Hybrid Programming

Renato Neves

Universidade do Minho

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

Table of Contents

Overview

Semantics

Design Patterns

Conclusions

Last Lectures

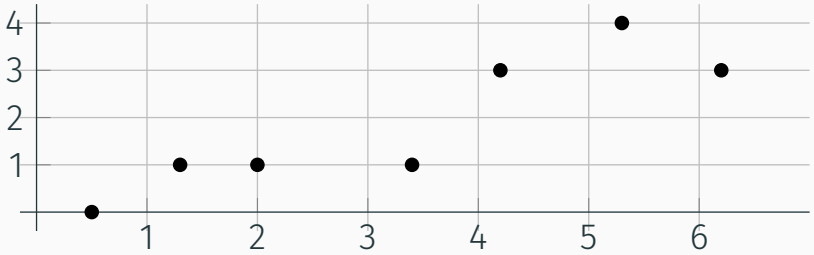
Explored a simple language (ccs) and its semantics

Used it in the design of communicating systems

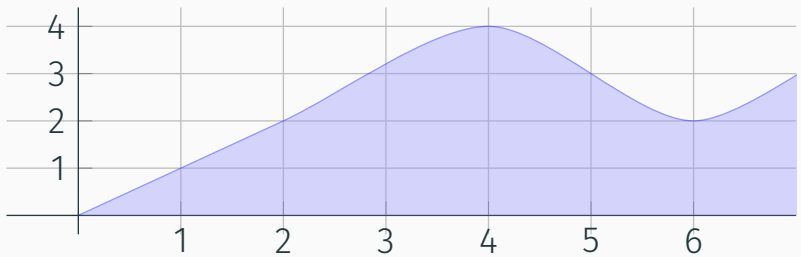
Expanded this study to the timed setting

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

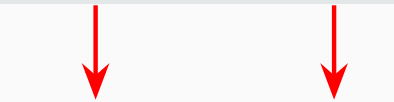
... languages with such features are still underdeveloped

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

The hybrid while-Language

Linear Terms

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



Atomic Programs

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

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

Programs

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

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

Overview

Semantics

Design Patterns

Conclusions

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

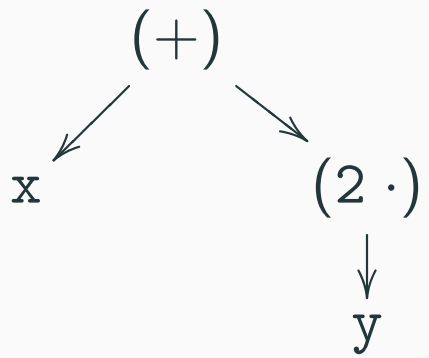
$\langle \mathtt{t}, \sigma \rangle \Downarrow \mathtt{r}$ means that $\mathtt{t}$ outputs $\mathtt{r}$ if current memory is σ

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

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

The semantics at work

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



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

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

Exercises

Write down the corresponding derivation trees for

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

Exercises

Write down the corresponding derivation trees for

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

Boring computations? If so why not implement the semantics?

A language of Boolean terms and its semantics

Boolean terms

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

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

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

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

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

A while-language and its semantics

Programs

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

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

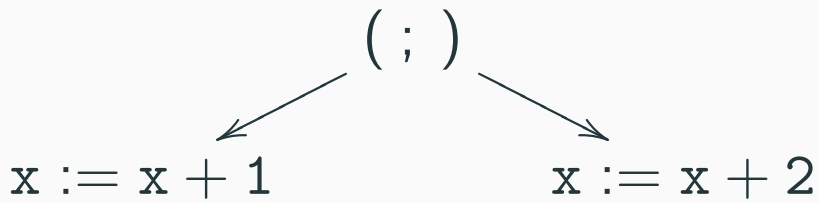
$$\frac{\langle \mathtt{b}, \sigma\rangle \Downarrow \mathtt{tt} \qquad \langle \mathtt{p}, \sigma\rangle \Downarrow \sigma'}{\langle \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q}, \sigma\rangle \Downarrow \sigma'}\ (\mathrm{if1}) \qquad \frac{\langle \mathtt{b}, \sigma\rangle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

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

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

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'

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

Pause for Meditations

Designed our first programming language

... and its semantics

Which program features would you like to add next ?

