

# Semantics for (Hybrid) Programming

---

Renato Neves



Universidade do Minho



# Table of Contents

Overview

Semantics for Linear Terms

Semantics for Boolean Terms

Semantics for While Programs

Semantics for Hybrid-while Programs

A Zoo of Hybrid Programs

Examples of what not to do

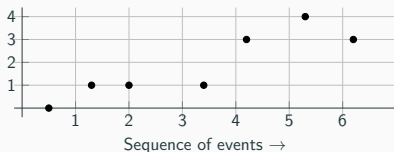
Explored a simple language (CCS) and its semantics

Used it to model and analyse communicating systems

Expanded our study to the timed setting, via UPPAAL

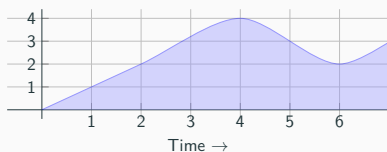
Used it to save us from zombies!

# 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  $X = \{x_1, \dots, x_n\}$ . Then we have,

## Linear Terms

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


real number

## Atomic Programs

$$\text{At}(X) \ni x := t \mid x'_1 = t_1, \dots, x'_n = t_n \text{ for } t$$


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

## Hybrid Programs

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

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

Overview

Semantics for Linear Terms

Semantics for Boolean Terms

Semantics for While Programs

Semantics for Hybrid-while Programs

A Zoo of Hybrid Programs

Examples of what not to do



# A Language of Linear Terms and its Semantics

## Linear Terms

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

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

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

$$\frac{}{\langle x, \sigma \rangle \Downarrow \sigma(x)} \text{ (var)}$$

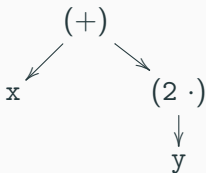
$$\frac{}{\langle r, \sigma \rangle \Downarrow r} \text{ (con)}$$

$$\frac{\langle t, \sigma \rangle \Downarrow r}{\langle s \cdot t, \sigma \rangle \Downarrow s \cdot r} \text{ (scl)}$$

$$\frac{\langle t_1, \sigma \rangle \Downarrow r_1 \quad \langle t_2, \sigma \rangle \Downarrow r_2}{\langle t_1 + t_2, \sigma \rangle \Downarrow r_1 + r_2} \text{ (add)}$$

# The Semantics at Work

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



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

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

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

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

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

# Equivalence of Linear Terms

The previous semantics yields the following notion of **equivalence**:  
 $t \sim s$  if for all environments  $\sigma$

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

Examples of equivalent terms:

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

# Table of Contents

Overview

Semantics for Linear Terms

Semantics for Boolean Terms

Semantics for While Programs

Semantics for Hybrid-while Programs

A Zoo of Hybrid Programs

Examples of what not to do

# A Language of Boolean Terms and its Semantics

## Boolean Terms

$\text{BTerm}(X) \ni t_1 \leq t_2 \mid b \wedge c \mid \neg b$

# A Language of Boolean Terms and its Semantics

## Boolean Terms

$$\text{BTerm}(X) \ni t_1 \leq t_2 \mid b \wedge c \mid \neg b$$

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

$$\frac{\langle t_1, \sigma \rangle \Downarrow r_1 \quad \langle t_2, \sigma \rangle \Downarrow r_2 \quad r_1 \leq r_2}{\langle t_1 \leq t_2, \sigma \rangle \Downarrow \text{tt}} \text{ (leq)}$$

$$\frac{\langle t_1, \sigma \rangle \Downarrow r_1 \quad \langle t_2, \sigma \rangle \Downarrow r_2 \quad r_1 \not\leq r_2}{\langle t_1 \leq t_2, \sigma \rangle \Downarrow \text{ff}} \text{ (gtr)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow v}{\langle \neg b, \sigma \rangle \Downarrow \neg v} \text{ (not)}$$

$$\frac{\langle b_1, \sigma \rangle \Downarrow v_1 \quad \langle b_2, \sigma \rangle \Downarrow v_2}{\langle b_1 \wedge b_2, \sigma \rangle \Downarrow v_1 \wedge v_2} \text{ (and)}$$



