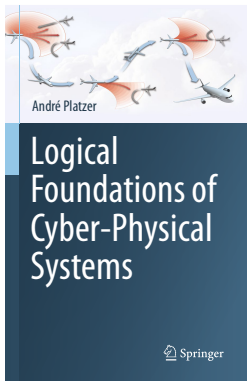


03: Choice & Control

Logical Foundations of Cyber-Physical Systems



André Platzer

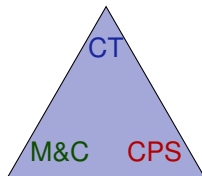
Karlsruhe Institute of Technology
Department of Informatics

Computer Science Department
Carnegie Mellon University

- 1 Learning Objectives
- 2 Gradual Introduction to Hybrid Programs
- 3 Hybrid Programs
 - Syntax
 - Semantics
 - Notational Convention
- 4 Examples
- 5 Summary

- 1 Learning Objectives
- 2 Gradual Introduction to Hybrid Programs
- 3 Hybrid Programs
 - Syntax
 - Semantics
 - Notational Convention
- 4 Examples
- 5 Summary

nondeterminism
abstraction
programming languages for CPS
semantics
compositionality



models
core principles
discrete+
continuous

operational effect
operational precision



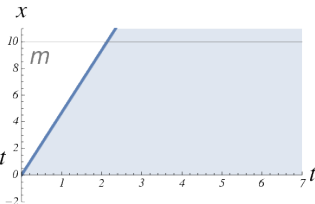
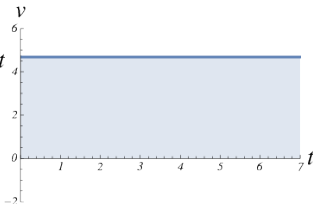
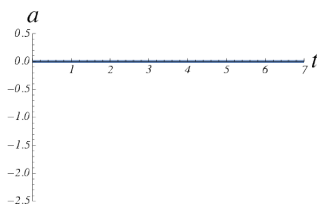
- 1 Learning Objectives
- 2 Gradual Introduction to Hybrid Programs**
- 3 Hybrid Programs
 - Syntax
 - Semantics
 - Notational Convention
- 4 Examples
- 5 Summary

Example (Speedy the point)

$$\{x' = v, v' = a\}$$

Purely continuous dynamics

What about the cyber?

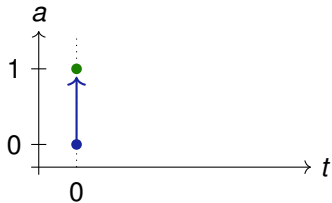


Example (Speedy the point)

$$a := a + 1$$

Purely discrete dynamics

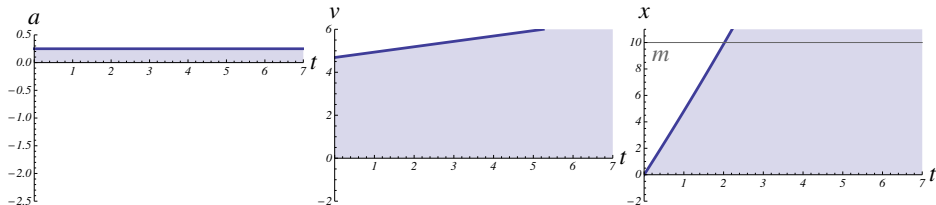
How do both meet?



Example (Speedy the point)

$$a := a + 1; \{x' = v, v' = a\}$$

Hybrid dynamics, i.e., composition of continuous and discrete dynamics
 Here: sequential composition first;second



Example (Speedy the point)

$$a := -2; \{x' = v, v' = a\};$$

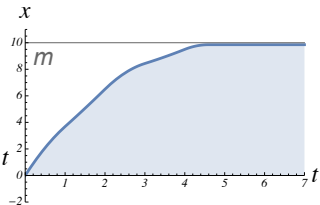
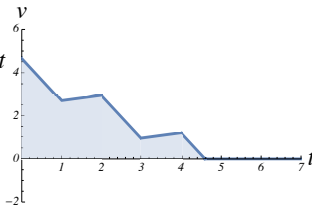
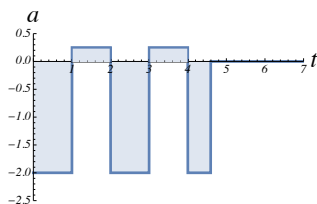
$$a := 0.25; \{x' = v, v' = a\};$$

$$a := -2; \{x' = v, v' = a\};$$

$$a := 0.25; \{x' = v, v' = a\};$$

$$a := -2; \{x' = v, v' = a\};$$

$$a := 0.25; \{x' = v, v' = a\}$$



Example (Speedy the point)

$$a := -2; \{x' = v, v' = a\};$$

$$a := 0.25; \{x' = v, v' = a\};$$

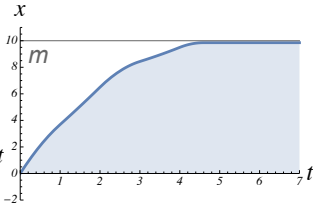
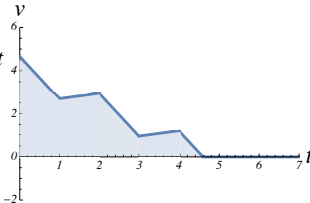
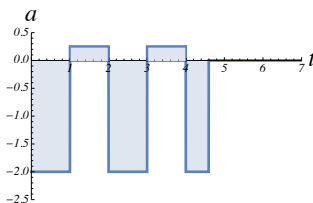
$$a := -2; \{x' = v, v' = a\};$$

$$a := 0.25; \{x' = v, v' = a\};$$

$$a := -2; \{x' = v, v' = a\};$$

$$a := 0.25; \{x' = v, v' = a\}$$

How long to follow an ODE?



Example (Speedy the point)

$$a := -2; \{x' = v, v' = a\};$$

$$a := 0.25; \{x' = v, v' = a\};$$

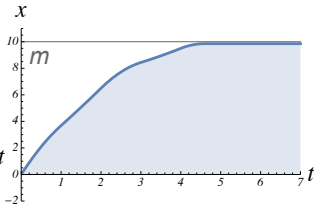
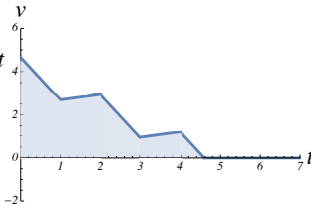
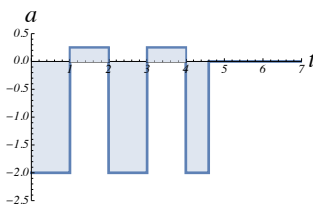
$$a := -2; \{x' = v, v' = a\};$$

$$a := 0.25; \{x' = v, v' = a\};$$

$$a := -2; \{x' = v, v' = a\};$$

$$a := 0.25; \{x' = v, v' = a\}$$

How to check conditions before actions?



Example (Speedy the point)

```
if( $v < 4$ )  $a := a + 1$  else  $a := -b$ ;  
{ $x' = v, v' = a$ }
```

Velocity-dependent control

Example (Speedy the point)

$$\text{if}(x - m > s) a := a + 1 \text{ else } a := -b;$$
$$\{x' = v, v' = a\}$$

Distance-dependent control for obstacle m

Example (Speedy the point)

$$\text{if}(x - m > s \wedge v < 4) a := a + 1 \text{ else } a := -b;$$
$$\{x' = v, v' = a\}$$

Velocity **and** distance-dependent control

Iterative Design

Start as simple as possible, then add challenges once basics are correct.

Example (Speedy the point)

$$\text{if}(x - m > s \wedge v < 4 \wedge \text{efficiency}) a := a + 1 \text{ else } a := -b;$$
$$\{x' = v, v' = a\}$$

Also only accelerate if it's efficient to do so

Example (Speedy the point)

$$\text{if}(x - m > s \wedge v < 4 \wedge \text{efficiency}) a := a + 1 \text{ else } a := -b;$$
$$\{x' = v, v' = a\}$$

Exact models are unnecessarily complex. Not all features are safety-critical.