Pause for Meditations

Designed our first programming language

... and its semantics

Which program features would you like to add next ?

From our end we will add differential operations

Preliminaries

Systems of diff. eqs. $\mathtt{x_1'} = \mathtt{t_1}, \ldots, \mathtt{x_n'} = \mathtt{t_n}$ have unique solutions

$$\phi : \mathbb{R}^n \times [0, \infty) \longrightarrow \mathbb{R}^n$$

Obtained via Linear Algebra

Example (Continuous Dynamics of a Vehicle)

$\mathtt{p}' = \mathtt{v}, \mathtt{v}' = \mathtt{a}$ admits the solution

$$\phi\left((x_0, v_0), t\right) = \left(x_0 + v_0 t + \tfrac{1}{2} a t^2, v_0 + at\right)$$

Initial position and initial velocity

Conventions

Often abbreviate a list $v_1, \ldots, v_n$ to $\vec{v}$

$\sigma[\vec{v}/\vec{x}]$ denotes the memory that maps each x_i in $\vec{x}$ to v_i in $\vec{v}$ and all other variables the same way as σ

Example

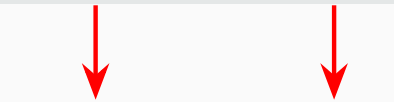
$$\sigma[v_1, v_2/x_1, x_2](y) = \begin{cases} v_1 & \text{if } y = x_1 \\ v_2 & \text{if } y = x_2 \\ \sigma(y) & \text{otherwise} \end{cases}$$

Often treat $\sigma : \{\mathtt{x_1}, \ldots, \mathtt{x_n}\} \to \mathbb{R}$ as a list $[\sigma(\mathtt{x_1}), \ldots, \sigma(\mathtt{x_n})]$

The hybrid while-language and ...

Linear Terms

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



Atomic Programs

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

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

Hybrid Programs

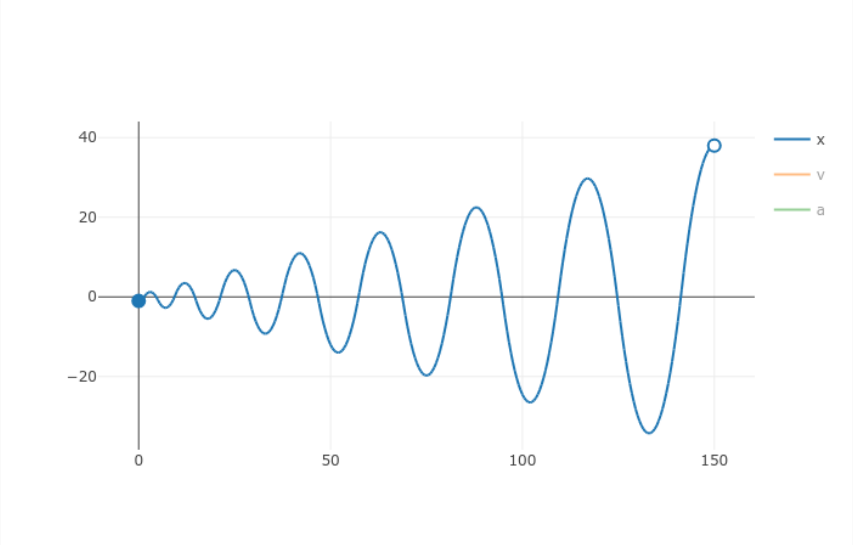
$$\mathtt{Prog} \ni \mathtt{a} \mid \mathtt{p}\,;\,\mathtt{q} \mid \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q} \mid \mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\,\mathtt{p}\,\}$$

... its semantics

Evaluation of programs is now time-dependent

$$\langle \mathrm{p}, \sigma, t \rangle \Downarrow \sigma'$$

LINCE relies on such a semantics: evaluation of $\langle \mathrm{p}, \sigma, t_i \rangle$ for a "big" sequence $t_1, \ldots, t_k$ yields a trajectory, such as