# Table of Contents

Overview

Semantics for Linear Terms

Semantics for Boolean Terms

Semantics for While Programs

Semantics for Hybrid-while Programs

A Zoo of Hybrid Programs

Examples of what not to do

# A While-language and its Semantics

## While-Programs

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

$$\frac{\langle t, \sigma \rangle \Downarrow r}{\langle x := t, \sigma \rangle \Downarrow \sigma[r/x]} \text{ (asg)}$$

$$\frac{\langle p, \sigma \rangle \Downarrow \sigma' \quad \langle q, \sigma' \rangle \Downarrow \sigma''}{\langle p ; q, \sigma \rangle \Downarrow \sigma''} \text{ (seq)}$$

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

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

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

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

# The Semantics at Work

The program  $x := x + 1 ; x := x + 2$  corresponds to the tree



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

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

# Exercise

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

# Equivalence of While-Programs

The previous semantics yields the following notion of **equivalence**:  
 $p \sim q$  if for all environments  $\sigma$

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

Examples of equivalent terms:

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

# 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**

# Table of Contents

Overview

Semantics for Linear Terms

Semantics for Boolean Terms

Semantics for While Programs

Semantics for Hybrid-while Programs

A Zoo of Hybrid Programs

Examples of what not to do

# Preliminaries about Differential Equations

Consider a stock  $\mathcal{X} = \{x_1, \dots, x_n\}$  of variables

Systems of differential equations  $x'_1 = t_1, \dots, x'_n = t_n$  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)

$p' = v, v' = a$  which admits the solution

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



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

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

## 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  $\sigma : \{x_1, \dots, x_n\} \rightarrow \mathbb{R}$  as a list  $[\sigma(x_1), \dots, \sigma(x_n)]$

# The Hybrid While-Language and ...

Fix a stock of variables  $X = \{x_1, \dots, x_n\}$ . Then we have,

## Linear Terms

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



real number

## Atomic Programs

$$\text{At}(X) \ni x := t \mid x'_1 = t_1, \dots, x'_n = t_n \text{ for } t$$



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

## Hybrid Programs

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

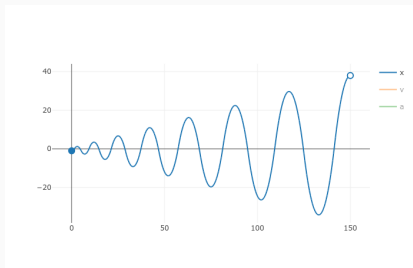
## ... its semantics

The evaluation of programs is now **time-dependent**

$$\langle p, \sigma, t \rangle \Downarrow \sigma'$$

... different time instants, different outputs

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



# The Semantic Rules pt. I

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

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

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

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

# Examples

$$\begin{array}{c}
 \langle 1, (x \mapsto 2) \rangle \Downarrow 1 \quad \frac{1}{2} < 1 \\
 \hline
 \langle x' = 0 \text{ for } 1, (x \mapsto 2), \frac{1}{2} \rangle \Downarrow \text{stop}, (x \mapsto 2) \\
 \hline
 \langle (x' = 0 \text{ for } 1) ; (x' = 1 \text{ for } 1), (x \mapsto 2), \frac{1}{2} \rangle \Downarrow \text{stop}, (x \mapsto 2) \\
 \downarrow \\
 = (x \mapsto 2)[\phi(2, \frac{1}{2})/x]
 \end{array}$$

$$\begin{array}{c}
 \dots \quad \dots \\
 \hline
 \langle x' = 0 \text{ for } 1, (x \mapsto 2), 1 \rangle \Downarrow \text{skip}, (x \mapsto 2) \quad \langle x' = 1 \text{ for } 1, (x \mapsto 2), \frac{1}{2} \rangle \Downarrow \text{stop}, (x \mapsto 2 + \frac{1}{2}) \\
 \hline
 \langle (x' = 0 \text{ for } 1) ; (x' = 1 \text{ for } 1), (x \mapsto 2), 1 + \frac{1}{2} \rangle \Downarrow \text{stop}, (x \mapsto 2 + \frac{1}{2}) \\
 \downarrow \\
 = (x \mapsto 2)[\phi(2, \frac{1}{2})/x] = (x \mapsto 2)[2 + \frac{1}{2}/x] = x \mapsto 2 + \frac{1}{2}
 \end{array}$$