Example (Speedy the point)

$$(a := a + 1 \cup a := -b); \\ \{x' = v, v' = a\}$$

Nondeterministic choice \cup allows either side to be run, arbitrarily

Power of Abstraction

Only include relevant aspects, elide irrelevant detail.

The model and its analysis become simpler. And apply to more systems.

Example (Speedy the point)

$$(a := a + 1 \cup a := -b); \\ \{x' = v, v' = a\}$$

Nondeterministic choice \cup allows either side to be run, arbitrarily
Oops, now it got too simple! Not every choice is always acceptable.

Example (Speedy the point)

$$(?v < 4; a := a + 1 \cup a := -b);$$
$$\{x' = v, v' = a\}$$

Test ?Q checks if formula Q is true in current state

Example (Speedy the point)

$$(?v < 4; a := a + 1 \cup a := -b); \\ \{x' = v, v' = a\}$$

Test $?Q$ checks if formula Q is true in current state, otherwise run fails.

Discarding failed runs and backtracking

System runs that fail tests are discarded and not considered further.

$$?v < 4; v := v + 1 \quad \text{only runs if} \\ v := v + 1; ?v < 4 \quad \text{only runs if}$$

Broader significance of nondeterminism

Nondeterminism is a tool for abstraction to focus on critical aspects.

Nondeterminism is essential to describe imperfectly known environment.

Example (Speedy the point)

$$(?v < 4; a := a + 1 \cup a := -b); \\ \{x' = v, v' = a\}$$

Test $?Q$ checks if formula Q is true in current state, otherwise run fails.

Discarding failed runs and backtracking

System runs that fail tests are discarded and not considered further.

$?v < 4; v := v + 1$ only runs if $v < 4$ initially true
 $v := v + 1; ?v < 4$ only runs if $v < 3$ initially true

Broader significance of nondeterminism

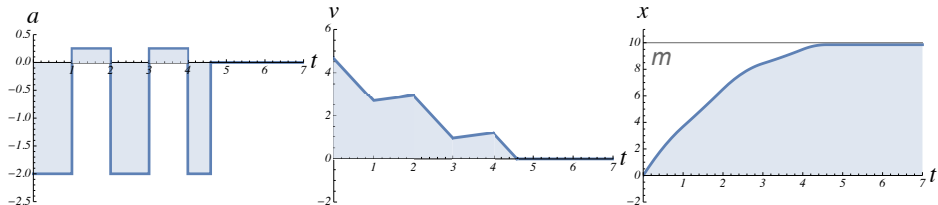
Nondeterminism is a tool for abstraction to focus on critical aspects.

Nondeterminism is essential to describe imperfectly known environment.

Example (Speedy the point)

$$\begin{aligned}
 & (?v < 4; a := a + 1 \cup a := -b); \\
 & \{x' = v, v' = a\}; \\
 & (?v < 4; a := a + 1 \cup a := -b); \\
 & \{x' = v, v' = a\}; \\
 & (?v < 4; a := a + 1 \cup a := -b); \\
 & \{x' = v, v' = a\}
 \end{aligned}$$

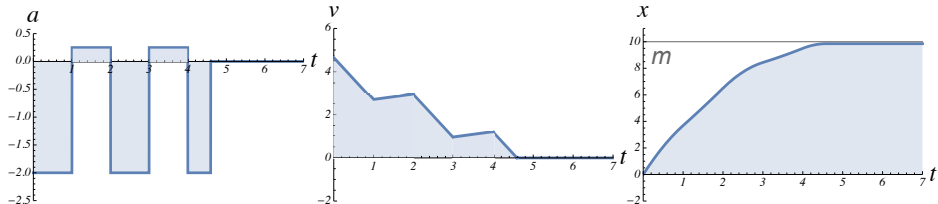
Repeated control needs longer programs, e.g., by copy&paste



Example (Speedy the point)

$$((?v < 4; a := a + 1 \cup a := -b); \{x' = v, v' = a\})^*$$

Nondeterministic repetition * repeats *any* arbitrary number of times



- 1 Learning Objectives
- 2 Gradual Introduction to Hybrid Programs
- 3 Hybrid Programs**
 - Syntax
 - Semantics
 - Notational Convention
- 4 Examples
- 5 Summary

Definition (Syntax of hybrid program α)

$$\alpha, \beta ::= x := e \mid ?Q \mid x' = f(x) \& Q \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^*$$

Definition (Syntax of hybrid program α)

$\alpha, \beta ::= x := e \mid ?Q \mid x' = f(x) \& Q \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^*$

Discrete
Assign

Test
Condition

Differential
Equation

Nondet.
Choice

Seq.
Compose

Nondet.
Repeat

Definition (Syntax of hybrid program α)

$$\alpha, \beta ::= x := e \mid ?Q \mid x' = f(x) \& Q \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^*$$

Discrete
Assign

Test
Condition

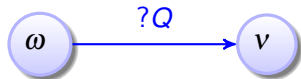
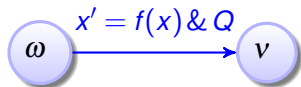
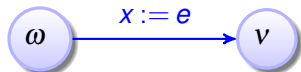
Differential
Equation

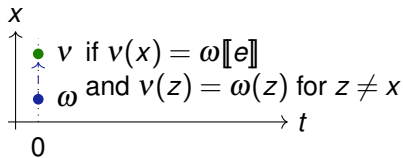
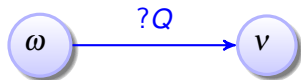
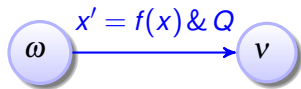
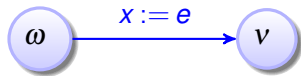
Nondet.
Choice

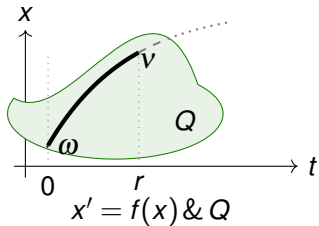
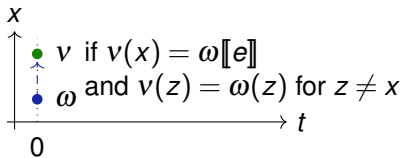
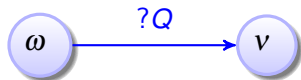
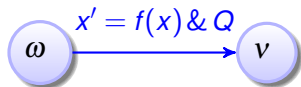
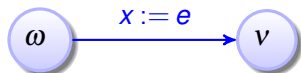
Seq.
Compose

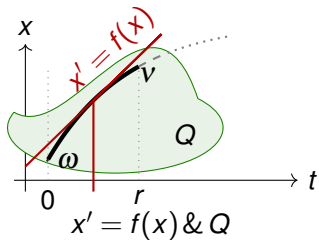
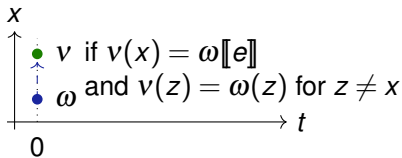
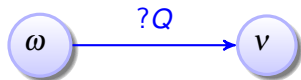
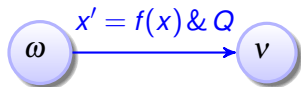
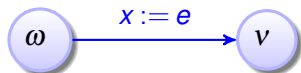
Nondet.
Repeat

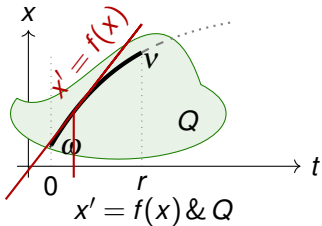
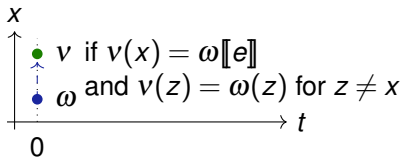
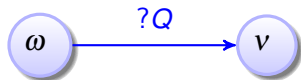
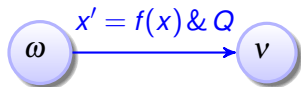
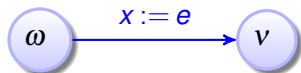
Like regular expressions. Everything nondeterministic