The semantic rules pt. I

$$\frac{\langle \mathtt{s}, \sigma \rangle \Downarrow \mathtt{r} \qquad t < \mathtt{r}}{\langle \vec{\mathtt{x}}' = \vec{\mathtt{t}}\ \mathtt{for}\ \mathtt{s}, \sigma, t \rangle \Downarrow \mathtt{stop}, \sigma[\phi(\sigma, t)/\vec{x}]}$$

$$\frac{\langle \mathtt{s}, \sigma \rangle \Downarrow \mathtt{r} \qquad t = \mathtt{r}}{\langle \vec{\mathtt{x}}' = \vec{\mathtt{t}}\ \mathtt{for}\ \mathtt{s}, \sigma, t \rangle \Downarrow \mathtt{skip}, \sigma[\phi(\sigma, t)/\vec{x}]}$$

$$\frac{\langle \mathtt{t}, \sigma \rangle \Downarrow \mathtt{r}}{\langle \mathtt{x} := \mathtt{t}, \sigma, 0 \rangle \Downarrow \mathtt{skip}, \sigma[\mathtt{r}/\mathtt{x}]} \qquad \frac{\langle \mathtt{p}, \sigma, t \rangle \Downarrow \mathtt{stop}, \sigma'}{\langle \mathtt{p}\ ;\ \mathtt{q}, \sigma, t \rangle \Downarrow \mathtt{stop}, \sigma'}$$

$$\frac{\langle \mathtt{p}, \sigma, t \rangle \Downarrow \mathtt{skip}, \sigma' \qquad \langle \mathtt{q}, \sigma, t' \rangle \Downarrow \mathtt{s}, \sigma''}{\langle \mathtt{p}\ ;\ \mathtt{q}, \sigma, t + t' \rangle \Downarrow \mathtt{s}, \sigma''}$$

Examples

$$\dfrac{\dfrac{\langle 1, (\mathtt{x} \mapsto 2)\rangle \Downarrow 1 \qquad \frac{1}{2} < 1}{\langle \mathtt{x}' = 0 \ \mathtt{for}\ 1, (\mathtt{x} \mapsto 2), \frac{1}{2}\rangle \Downarrow \mathtt{stop}, (\mathtt{x} \mapsto 2)}}{\langle (\mathtt{x}' = 0 \ \mathtt{for}\ 1)\ ;\ (\mathtt{x}' = 1 \ \mathtt{for}\ 1), (\mathtt{x} \mapsto 2), \frac{1}{2}\rangle \Downarrow \mathtt{stop}, (\mathtt{x} \mapsto 2)}$$

$$\downarrow$$

$$= (\mathtt{x} \mapsto 2)[\phi(2, \tfrac{1}{2})/\mathtt{x}]$$

$$\dfrac{\dfrac{\cdots}{\langle \mathtt{x}' = 0 \ \mathtt{for}\ 1, (\mathtt{x} \mapsto 2), 1\rangle \Downarrow \mathtt{skip}, (\mathtt{x} \mapsto 2)} \qquad \dfrac{\cdots}{\langle \mathtt{x}' = 1 \ \mathtt{for}\ 1, (\mathtt{x} \mapsto 2), \frac{1}{2}\rangle \Downarrow \mathtt{stop}, (\mathtt{x} \mapsto 2 + \frac{1}{2})}}{\langle (\mathtt{x}' = 0 \ \mathtt{for}\ 1)\ ;\ (\mathtt{x}' = 1 \ \mathtt{for}\ 1), (\mathtt{x} \mapsto 2), 1 + \frac{1}{2}\rangle \Downarrow \mathtt{stop}, (\mathtt{x} \mapsto 2 + \frac{1}{2})}$$

$$\downarrow$$

$$= (\mathtt{x} \mapsto 2)[\phi(2, \tfrac{1}{2})/\mathtt{x}] = (\mathtt{x} \mapsto 2)[2 + \tfrac{1}{2}/\mathtt{x}] = x \mapsto 2 + \tfrac{1}{2}$$

Exercise

Write down the corresponding derivation trees for

- $(\mathtt{x}' = 1 \texttt{ for } 1) \,;\, (\mathtt{x}' = -1 \texttt{ for } 1)$ at time instant $\frac{1}{2}$
- $(\mathtt{x}' = 1 \texttt{ for } 1) \,;\, (\mathtt{x}' = -1 \texttt{ for } 1)$ at time instant 2

