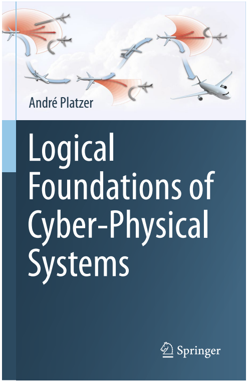
03: Choice & Control

Logical Foundations of Cyber-Physical Systems



Stefan Mitsch

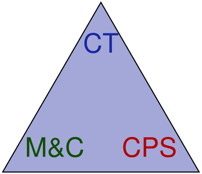
Carnegie Mellon University
Computer Science Department

Outline

Outline

Learning Objectives: Choice & Control

nondeterminism
abstraction
programming languages for CPS
semantics
compositionality



models
core principles
discrete+continuous

operational effect
operational precision

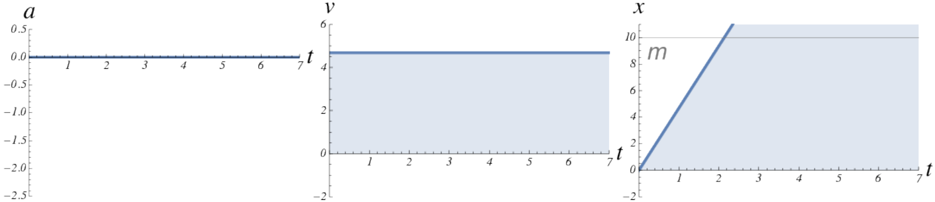
Outline

Example (Point mass motion)

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

Purely continuous dynamics

What about discrete dynamics?

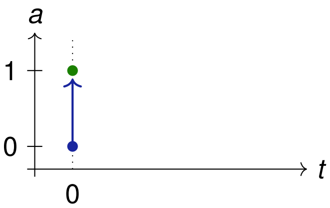


Example (Point mass motion)

$$a := a + 1$$

Purely discrete dynamics

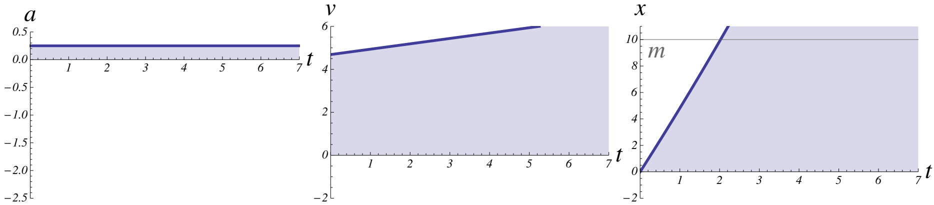
How do both meet?



Example (Point mass motion)

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

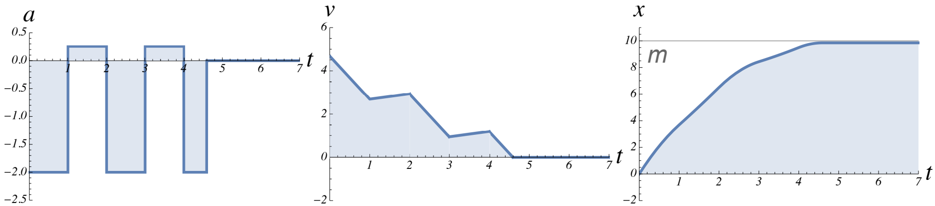
Hybrid dynamics, i.e., composition of continuous and discrete dynamics
Here: sequential composition first;second



Playing with Acceleration and Braking

Example (Point mass motion)

$$
\begin{aligned}
&a := -2;\ \{x' = v, v' = a\};\\
&a := 0.25; \{x' = v, v' = a\};\\
&a := -2;\ \{x' = v, v' = a\};\\
&a := 0.25; \{x' = v, v' = a\};\\
&a := -2;\ \{x' = v, v' = a\};\\
&a := 0.25; \{x' = v, v' = a\}
\end{aligned}
$$

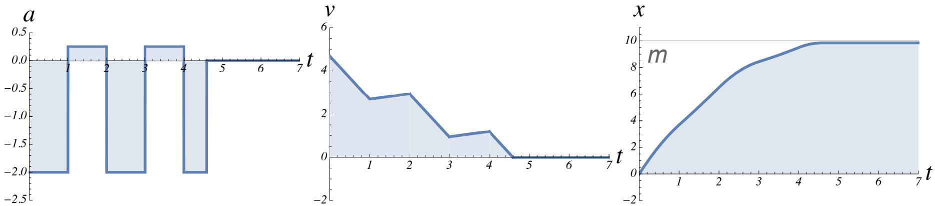


Playing with Acceleration and Braking

Example (Point mass motion)

$$
\begin{aligned}
&a := -2;\ \{x' = v, v' = a\};\\
&a := 0.25;\ \{x' = v, v' = a\};\\
&a := -2;\ \{x' = v, v' = a\};\\
&a := 0.25;\ \{x' = v, v' = a\};\\
&a := -2;\ \{x' = v, v' = a\};\\
&a := 0.25;\ \{x' = v, v' = a\}
\end{aligned}
$$