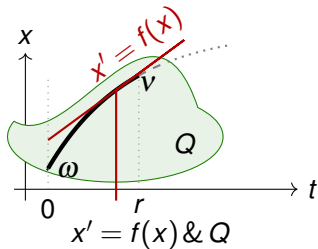
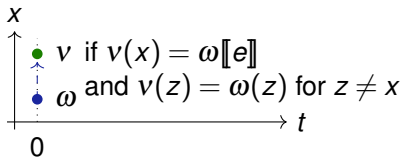
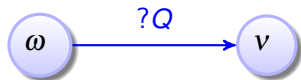
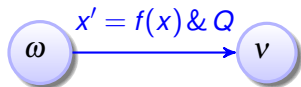
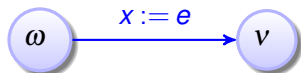


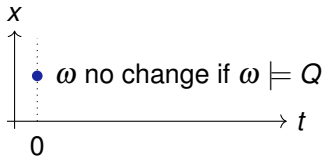
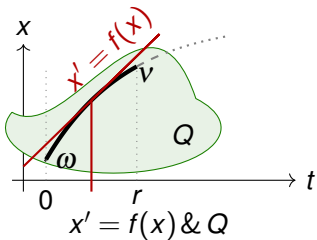
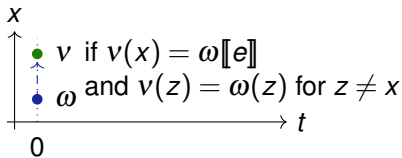
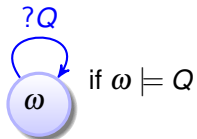
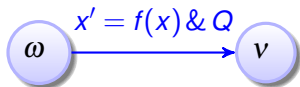
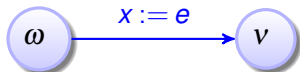


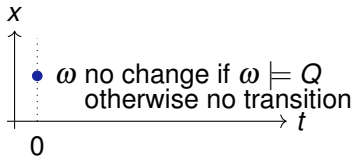
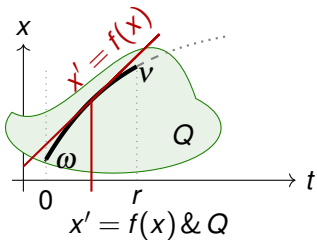
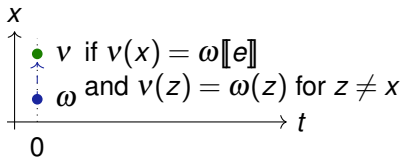
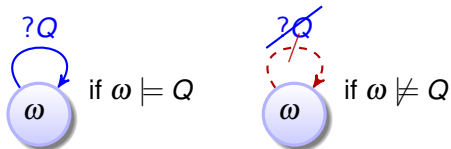
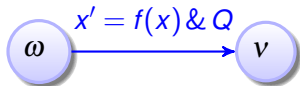
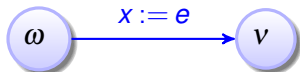


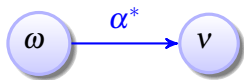
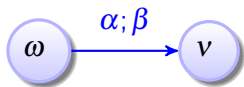
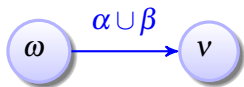


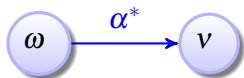
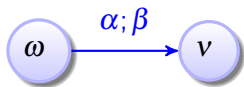
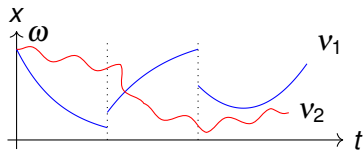
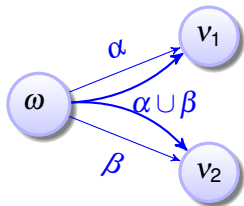


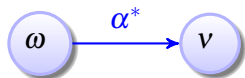
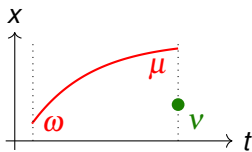
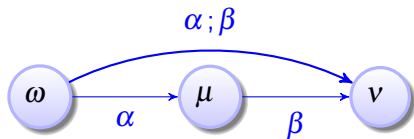
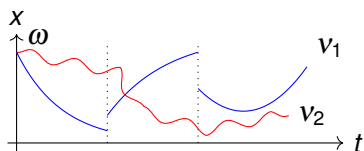
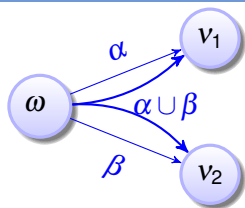


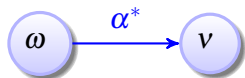
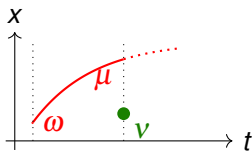
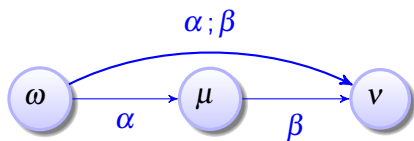
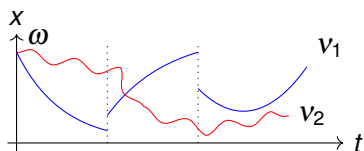
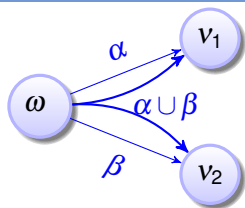


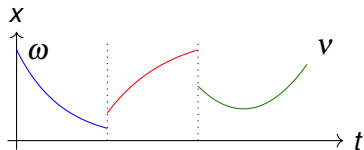
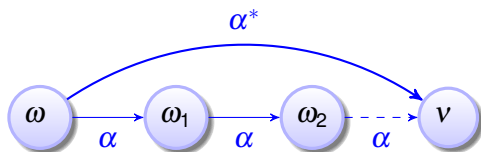
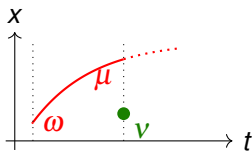
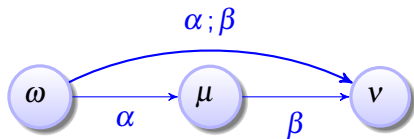
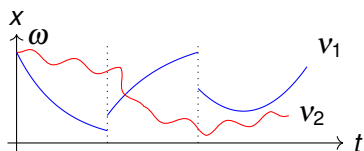
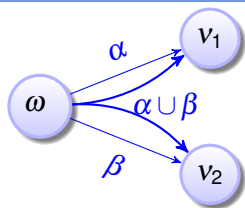


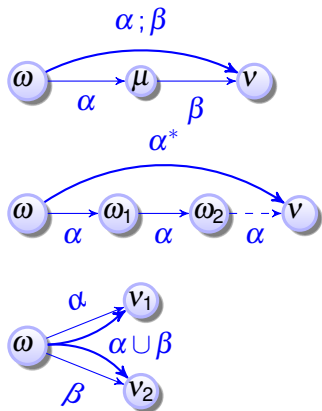


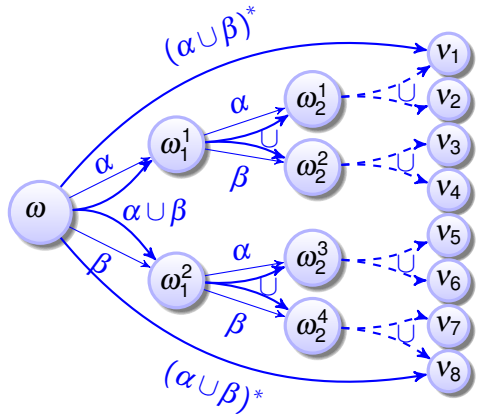
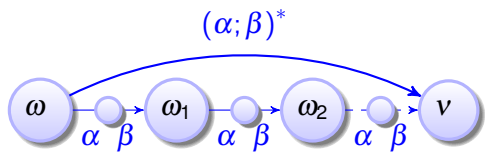
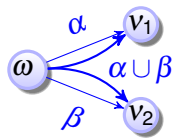
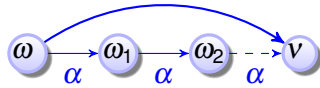
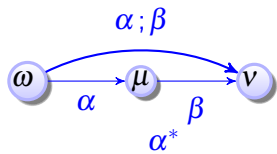