The semantic rules pt. II

$$\frac{\langle \mathtt{b}, \sigma\rangle \Downarrow \mathtt{tt} \qquad \langle \mathtt{p}, \sigma, t\rangle \Downarrow \mathtt{s}, \sigma'}{\langle \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q}, \sigma, t\rangle \Downarrow \mathtt{s}, \sigma'} \qquad \frac{\langle \mathtt{b}, \sigma\rangle \Downarrow \mathtt{ff} \qquad \langle \mathtt{q}, \sigma, t\rangle \Downarrow \mathtt{s}, \sigma'}{\langle \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q}, \sigma, t\rangle \Downarrow \mathtt{s}, \sigma'}$$

$$\frac{\langle \mathtt{b}, \sigma\rangle \Downarrow \mathtt{tt} \qquad \langle \mathtt{p}\ ;\ \mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}, \sigma, t\rangle \Downarrow \mathtt{s}, \sigma'}{\langle \mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}, \sigma, t\rangle \Downarrow \mathtt{s}, \sigma'}$$

$$\frac{\langle \mathtt{b}, \sigma\rangle \Downarrow \mathtt{ff}}{\langle \mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}, \sigma, 0\rangle \Downarrow \mathtt{skip}, \sigma}$$

A Zoo of Newtonian Hybrid Programs

- Cruise controller (speed regulation)
- Landing system
- Bouncing Ball
- Moving a particle from point A to B
- Following a leader

Table of Contents

Overview

Semantics

Design Patterns

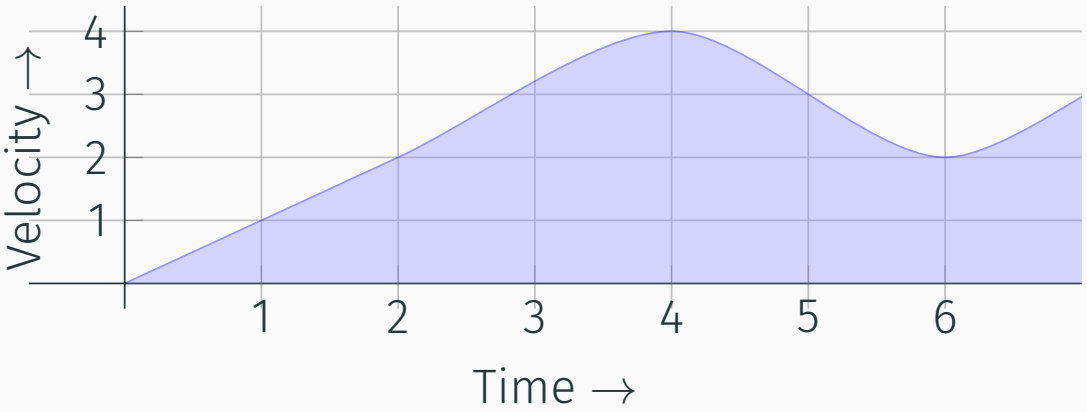
Conclusions

A selection of design patterns

We explore the last two (ubiquituous) scenarios

Tackle them via Analytic Geometry

Moving a particle

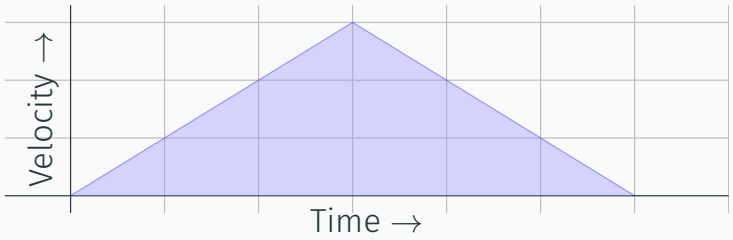


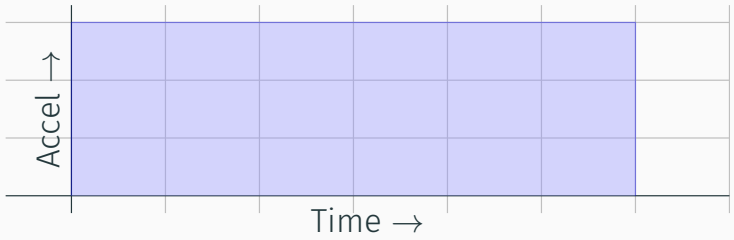
→ Area = distance travelled

What should be the function's shape?

