Hybrid Programming

Renato Neves

Universidade do Minho

HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Table of Contents

Last Lectures

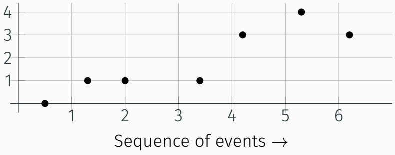
Explored a simple language (ccs) and its semantics

Used it in the design of communicating systems

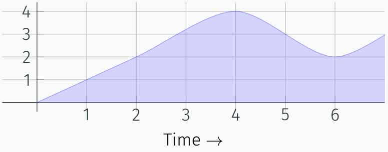
Expanded this study to the timed setting

Used results to save us all 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 and no communication

... languages with such features are still underdeveloped

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

The hybrid while-Language

Linear Terms

$\mathtt{LTerm} ::= \mathtt{r} \mid \mathtt{r} \cdot \mathtt{t} \mid \mathtt{x} \mid \mathtt{t} + \mathtt{s}$

real number (r) — variable (x)

Atomic Programs

$\mathtt{At} ::= \mathtt{x} := \mathtt{t} \mid \mathtt{x}_1' = \mathtt{t}_1, \ldots, \mathtt{x}_n' = \mathtt{t}_n \; \mathtt{for} \; \mathtt{t}$

"run" the system of differential equations for `t` seconds

Programs

$\mathtt{Prog} ::= \mathtt{a} \mid \mathtt{p} \; ; \; \mathtt{q} \mid \mathtt{if} \; \mathtt{b} \; \mathtt{then} \; \mathtt{p} \; \mathtt{else} \; \mathtt{q} \mid \mathtt{while} \; \mathtt{b} \; \mathtt{do} \; \{ \, \mathtt{p} \, \}$

Overview

We study a while-language without differential equations

Then move to the hybrid case and see how semantics aids in the analysis of 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

$\mathtt{LTerm} ::= \mathtt{r} \mid \mathtt{r} \cdot \mathtt{t} \mid \mathtt{x} \mid \mathtt{t} + \mathtt{s}$

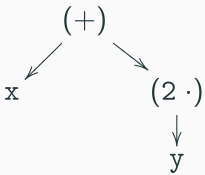
$\sigma : X \to \mathbb{R}$ denote a memory

$\langle \mathtt{t}, \sigma \rangle \Downarrow \mathtt{r}$ means that $\mathtt{t}$ outputs $\mathtt{r}$ if current memory is σ

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

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

$x + 2 \cdot y$ corresponds to the syntax tree



$\sigma(x) = 3$ and $\sigma(y) = 4$ yield the "semantic tree"

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

Exercises

Write down the corresponding derivation trees for

- $2 \cdot \mathrm{x} + 2 \cdot \mathrm{y}$
- $3 \cdot (2 \cdot \mathrm{x}) + 2 \cdot (\mathrm{y} + \mathrm{z})$

Exercises

Write down the corresponding derivation trees for

- $2 \cdot \mathtt{x} + 2 \cdot \mathtt{y}$
- $3 \cdot (2 \cdot \mathtt{x}) + 2 \cdot (\mathtt{y} + \mathtt{z})$

Boring computations? If so why not implement the semantics?

A language of Boolean terms and its semantics

Boolean terms

$\texttt{BTerm} ::= \texttt{t} \leq \texttt{t} \mid \texttt{b} \wedge \texttt{b} \mid \neg \texttt{b}$

$\langle \texttt{b}, \sigma \rangle \Downarrow \texttt{v}$ tells that $\texttt{b}$ outputs $\texttt{v}$ if the memory is σ

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

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

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

A while-language and its semantics

Programs

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

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

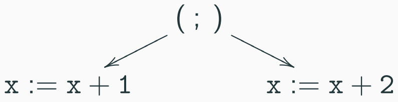
$$\frac{\langle \mathtt{b}, \sigma \rangle \Downarrow \mathtt{tt} \qquad \langle \mathtt{p}, \sigma \rangle \Downarrow \sigma'}{\langle \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q}, \sigma \rangle \Downarrow \sigma'}\ (\text{if1}) \qquad \frac{\langle \mathtt{b}, \sigma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

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

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

The semantics at work

$\mathtt{x} := \mathtt{x} + 1 \; ; \; \mathtt{x} := \mathtt{x} + 2$ corresponds to the 'syntax tree'



Memory $\sigma = x \mapsto 3$ yields the 'semantic tree'

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