Definition (Syntax of hybrid program α)

$$\alpha, \beta ::= x := e \mid ?Q \mid x' = f(x) \& Q \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^*$$

Definition (Semantics of hybrid programs) $([\![\cdot]\!] : \text{HP} \rightarrow \wp(\mathcal{I} \times \mathcal{I}))$

$$[\![x := e]\!] = \{(\omega, \nu) : \nu = \omega \text{ except } \nu[x] = \omega[e]\}$$

$$[\![?Q]\!] = \{(\omega, \omega) : \omega \models Q\}$$

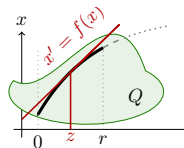
$$[\![x' = f(x)]\!] = \{(\varphi(0), \varphi(r)) : \varphi \models x' = f(x) \text{ for some duration } r \geq 0\}$$

$$[\![\alpha \cup \beta]\!] = [\![\alpha]\!] \cup [\![\beta]\!]$$

$$[\![\alpha; \beta]\!] = [\![\alpha]\!] \circ [\![\beta]\!] = \{(\omega, \nu) : (\omega, \mu) \in [\![\alpha]\!] \text{ and } (\mu, \nu) \in [\![\beta]\!]\}$$

$$[\![\alpha^*]\!] = [\![\alpha]\!]^* = \bigcup_{n \in \mathbb{N}} [\![\alpha^n]\!] \quad \alpha^n \equiv \underbrace{\alpha; \alpha; \alpha; \dots; \alpha}_{n \text{ times}}$$

compositional



Definition (Syntax of hybrid program α)

$$\alpha, \beta ::= x := e \mid ?Q \mid x' = f(x) \& Q \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^*$$

Definition (Semantics of hybrid programs) $(\llbracket \cdot \rrbracket : \text{HP} \rightarrow \wp(\mathcal{I} \times \mathcal{I}))$

$$\llbracket x := e \rrbracket = \{(\omega, \nu) : \nu = \omega \text{ except } \nu[x] = \omega[e]\}$$

$$\llbracket ?Q \rrbracket = \{(\omega, \omega) : \omega \models Q\}$$

$$\llbracket x' = f(x) \rrbracket = \{(\varphi(0), \varphi(r)) : \varphi \models x' = f(x) \text{ for some duration } r \geq 0\}$$

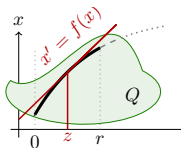
$$\llbracket \alpha \cup \beta \rrbracket = \llbracket \alpha \rrbracket \cup \llbracket \beta \rrbracket$$

$$\llbracket \alpha; \beta \rrbracket = \llbracket \alpha \rrbracket \circ \llbracket \beta \rrbracket$$

$$\llbracket \alpha^* \rrbracket = \llbracket \alpha \rrbracket^* = \bigcup_{n \in \mathbb{N}} \llbracket \alpha^n \rrbracket$$

compositional

- 1 $\varphi(z)(x') = \frac{d\varphi(t)(x)}{dt}(z)$ exists at all times $0 \leq z \leq r$
- 2 $\varphi(z) \models x' = f(x) \wedge Q$ for all times $0 \leq z \leq r$
- 3 $\varphi(z) = \varphi(0)$ except at x, x'



Example (Naming Conventions)

Letters	Convention
x, y, z	variables
e, \tilde{e}	terms
P, Q	formulas
α, β	programs
c	constant symbols
f, g, h	function symbols
p, q, r	predicate symbols

In CPS applications, all bets are off because names follow application:
 x position v velocity and a acceleration variables

Convention (Operator Precedence)

- 1 Unary operators (including $*$, \neg and $\forall x, \exists x$) bind stronger than binary.
- 2 \wedge binds stronger than \vee , which binds stronger than $\rightarrow, \leftrightarrow$
- 3 $;$ binds stronger than \cup
- 4 Arithmetic operators $+, -, \cdot$ associate to the left
- 5 Logical and program operators associate to the right

Example (Operator Precedence)

$$\forall x P \wedge Q \equiv (\forall x P) \wedge Q \qquad \forall x P \rightarrow Q \equiv (\forall x P) \rightarrow Q.$$

$$\alpha; \beta \cup \gamma \equiv (\alpha; \beta) \cup \gamma \qquad \alpha \cup \beta; \gamma \equiv \alpha \cup (\beta; \gamma) \qquad \alpha; \beta^* \equiv \alpha; (\beta^*)$$

$$P \rightarrow Q \rightarrow R \equiv P \rightarrow (Q \rightarrow R).$$

But $\rightarrow, \leftrightarrow$ expect explicit parentheses. Illegal: $P \rightarrow Q \leftrightarrow R$ $P \leftrightarrow Q \rightarrow R$

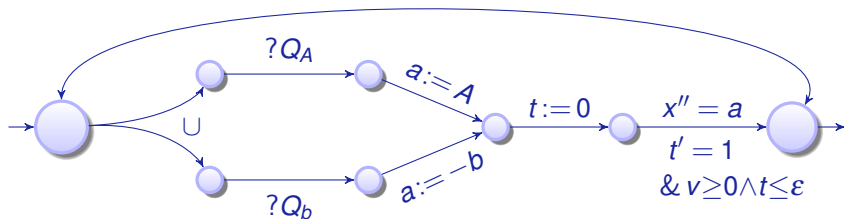
- 1 Learning Objectives
- 2 Gradual Introduction to Hybrid Programs
- 3 Hybrid Programs
 - Syntax
 - Semantics
 - Notational Convention
- 4 Examples**
- 5 Summary

Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_B; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \varepsilon\}$

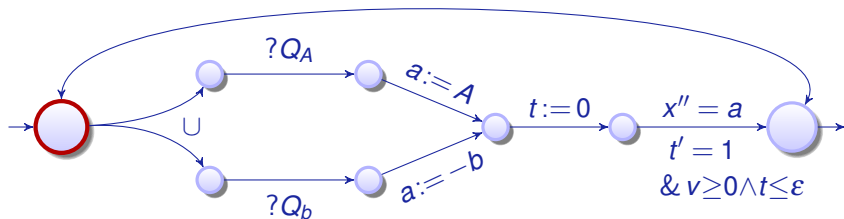


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

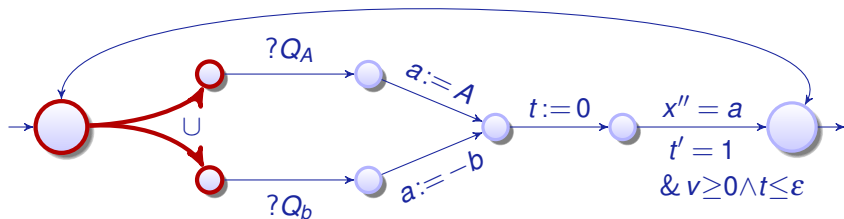


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_B; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