Moving a particle with a fixed acceleration

We accelerate and then brake

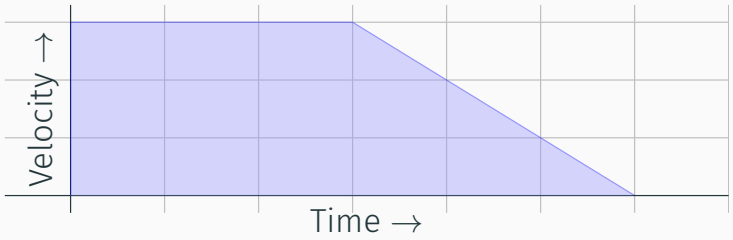


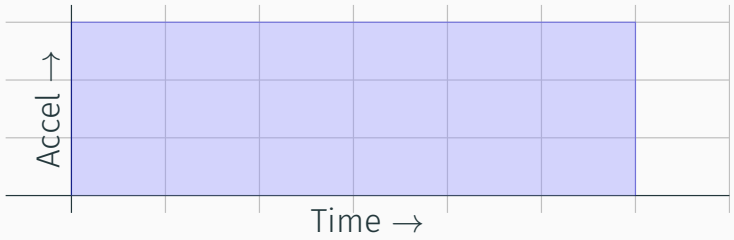


$$\begin{cases} \text{dist} = \frac{1}{2} \cdot b \cdot h \\ h = \frac{1}{2} \cdot b \cdot \text{accel} \end{cases} \implies b = \sqrt{\frac{4 \cdot \text{dist}}{\text{accel}}}$$

Moving a particle with positive velocity

We maintain velocity and then brake

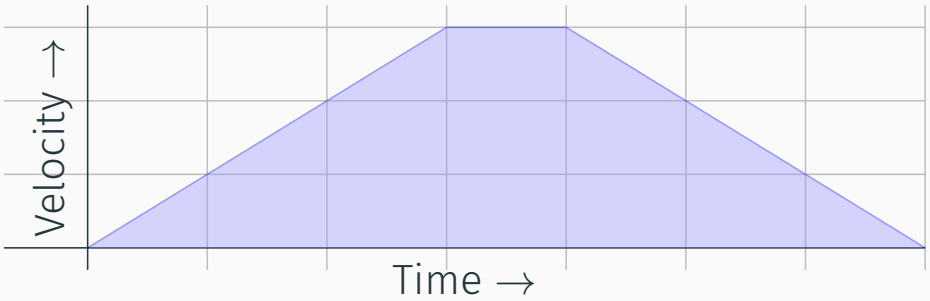




$$\begin{cases} \text{dist} = v \cdot b_1 + \frac{1}{2} \cdot v \cdot b_2 \\ v = b_2 \cdot \text{accel} \end{cases} \implies b_1 = \frac{2 \cdot \text{dist} - \frac{v^2}{a}}{2 \cdot v}$$

The more general case

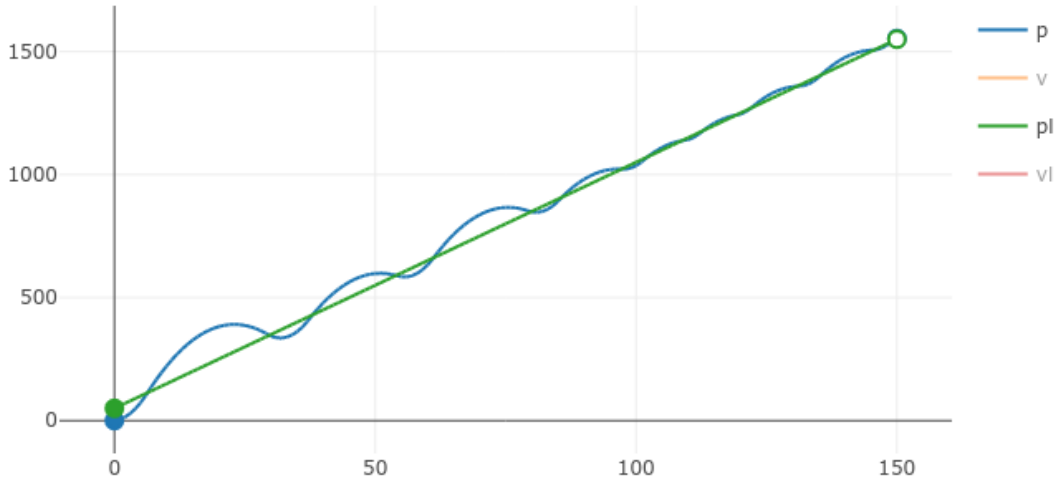
We accelerate, maintain velocity, and then brake