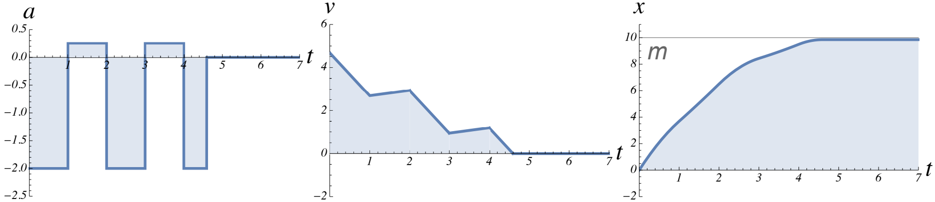
How long to follow an ODE?



Example (Point mass motion)

$$a := -2; \ \{x' = v, v' = a\};$$
$$a := 0.25; \{x' = v, v' = a\};$$
$$a := -2; \ \{x' = v, v' = a\};$$
$$a := 0.25; \{x' = v, v' = a\};$$
$$a := -2; \ \{x' = v, v' = a\};$$
$$a := 0.25; \{x' = v, v' = a\}$$

How to check conditions before actions?



Playing with Acceleration and Braking

Example (Point mass motion)

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

Velocity-dependent control

Playing with Acceleration and Braking

Example (Point mass motion)

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

Distance-dependent control for obstacle m

Playing with Acceleration and Braking

Example (Point mass motion)

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

Velocity and distance-dependent control

Iterative Design

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

Playing with Acceleration and Braking

Example (Point mass motion)

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

Also only accelerate if it's comfortable to do so

Example (Point mass motion)

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

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

Playing with Acceleration and Braking

Example (Point mass motion)

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

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.

Playing with Acceleration and Braking

Example (Point mass motion)

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

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

Oops, now it got too simple! Not every choice is always acceptable.

Playing with Acceleration and Braking

Example (Point mass motion)

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

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

Playing with Acceleration and Braking

Example (Point mass motion)

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

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

$v := v+1;\ ?v < 4$ only runs if

Playing with Acceleration and Braking

Example (Point mass motion)

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

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

Playing with Acceleration and Braking

Example (Point mass motion)

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

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

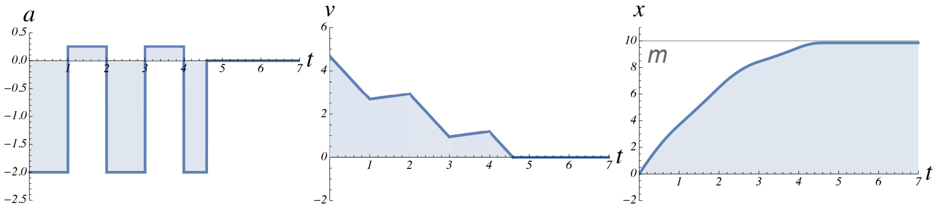
Broader significance of nondeterminism

Nondeterminism is a tool for abstraction to focus on critical aspects.
Nondeterminism is essential to describe imperfectly known environment.

Example (Point mass motion)

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

Nondeterministic repetition * repeats *any* arbitrary number of times



Outline

Hybrid Programs: Syntax

Definition (Syntax of hybrid program α)

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

Hybrid Programs: Syntax

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

Hybrid Programs: Syntax

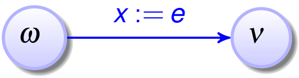
Definition (Syntax of hybrid program α)

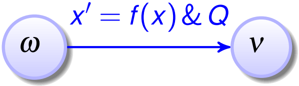
$$\alpha, \beta ::= x := e \mid ?Q \mid x' = f(x) \,\&\, Q \mid \alpha \cup \beta \mid \alpha ; \beta \mid \alpha^*$$

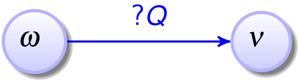
- Discrete Assign
- Test Condition
- Differential Equation
- Nondet. Choice
- Seq. Compose
- Nondet. Repeat