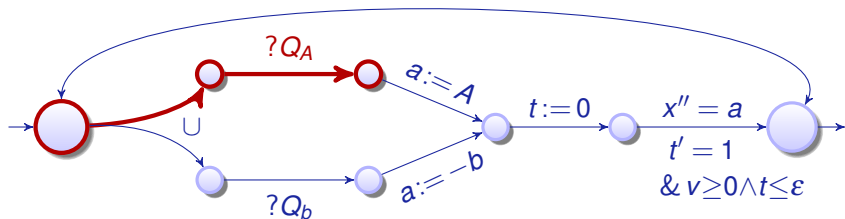


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

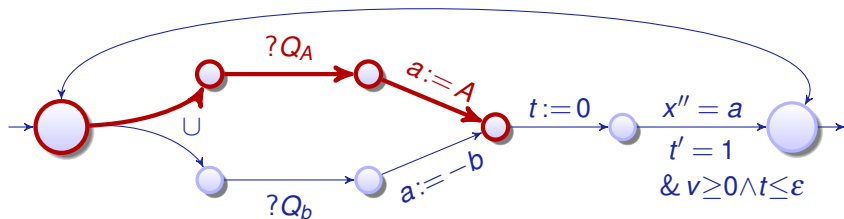


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

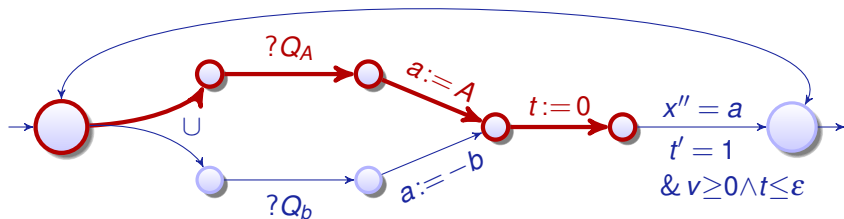


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

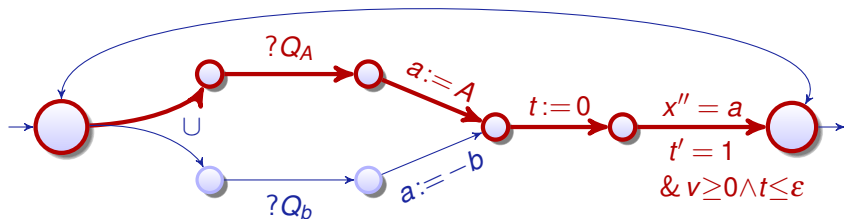


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_B; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

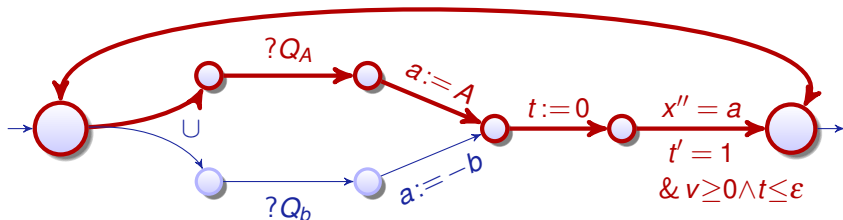


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \ \& \ v \geq 0 \wedge t \leq \epsilon\}$

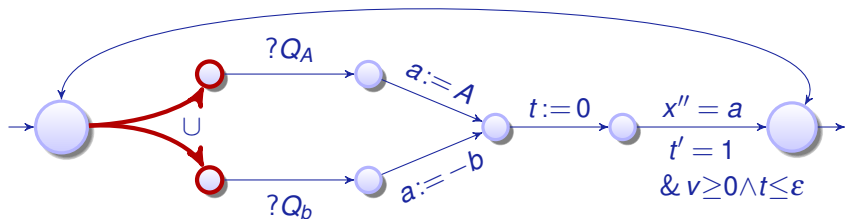


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

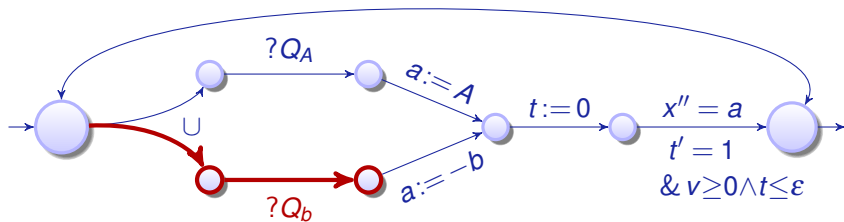


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

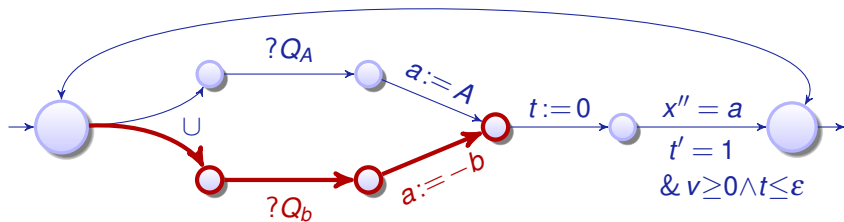


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

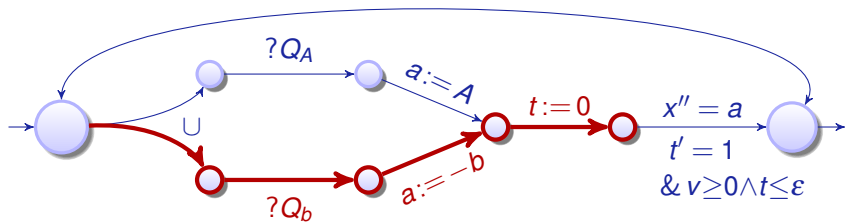


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

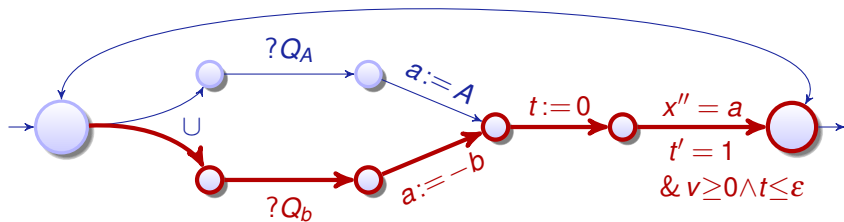


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

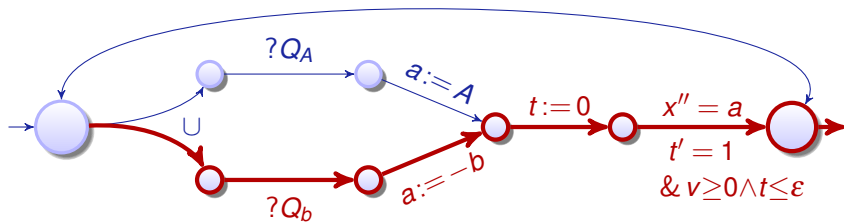


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$

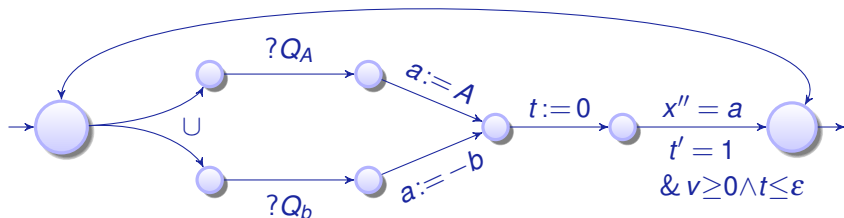


Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$



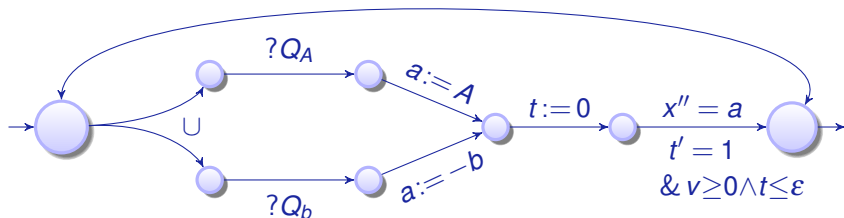
if(Q) α else $\beta \equiv$
 while(Q) $\alpha \equiv$