...

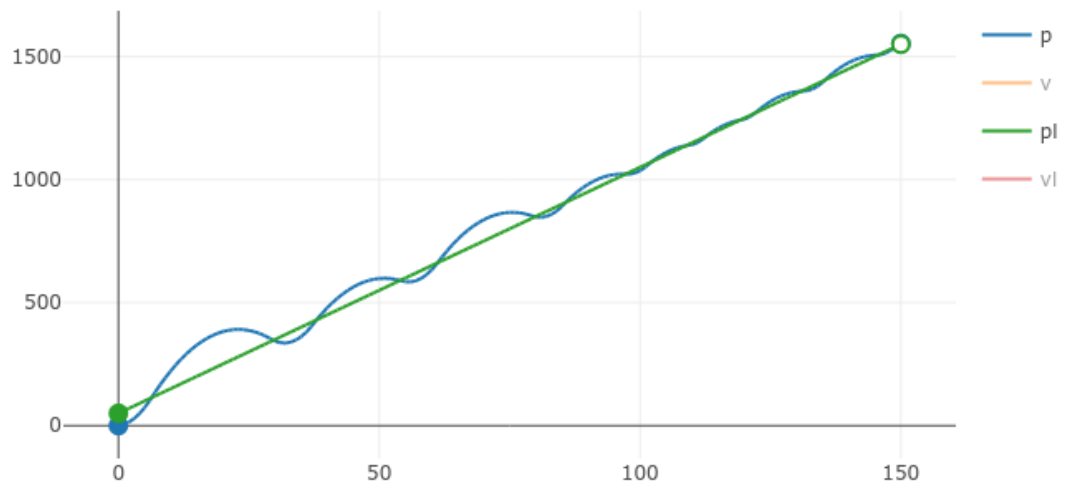
Following the leader pt. I

```
p:=0; v:=2; pl:=50; vl:=10;
while true do {
  if p + v + 2.5 < pl + 10
  then p'=v,v'=5 ,pl'=10 for 1
  else p'=v,v'=-2,pl'=10 for 1
}
```



Following the leader pt. I

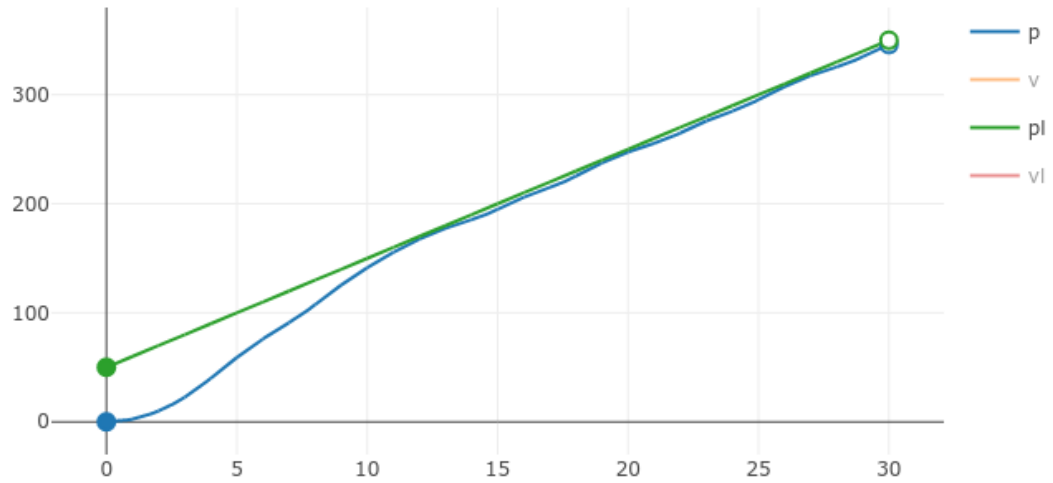
```
p:=0; v:=2; pl:=50; vl:=10;
while true do {
  if p + v + 2.5 < pl + 10
  then p'=v,v'=5 ,pl'=10 for 1
  else p'=v,v'=-2,pl'=10 for 1
}
```



