Hybrid Programming

Renato Neves

Universidade do Minho

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

Table of Contents

Last Lectures

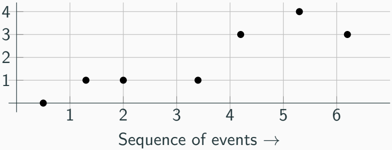
Explored a simple language (CCS) and its semantics

Used it to design communicating systems

Expanded this study to the timed setting

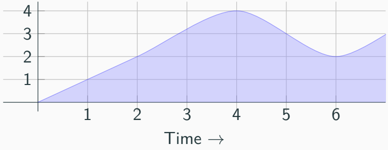
Used this to save us all from zombies !!

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

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

real number

variable

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

Programs

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

Overview

First we tackle 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

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

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

Expression $\langle \texttt{t}, \sigma \rangle \Downarrow \texttt{r}$ tells that $\texttt{t}$ outputs $\texttt{r}$ if 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

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

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

Equations $\sigma(\mathtt{x}) = 3$ 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

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

Exercises

Write down the corresponding derivation trees for

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

Boring computations? If so why not implement the semantics?

Equivalence of Linear Terms

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

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

Examples of equivalent terms:

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

A Language of Boolean Terms and its Semantics

Boolean Terms

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

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

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

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

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

While-Programs

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

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

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

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

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

The Semantics at Work

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

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

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

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

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 programs

- $(\mathtt{p} \mathbin{;} \mathtt{q}) \mathbin{;} \mathtt{r} \equiv \mathtt{p} \mathbin{;} (\mathtt{q} \mathbin{;} \mathtt{r})$
- $(\mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q}) \mathbin{;} \mathtt{r} \equiv \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p} \mathbin{;} \mathtt{r}\ \mathtt{else}\ \mathtt{q} \mathbin{;} \mathtt{r}$

Pause for Meditations

We designed our first programming language

... and used its semantics to prove program properties

Which program features would you like to add next ?

From our end we will add differential operations

Preliminaries about Differential Equations

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)

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

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

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

$\mathtt{LTerm} \ni \underbrace{\mathtt{r}}_{\text{real number}} \mid \mathtt{r} \cdot \mathtt{t} \mid \underbrace{\mathtt{x}}_{\text{variable}} \mid \mathtt{t} + \mathtt{s}$

Atomic Programs

$\mathtt{At} \ni \mathtt{x} := \mathtt{t} \mid \mathtt{x}_1' = \mathtt{t}_1, \dots, \mathtt{x}_n' = \mathtt{t}_n \underbrace{\mathtt{for}}_{\text{"run" the system of differential equations for } \mathtt{t} \text{ seconds}} \mathtt{t}$

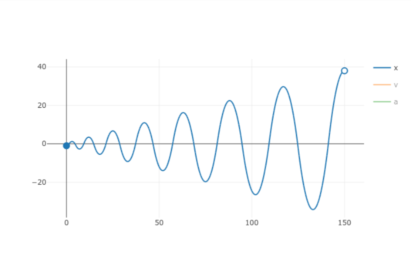
Hybrid Programs

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

Evaluation of programs is now time-dependent

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

LINCE relies on such 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

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

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

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

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

Examples

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

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

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

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

Exercise

Write down the corresponding derivation trees for

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

The Semantic Rules pt. II

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

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

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

Equivalence of While-Programs

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

$$\langle \mathtt{p}, \sigma, t \rangle \Downarrow \mathtt{s}, \sigma' \text{ iff } \langle \mathtt{q}, \sigma, t \rangle \Downarrow \mathtt{s}, \sigma'$$

Examples of equivalent terms:

- $(\mathrm{x}' = 1 \ \mathtt{for}\ 1) \mathbin{;} (\mathrm{x}' = 1 \ \mathtt{for}\ 1) \sim \mathrm{x}' = 1 \ \mathtt{for}\ 2$
- $(\mathtt{p} \mathbin{;} \mathtt{q}) \mathbin{;} \mathtt{r} \sim \mathtt{p} \mathbin{;} (\mathtt{q} \mathbin{;} \mathtt{r})$

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

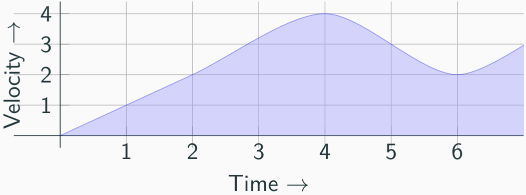
Table of Contents

A selection of design patterns

We explore the last two (ubiquitous) scenarios

Tackle them via Analytic Geometry

Moving a particle

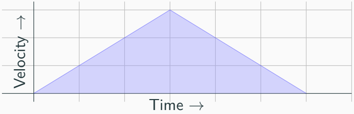


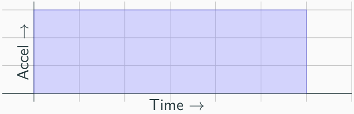
Area = distance travelled

What should be the function's shape?

