtt{v}} \text{ (not)} \qquad \frac{\langle \mathtt{b}_1, \sigma \rangle \Downarrow \mathtt{v}_1 \qquad \langle \mathtt{b}_2, \sigma \rangle \Downarrow \mathtt{v}_2}{\langle \mathtt{b}_1 \wedge \mathtt{b}_2, \sigma \rangle \Downarrow \mathtt{v}_1 \wedge \mathtt{v}_2} \text{ (and)}$$

Table of Contents

While-Programs

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

$$\frac{\langle \mathtt{t}, \sigma \rangle \Downarrow \mathtt{r}}{\langle \mathtt{x} := \mathtt{t}, \sigma \rangle \Downarrow \sigma[\mathtt{r}/\mathtt{x}]}\ (\text{asg}) \qquad \frac{\langle \mathtt{p}, \sigma \rangle \Downarrow \sigma' \qquad \langle \mathtt{q}, \sigma' \rangle \Downarrow \sigma''}{\langle \mathtt{p} \mathbin{;} \mathtt{q}, \sigma \rangle \Downarrow \sigma''}\ (\text{seq})$$

$$\frac{\langle \mathtt{b}, \sigma \rangle \Downarrow \mathtt{tt} \qquad \langle \mathtt{p}, \sigma \rangle \Downarrow \sigma'}{\langle \mathbf{if}\ \mathtt{b}\ \mathbf{then}\ \mathtt{p}\ \mathbf{else}\ \mathtt{q}, \sigma \rangle \Downarrow \sigma'}\ (\text{if1}) \qquad \frac{\langle \mathtt{b}, \sigma \rangle \Downarrow \mathtt{ff} \qquad \langle \mathtt{q}, \sigma \rangle \Downarrow \sigma'}{\langle \mathbf{if}\ \mathtt{b}\ \mathbf{then}\ \mathtt{p}\ \mathbf{else}\ \mathtt{q}, \sigma \rangle \Downarrow \sigma'}\ (\text{if2})$$

$$\frac{\langle \mathtt{b}, \sigma \rangle \Downarrow \mathtt{tt} \qquad \langle \mathtt{p}, \sigma \rangle \Downarrow \sigma' \qquad \langle \mathbf{while}\ \mathtt{b}\ \mathbf{do}\ \{\ \mathtt{p}\ \}, \sigma' \rangle \Downarrow \sigma''}{\langle \mathbf{while}\ \mathtt{b}\ \mathbf{do}\ \{\ \mathtt{p}\ \}, \sigma \rangle \Downarrow \sigma''}\ (\text{wh1})$$

$$\frac{\langle \mathtt{b}, \sigma \rangle \Downarrow \mathtt{ff}}{\langle \mathbf{while}\ \mathtt{b}\ \mathbf{do}\ \{\ \mathtt{p}\ \}, \sigma \rangle \Downarrow \sigma}\ (\text{wh2})$$

The Semantics at Work

The program $\mathtt{x} := \mathtt{x} + 1 \, ; \mathtt{x} := \mathtt{x} + 2$ corresponds to the tree

$$\begin{tikzcd}[column sep=small] & (\, ; \,) \arrow[dl] \arrow[dr] & \\ \mathtt{x} := \mathtt{x} + 1 & & \mathtt{x} := \mathtt{x} + 2 \end{tikzcd}$$

Consider the environment $\sigma = x \mapsto 3$. We build the following derivation tree:

$$\dfrac{\dfrac{\langle \mathtt{x} + 1, \mathtt{x} \mapsto 3 \rangle \Downarrow 4}{\langle \mathtt{x} := \mathtt{x} + 1, \mathtt{x} \mapsto 3 \rangle \Downarrow \mathtt{x} \mapsto 4} \qquad \dfrac{\langle \mathtt{x} + 2, \mathtt{x} \mapsto 4 \rangle \Downarrow 6}{\langle \mathtt{x} := \mathtt{x} + 2, \mathtt{x} \mapsto 4 \rangle \Downarrow \mathtt{x} \mapsto 6}}{\langle \mathtt{x} := \mathtt{x} + 1 \, ; \mathtt{x} := \mathtt{x} + 2, \mathtt{x} \mapsto 3 \rangle \Downarrow \mathtt{x} \mapsto 6}$$

Exercise

- $\mathtt{x} := 0\,;\,\mathtt{y} := 1\,;\,\mathtt{while}\ \mathtt{x} \leq \mathtt{y}\ \mathtt{do}\ \{\mathtt{x} := \mathtt{x} + \mathtt{y}\,;\,\mathtt{y} := \mathtt{y} + 1\} \Downarrow\ ?$

Equivalence of While-Programs

The previous semantics yields the following notion of equivalence:
$\mathtt{p} \sim \mathtt{q}$ if for all environments σ

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

Examples of equivalent terms:

- $\mathtt{x} := \mathtt{x} + 1 \,;\, \mathtt{x} := \mathtt{x} + 2 \sim \mathtt{x} := \mathtt{x} + 3$
- $(\mathtt{p} \,;\, \mathtt{q}) \,;\, \mathtt{r} \sim \mathtt{p} \,;\, (\mathtt{q} \,;\, \mathtt{r})$

Pause for Meditations

We have just built and implemented our first progr. language

Note that we used its semantics to run our programs and also to prove properties about them

Which features would you like to add to this language next? Probabilistic operations or perhaps concurrency?

Next step: add differential operations