Semantics for (Hybrid) Programming

Renato Neves

Universidade do Minho

HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Table of Contents

Last Lectures

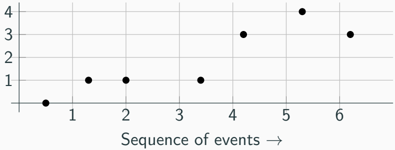
Explored a simple programming language (CCS) and its semantics

Used both to model and analyse communicating systems

Expanded our journey to the timed setting, through timed automata and UPPAAL

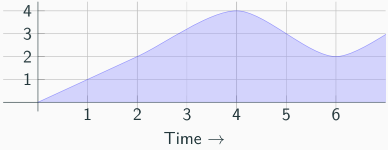
Used both to save us from zombies!

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 functional capabilities

(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 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 analyse hybrid programs

Throughout the journey, we will do:

- implementations in HASKELL
- analysis in LINCE

Table of Contents

A Language of Linear Terms and its Semantics

Linear Terms

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

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

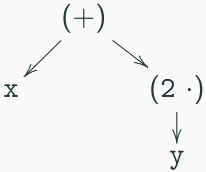
The expression $\langle \mathtt{t}, \sigma \rangle \Downarrow \mathtt{r}$ says that the linear expression $\mathtt{t}$ outputs $\mathtt{r}$ if the current memory is σ

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

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

The Semantics at Work

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



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

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

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

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

A Language of Boolean Terms and its Semantics

Boolean Terms

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

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

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

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

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

Table of Contents

A While-language and its Semantics

While-Programs

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

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

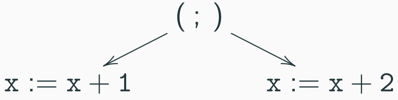
$$\frac{\langle \texttt{b}, \sigma \rangle \Downarrow \texttt{tt} \qquad \langle \texttt{p}, \sigma \rangle \Downarrow \sigma'}{\langle \texttt{if}\ \texttt{b}\ \texttt{then}\ \texttt{p}\ \texttt{else}\ \texttt{q}, \sigma \rangle \Downarrow \sigma'}\ (\text{if1}) \qquad \frac{\langle \texttt{b}, \sigma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

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

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

The Semantics at Work

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



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

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

Exercise

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

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 the differential operations

Table of Contents

Preliminaries about Differential Equations

Consider a stock $\mathcal{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$ of variables

Systems of differential equations $\mathtt{x}_1' = \mathtt{t}_1, \ldots, \mathtt{x}_n' = \mathtt{t}_n$ always have unique solutions

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

Systematically obtained via linear algebra tools

Example (The Continuous Dynamics of a Vehicle)

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

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

Conventions

We will often abbreviate a list $v_1, \dots, v_n$ simply to $\overline{v}$

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

Example

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

We will often treat an environment $\sigma : \{\mathtt{x}_1, \dots, \mathtt{x}_\mathtt{n}\} \to \mathbb{R}$ as a list $[\sigma(\mathtt{x}_1), \dots, \sigma(\mathtt{x}_\mathtt{n})]$

Fix a stock of variables $X = \{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$. 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

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

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

Hybrid Programs

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

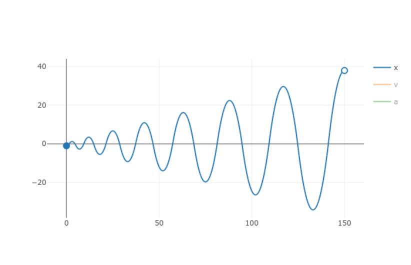
. . . its semantics

The evaluation of programs is now time-dependent

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

. . . different time instants, different outputs

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



The Semantic Rules pt. I

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

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

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

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

Examples

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

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

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

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

Exercise

$$\langle (\mathrm{x}' = 1 \ \texttt{for}\ 1) \,;\, (\mathrm{x}' = -1 \ \texttt{for}\ 1), (\mathrm{x} \mapsto 5), 2 \rangle \Downarrow ?$$

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:

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

Table of Contents

A Zoo of Hybrid Programs

- Traffic Lights
- Cruise Controller
- Landing System

A Million-Dollar Question

How to simulate a differential statement that terminates as soon as a certain event occurs?

$$x' = 1 \;\texttt{until}\; x = 2$$

A: No general solution for simulation with exact precision; and even approximated simulation raises problems :-(

$(x' = 1 \;\texttt{until}\; x = 2)$ collapses almost always to $(x' = 1 \;\texttt{for}\; \infty)$

A Million-Dollar Question

How to simulate a differential statement that terminates as soon as a certain event occurs?

$$x' = 1 \ \mathtt{until}\ x = 2$$

A: No general solution for simulation with exact precision; and even approximated simulation raises problems :-(

$(x' = 1 \ \mathtt{until}\ x = 2)$ collapses almost always to $(x' = 1 \ \mathtt{for}\ \infty)$

For this lecture we take a (naive) approach:

$$(\overline{x}' = \overline{t} \ \mathtt{until}_\epsilon\ b) \mathrel{\hat{=}} \mathtt{while}\ \neg b\ \{\overline{x}' = \overline{t}\ \mathtt{for}\ \epsilon\}$$

(Another) Zoo of Hybrid Programs

- Bouncing Ball
- Fireflies

Table of Contents

Checkpoint

We saw how to analyse hybrid programs formally

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

Checkpoint

We saw how to analyse hybrid programs formally

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

We now go over examples of what not to do in hybrid programming

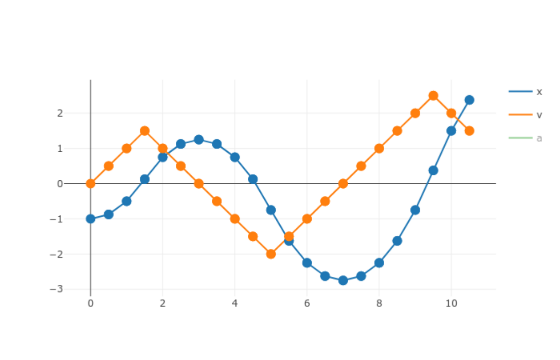
What not to do

Neglect:

- error accumulation or
- analytical testing

Error Accumulation and Particle Positioning

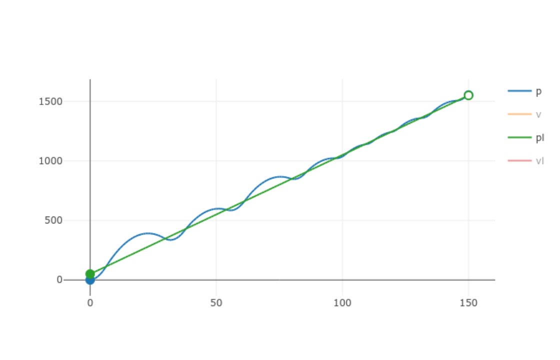
```
x:= -1; v:= 0; a:= 1;
while true do {
  if x <= 0 then a:= 1 else {a:=-1 };
      x' = v, v' = a  for 0.5
}
```



Q: What is the position of the particle the first time it stops?

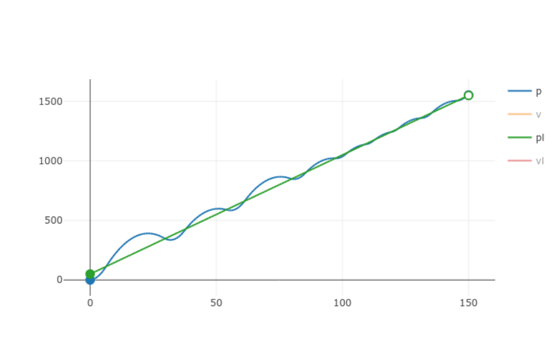
Analytical Testing and Following the Leader pt. I

```
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
}
```