Like regular expressions. Everything nondeterministic

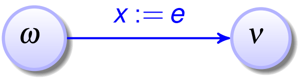
Hybrid Programs: Semantics

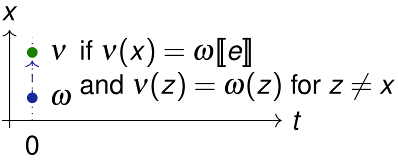


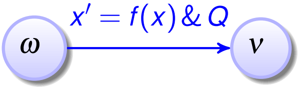


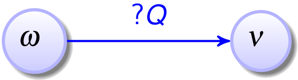


Hybrid Programs: Semantics

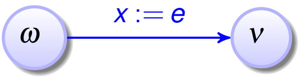


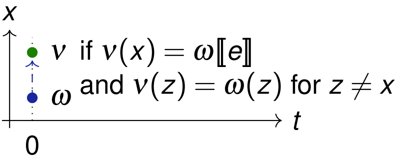


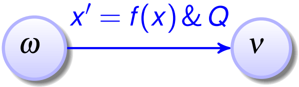


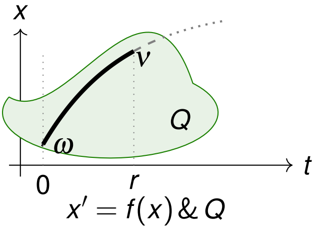


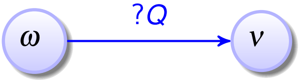
Hybrid Programs: Semantics

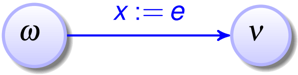


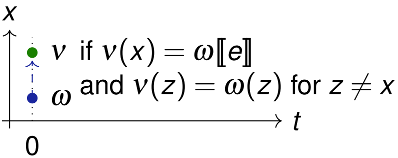


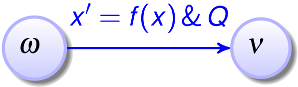


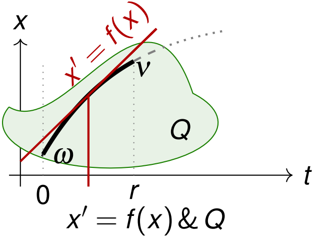


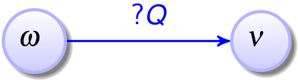



ω
$x := e$
ν

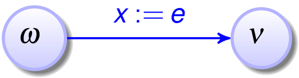

x
ν if $\nu(x) = \omega[\![e]\!]$
ω and $\nu(z) = \omega(z)$ for $z \neq x$
t
0

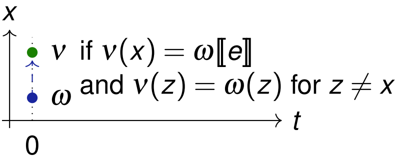

ω
$x' = f(x) \,\&\, Q$
ν

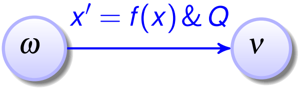

x
$x' = f(x)$
ν
Q
ω
t
0
r
$x' = f(x) \,\&\, Q$

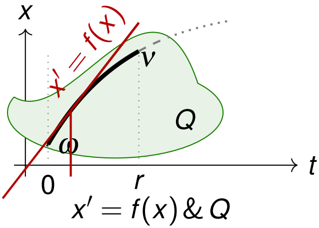

ω
$?Q$
ν

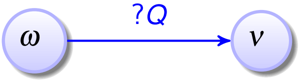
Hybrid Programs: Semantics



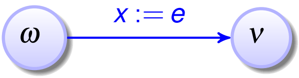


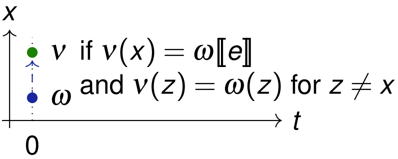


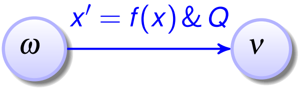


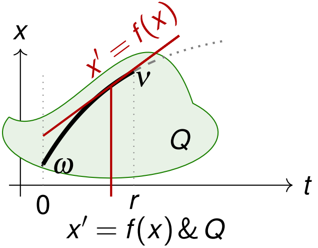