Robot \equiv (ctrl; drive)*

ctrl \equiv (? Q_A ; $a := A$)

\cup (? Q_B ; $a := -b$)

drive \equiv $t := 0$; { $x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon$ }



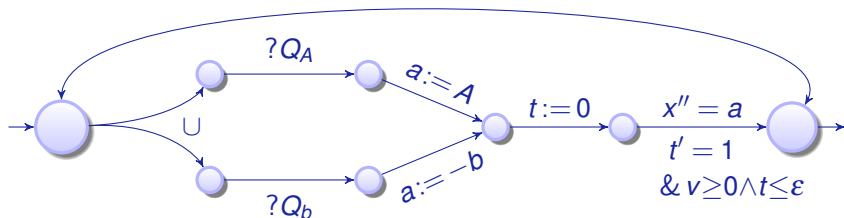
if(Q) α else $\beta \equiv (?Q; \alpha) \cup (? \neg Q; \beta)$
 while(Q) $\alpha \equiv$

Robot $\equiv (\text{ctrl}; \text{drive})^*$

ctrl $\equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

drive $\equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$



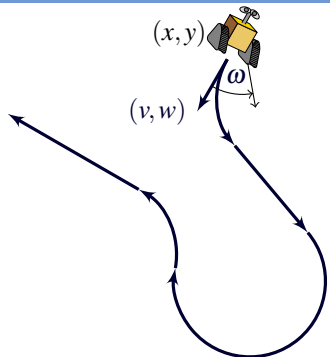
$\text{if}(Q) \alpha \text{ else } \beta \equiv (?Q; \alpha) \cup (? \neg Q; \beta)$
 $\text{while}(Q) \alpha \equiv (?Q; \alpha)^*; ? \neg Q$

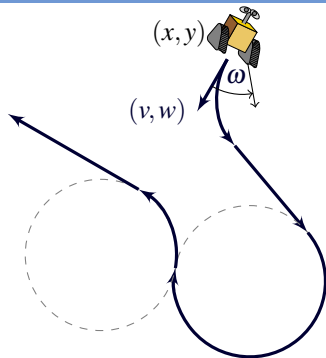
$\text{Robot} \equiv (\text{ctrl}; \text{drive})^*$

$\text{ctrl} \equiv (?Q_A; a := A)$

$\cup (?Q_b; a := -b)$

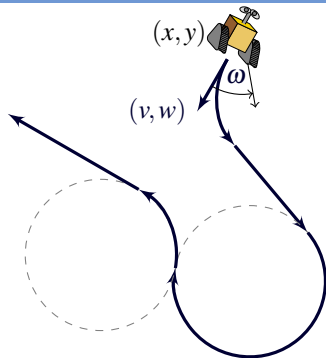
$\text{drive} \equiv t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \wedge t \leq \epsilon\}$





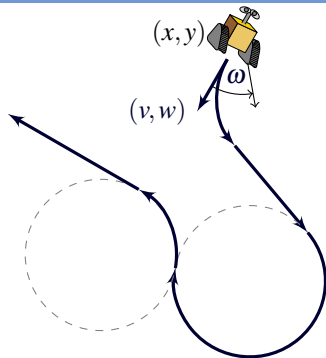
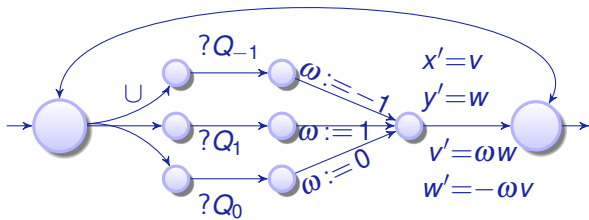
Example (Runaround Robot)

$$((\omega := -1 \cup \omega := 1 \cup \omega := 0); \\ \{x' = v, y' = w, v' = \omega w, w' = -\omega v\})^*$$



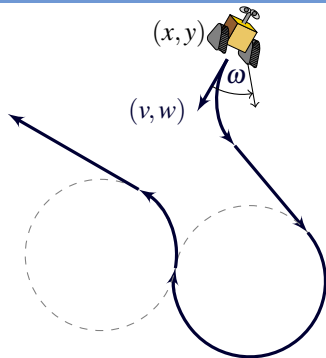
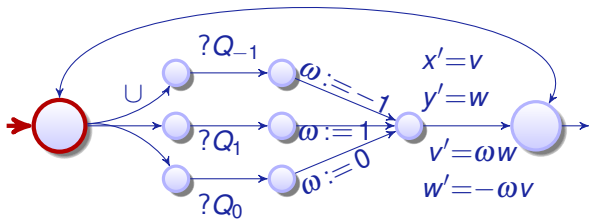
Example (Runaround Robot)

$$((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \\ \{x' = v, y' = w, v' = \omega w, w' = -\omega v\})^*$$



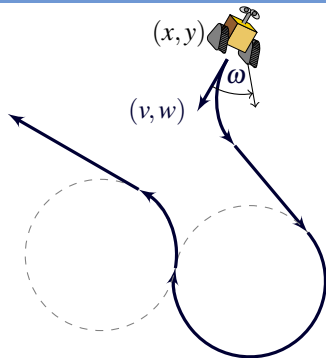
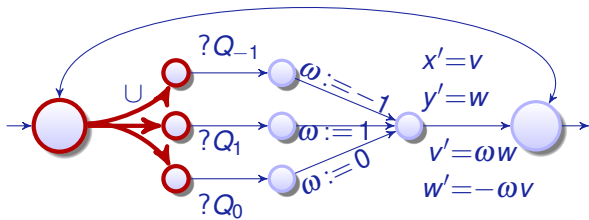
Example (Runaround Robot)

$$((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \{x' = v, y' = w, v' = \omega w, w' = -\omega v\})^*$$



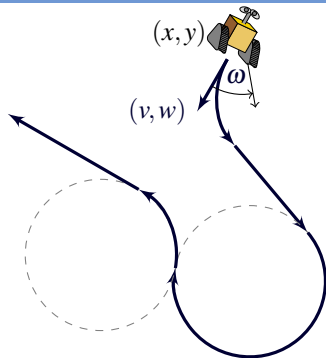
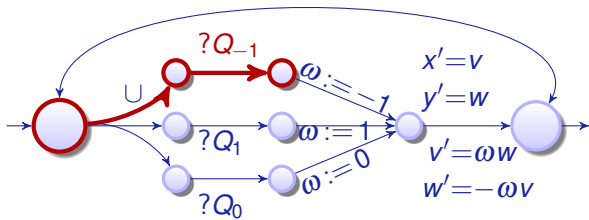
Example (Runaround Robot)

$$\left((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \right. \\ \left. \{x' = v, y' = w, v' = \omega w, w' = -\omega v\} \right)^*$$



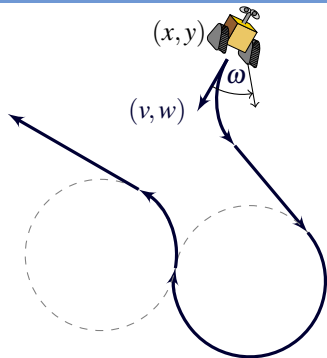
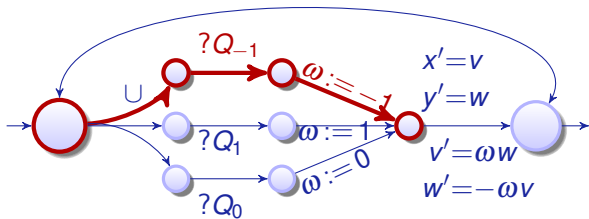
Example (Runaround Robot)

$$\begin{aligned} &((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \\ &\{x' = v, y' = w, v' = \omega w, w' = -\omega v\})^* \end{aligned}$$