Problem: Even if behind the leader in the next iteration, we might generate a velocity so high that we won't brake in time

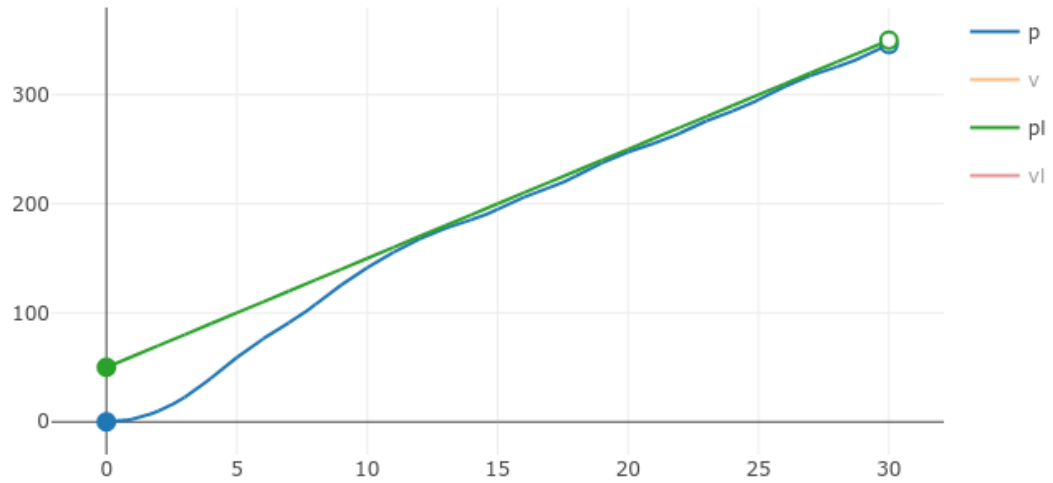
Following the leader pt. II

```
// Adaptive cruise control
// -- Follower --
p:=0; v:=0;     // position and velocity
// -- Leader --
pl:=50; vl:=10; // position and velocity
while true do{
  if (p+v+2.5 < pl+10) &&
     ((v-5)^2 +
      4*(p+v+2.5-pl-10) < 0)
  then p'=v,v'=5,pl'=10 for 1;
  else p'=v,v'=-2,pl'=10 for 1;
}
```



Following the leader pt. II

```
// Adaptive cruise control
// -- Follower --
p:=0; v:=0;     // position and velocity
// -- Leader --
pl:=50; vl:=10; // position and velocity
while true do{
  if (p+v+2.5 < pl+10) &&
     ((v-5)^2 +
      4*(p+v+2.5-pl-10) < 0)
  then p'=v,v'=5,pl'=10 for 1;
  else p'=v,v'=-2,pl'=10 for 1;
}
```



Conditional arises from solving the following equation for t

$$x_0 + v_0 t + \frac{1}{2}(-2)t^2 = y_0 + 10t$$

No solutions, means no collisions !!

Table of Contents

Overview

Semantics

Design Patterns

Conclusions

Conclusions

Studied fundamentals of program semantics

Visited a zoo of hybrid programs – which improved our ability to recognise them in the wild

Saw how to design hybrid programs formally

Conclusions

Studied fundamentals of program semantics

Visited a zoo of hybrid programs – which improved our ability to recognise them in the wild

Saw how to design hybrid programs formally

What next?

Scenarios we did not cover

Movement in n-dimensions

Trajectory correction

Orbital dynamics

...

Open Challenges

Integration of uncertainty, concurrency, and communication

A logical verification framework

A proper handle of exact real-number computation