Moving a particle with a fixed acceleration

We accelerate and then brake

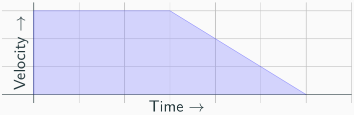


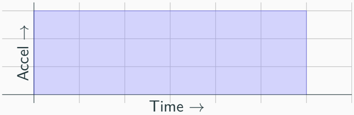


$$\begin{cases} \mathrm{dist} = \frac{1}{2} \cdot b \cdot h \\ h = \frac{1}{2} \cdot b \cdot \mathrm{accel} \end{cases} \implies b = \sqrt{\frac{4 \cdot \mathrm{dist}}{\mathrm{accel}}}$$

Moving a particle with positive velocity

We maintain velocity and then brake

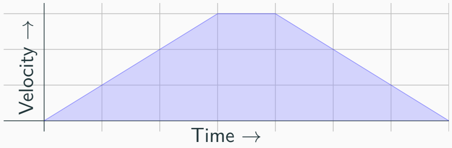




$$\begin{cases} \mathrm{dist} = v \cdot b_1 + \frac{1}{2} \cdot v \cdot b_2 \\ v = b_2 \cdot \mathrm{accel} \end{cases} \implies b_1 = \frac{2 \cdot \mathrm{dist} - \frac{v^2}{a}}{2 \cdot v}$$

The more general case

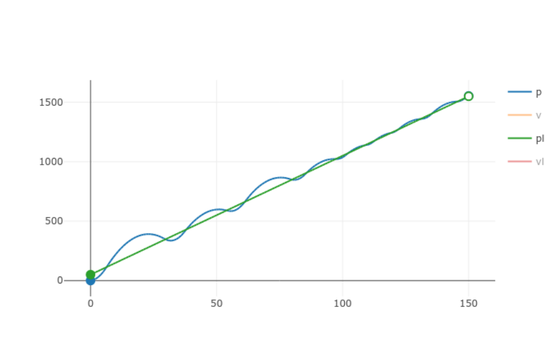
We accelerate, maintain velocity, and then brake



...

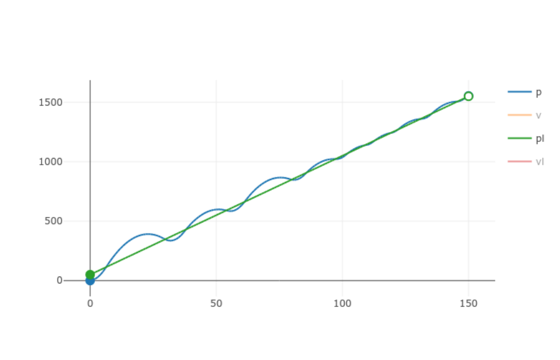
Following the leader pt. I

```
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

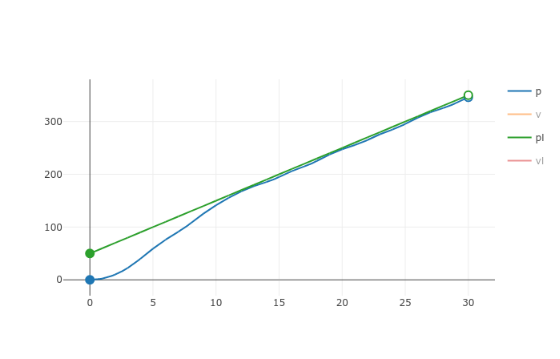
```
p:=0; v:=2; pl:=50; vl:=10;
while true do {
  if p + v + 2.5 < pl + 10
  then p'=v,v'=5 ,pl'=10 for 1
  else p'=v,v'=-2,pl'=10 for 1
}
```



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

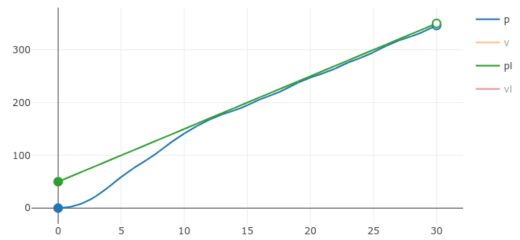
Following the leader pt. II

```
// 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 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

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