Analytical Testing and Following the Leader pt. I

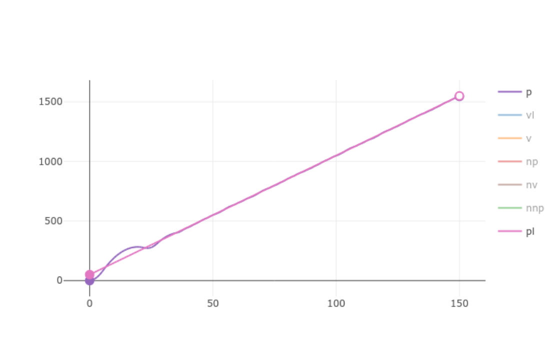
```
p:=0; v:=2; pl:=50; vl:=10;
while true do {
  if p + v + 2.5 < pl + 10
  then p'=v,v'=5 ,pl'=10 for 1
  else p'=v,v'=-2,pl'=10 for 1
}
```



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

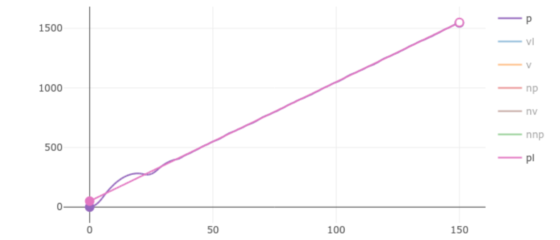
Analytical Testing and Following the Leader pt. II

```
p:=0; v:=2; pl:=50; vl:=10;
np:=0; nv:=0; nnp:=0;
while true do {
  np := p + v + 2.5;
  nv := v + 5;
  nnp := (p + v + 2.5) + (v + 5) - 1;
  if (np < pl + 10) /\ (nnp < pl + 20)
  then p'=v,v'=5 ,pl'=10 for 1
  else p'=v,v'=-2,pl'=10 for 1
}
```



Analytical Testing and Following the Leader pt. II

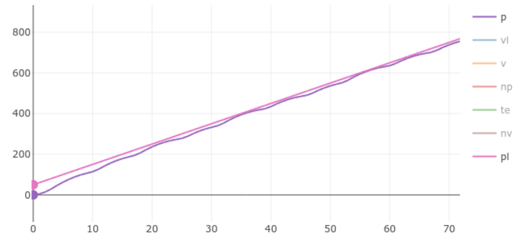
```
p:=0; v:=2; pl:=50; vl:=10;
np:=0; nv:=0; nnp:=0;
while true do {
  np := p + v + 2.5;
  nv := v + 5;
  nnp := (p + v + 2.5) + (v + 5) - 1;
  if (np < pl + 10) /\ (nnp < pl + 20)
  then p'=v,v'=5 ,pl'=10 for 1
  else p'=v,v'=-2,pl'=10 for 1
}
```



Problem: We might still generate an undetected, unsafe velocity – "safe" should amount to "not collide until velocity becomes lower than the leader" rather than "not collide at the end of the next iteration"

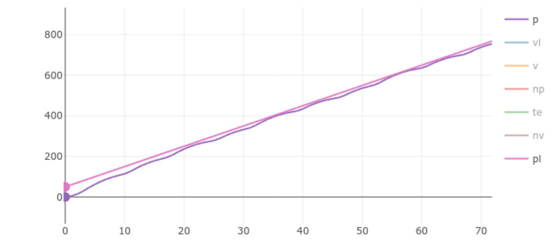
Analytical Testing and Following the Leader pt. III

```
p:=0; v:=2; pl:=50; vl:=10;
np:=0; nv:=0; te:=0;
while true do {
  np := p + v + 2.5;
  nv := v + 5;
  te := (nv -10)*45 + 4*(np - (pl + 10));
  if (np < pl + 10) /\ (te <= 0)
  then p'=v,v'=5 ,pl'=10 for 1
  else p'=v,v'=-2,pl'=10 for 1
}
```



Analytical Testing and Following the Leader pt. III

```
p:=0; v:=2; pl:=50; vl:=10;
np:=0; nv:=0; te:=0;
while true do {
  np := p + v + 2.5;
  nv := v + 5;
  te := (nv -10)*45 + 4*(np - (pl + 10));
  if (np < pl + 10) /\ (te <= 0)
  then p'=v,v'=5 ,pl'=10 for 1
  else p'=v,v'=-2,pl'=10 for 1
}
```



The conditional now 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!!