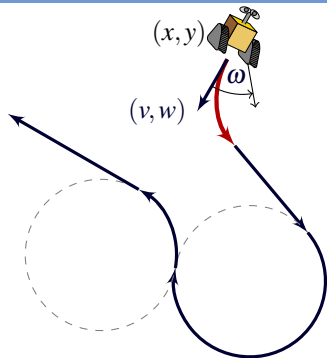
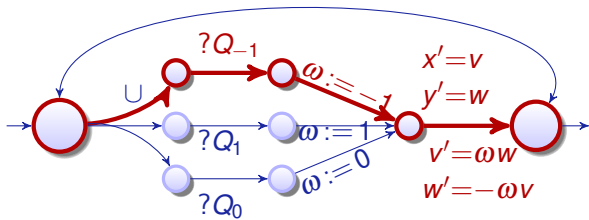
Example (Runaround Robot)

$$((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \\ \{x' = v, y' = w, v' = \omega w, w' = -\omega v\})^*$$



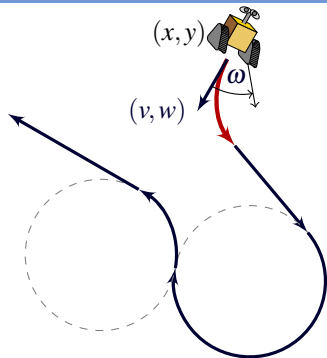
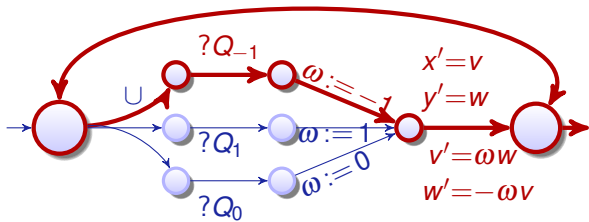
Example (Runaround Robot)

$$((?Q_{-1}; \omega := -1 U ?Q_1; \omega := 1 U ?Q_0; \omega := 0); \{x' = v, y' = w, v' = \omega w, w' = -\omega v\})^*$$



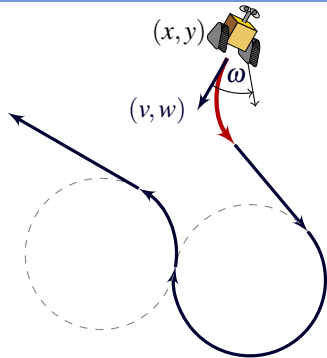
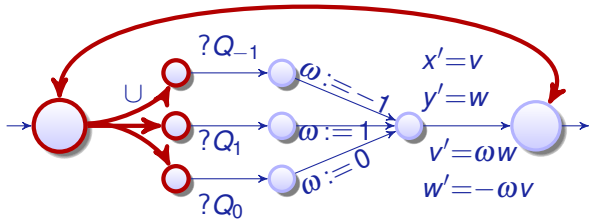
Example (Runaround Robot)

$$\left((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \right. \\
 \left. \{x' = v, y' = w, v' = \omega w, w' = -\omega v\} \right)^*$$



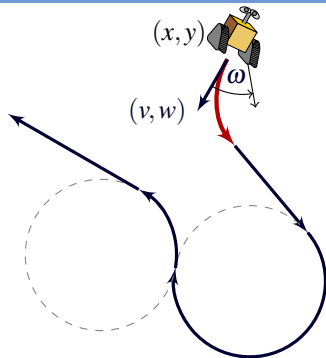
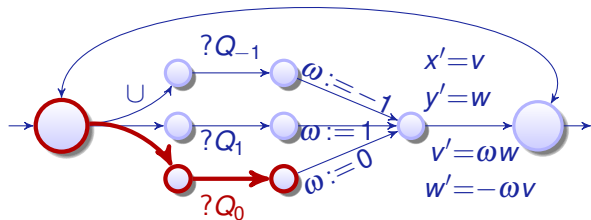
Example (Runaround Robot)

$$\left((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \right. \\
 \left. \{x' = v, y' = w, v' = \omega w, w' = -\omega v\} \right)^*$$



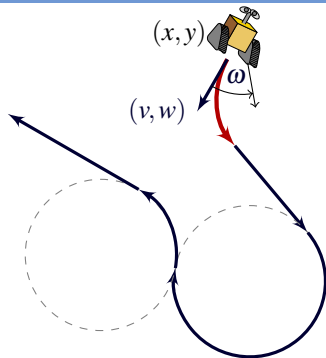
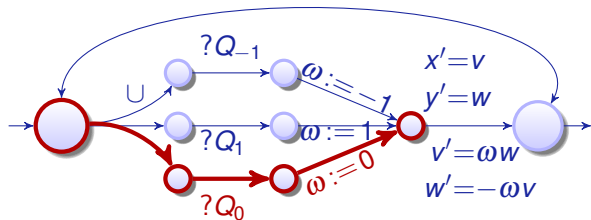
Example (Runaround Robot)

$$((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \{x' = v, y' = w, v' = \omega w, w' = -\omega v\})^*$$



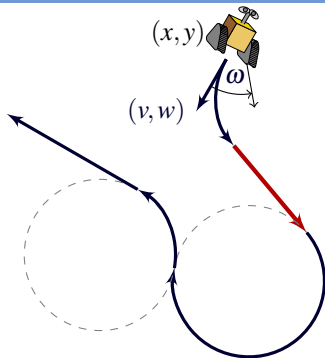
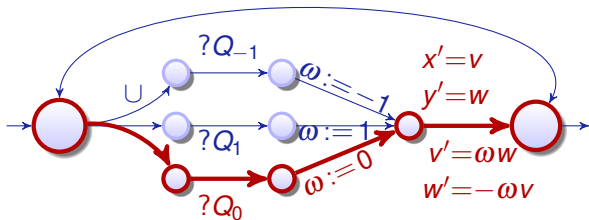
Example (Runaround Robot)

$$\begin{aligned}
 & ((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \\
 & \{x' = v, y' = w, v' = \omega w, w' = -\omega v\})^*
 \end{aligned}$$



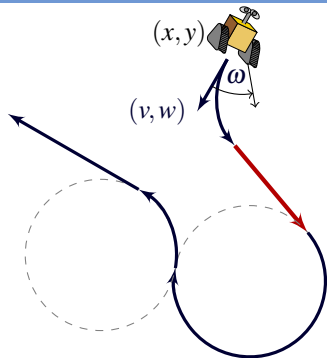
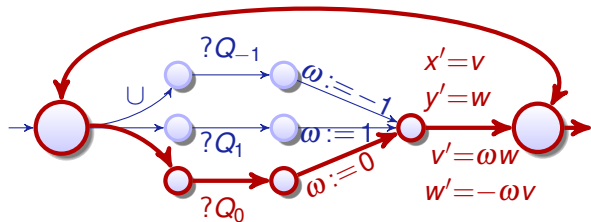
Example (Runaround Robot)

$$\left((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \right. \\
 \left. \{x' = v, y' = w, v' = \omega w, w' = -\omega v\} \right)^*$$



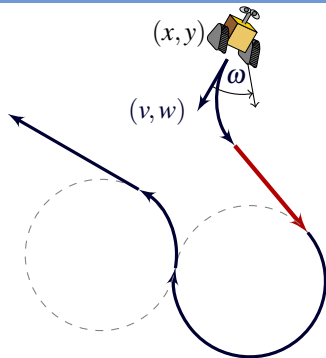
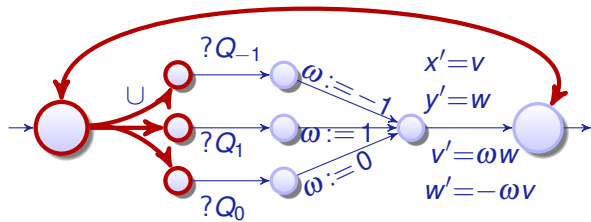
Example (Runaround Robot)

$$\left((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \right. \\ \left. \{x' = v, y' = w, v' = \omega w, w' = -\omega v\} \right)^*$$