# Exercise

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

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

## The Semantic Rules pt. II

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

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

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

# Equivalence of While-Programs

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

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

Examples of equivalent terms:

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



# Table of Contents

Overview

Semantics for Linear Terms

Semantics for Boolean Terms

Semantics for While Programs

Semantics for Hybrid-while Programs

**A Zoo of Hybrid Programs**

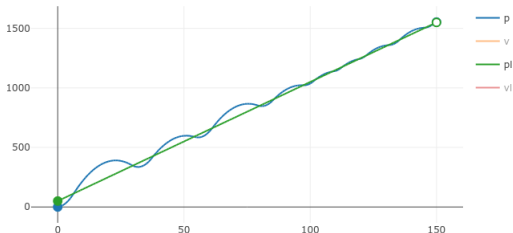
Examples of what not to do

# A Zoo of Hybrid Programs

- Traffic lights
- Cruise controller (speed regulation)
- Landing system
- Move a particle from A to B
- Follow the leader

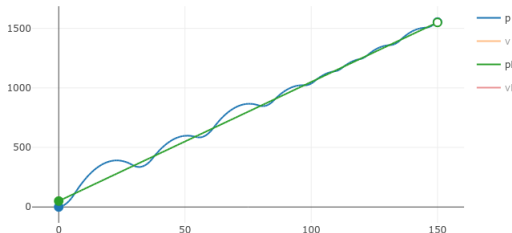
# Analytical Testing and Following the Leader pt. I

```
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

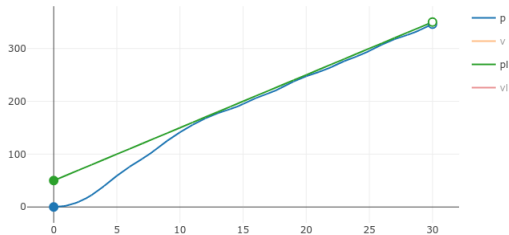
```
p:=0; v:=2; pl:=50; vl:=10;  
while true do {  
  if p + v + 2.5 < pl + 10  
  then p'=v, v'=5 ,pl'=10 for 1  
  else p'=v, v'=-2, pl'=10 for 1  
}
```



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

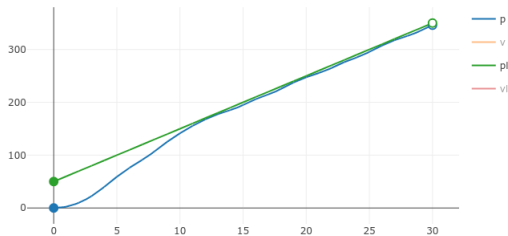
# Analytical Testing and Following the Leader pt. II

```
// 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;
}
```



# Analytical Testing and 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;
}
```



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!!

# Table of Contents

Overview

Semantics for Linear Terms

Semantics for Boolean Terms

Semantics for While Programs

Semantics for Hybrid-while Programs

A Zoo of Hybrid Programs

Examples of what not to do

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 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

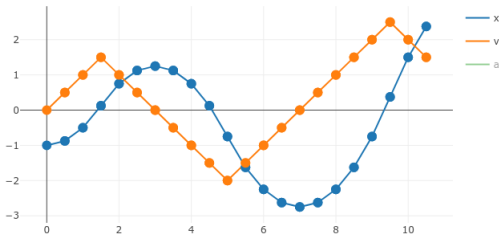
# What not to do

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?

# A Million-Dollar Question

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

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

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

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

# A Million-Dollar Question

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

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

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

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

For this lecture we take a (naive) approach:

$$(\bar{x}' = \bar{t} \text{ until}_\epsilon b) \triangleq \text{while } \neg b \{ \bar{x}' = \bar{t} \text{ for } \epsilon \}$$

# (Another) Zoo of Hybrid Programs

- Bouncing Ball
- Fireflies