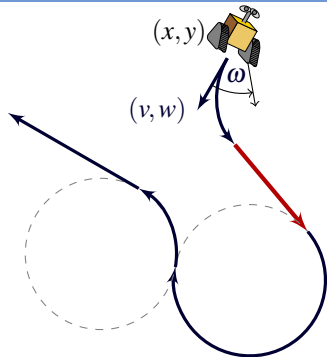
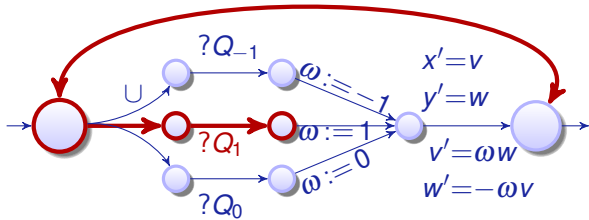
Example (Runaround Robot)

$$\left((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \right. \\
 \left. \{x' = v, y' = w, v' = \omega w, w' = -\omega v\} \right)^*$$



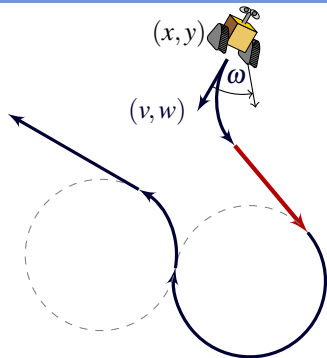
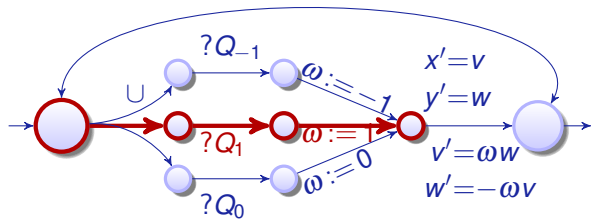
Example (Runaround Robot)

$$\left((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \right. \\ \left. \{x' = v, y' = w, v' = \omega w, w' = -\omega v\} \right)^*$$



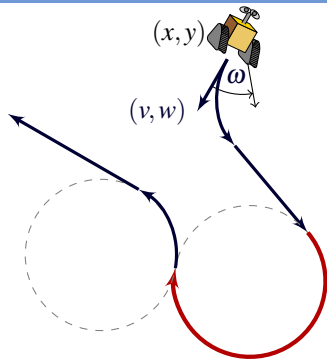
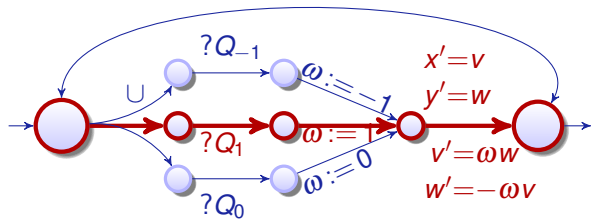
Example (Runaround Robot)

$$\left((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \right. \\
 \left. \{x' = v, y' = w, v' = \omega w, w' = -\omega v\} \right)^*$$



Example (Runaround Robot)

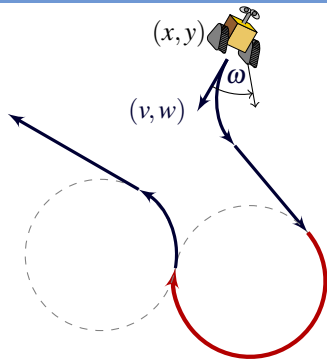
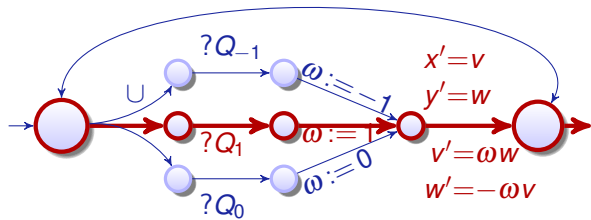
$$\begin{aligned}
 & ((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \\
 & \{x' = v, y' = w, v' = \omega w, w' = -\omega v\})^*
 \end{aligned}$$



Example (Runaround Robot)

$((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0);$

$\{x' = v, y' = w, v' = \omega w, w' = -\omega v\})^*$



Example (Runaround Robot)

$$\left((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0); \right. \\ \left. \{x' = v, y' = w, v' = \omega w, w' = -\omega v\} \right)^*$$

Example (Speedy the point)

$(?v < 4; a := a + 1 \cup a := -b);$

$\{x' = v, v' = a\};$

$(?v < 4; a := a + 1 \cup a := -b);$

$\{x' = v, v' = a\};$

$(?v < 4; a := a + 1 \cup a := -b);$

$\{x' = v, v' = a\}$

Example (Speedy the point)

? $v < 4$; $a := a + 1$;

{ $x' = v, v' = a$ };

? $v < 4$; $a := a + 1$;

{ $x' = v, v' = a$ };

? $v < 4$; $a := a + 1$;

{ $x' = v, v' = a$ }

Example (Speedy the point)

$?v < 4; a := a + 1;$

$\{x' = v, v' = a\};$

$?v < 4; a := a + 1;$

$\{x' = v, v' = a\};$

$?v < 4; a := a + 1;$

$\{x' = v, v' = a\}$

No wait, now it's a bad model! The HP assumes the test $v < 4$ passes after each ODE. No other choices are available.

Don't let your controller discard important cases!

- 1 Learning Objectives
- 2 Gradual Introduction to Hybrid Programs
- 3 Hybrid Programs
 - Syntax
 - Semantics
 - Notational Convention
- 4 Examples
- 5 Summary

Definition (Syntax of hybrid program α)

$$\alpha, \beta ::= x := e \mid ?Q \mid x' = f(x) \& Q \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^*$$

Definition (Semantics of hybrid programs) $(\llbracket \cdot \rrbracket : \text{HP} \rightarrow \wp(\mathcal{I} \times \mathcal{I}))$

$$\llbracket x := e \rrbracket = \{(\omega, \nu) : \nu = \omega \text{ except } \nu[x] = \omega[e]\}$$

$$\llbracket ?Q \rrbracket = \{(\omega, \omega) : \omega \models Q\}$$

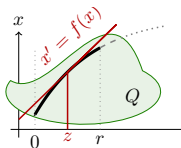
$$\llbracket x' = f(x) \rrbracket = \{(\varphi(0), \varphi(r)) : \varphi \models x' = f(x) \text{ for some duration } r \geq 0\}$$

$$\llbracket \alpha \cup \beta \rrbracket = \llbracket \alpha \rrbracket \cup \llbracket \beta \rrbracket$$

$$\llbracket \alpha; \beta \rrbracket = \llbracket \alpha \rrbracket \circ \llbracket \beta \rrbracket$$

$$\llbracket \alpha^* \rrbracket = \llbracket \alpha \rrbracket^* = \bigcup_{n \in \mathbb{N}} \llbracket \alpha^n \rrbracket$$

compositional





André Platzer.

Logical Foundations of Cyber-Physical Systems.

Springer, Cham, 2018.

doi:10.1007/978-3-319-63588-0.



André Platzer.

Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics.

Springer, Heidelberg, 2010.

doi:10.1007/978-3-642-14509-4.



André Platzer.

Logics of dynamical systems.

In *LICS*, pages 13–24, Los Alamitos, 2012. IEEE.

doi:10.1109/LICS.2012.13.



André Platzer.

Differential dynamic logic for hybrid systems.

J. Autom. Reas., 41(2):143–189, 2008.

doi:10.1007/s10817-008-9103-8.



André Platzer.

A complete uniform substitution calculus for differential dynamic logic.

J. Autom. Reas., 59(2):219–265, 2017.

[doi:10.1007/s10817-016-9385-1](https://doi.org/10.1007/s10817-016-9385-1).