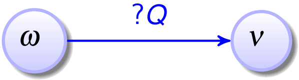
Hybrid Programs: Semantics



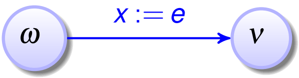


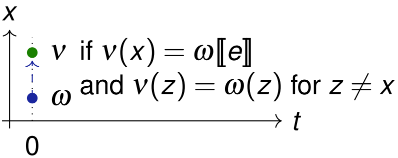


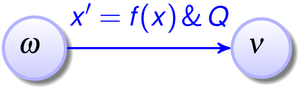


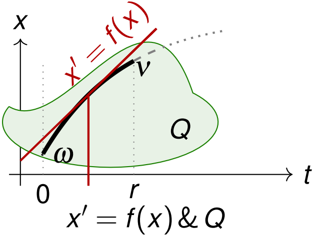


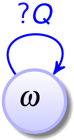
Hybrid Programs: Semantics



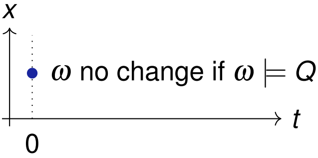




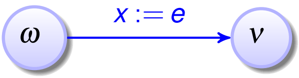


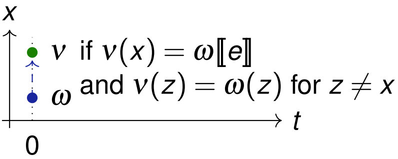


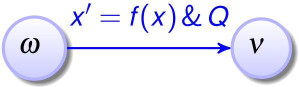
if $\omega \models Q$

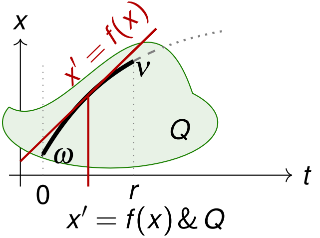


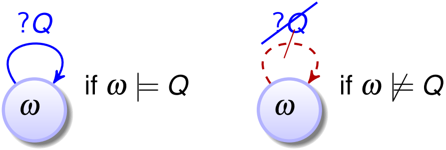
Hybrid Programs: Semantics

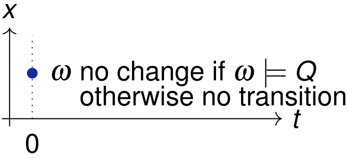




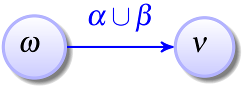


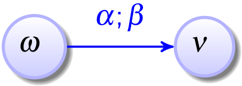


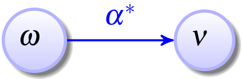




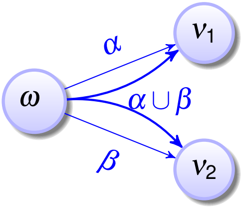
Hybrid Programs: Semantics

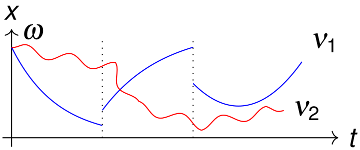


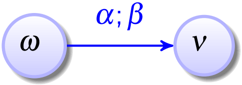


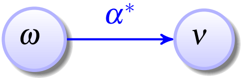


Hybrid Programs: Semantics

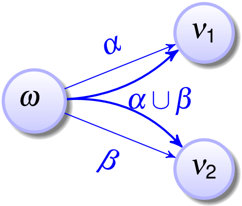


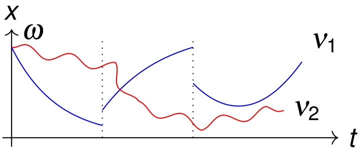


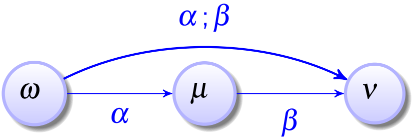


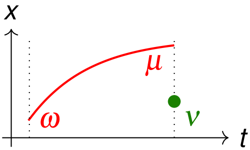


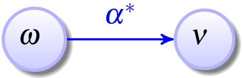
Hybrid Programs: Semantics



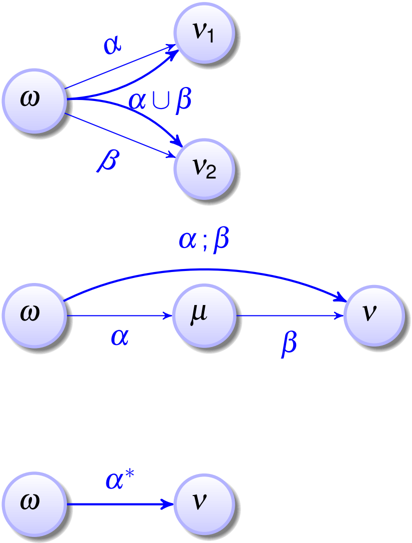


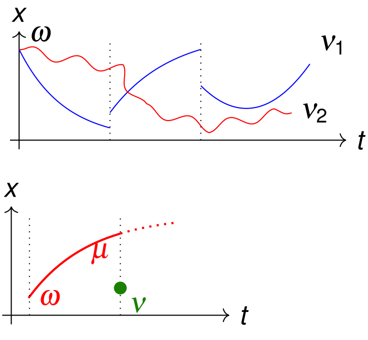




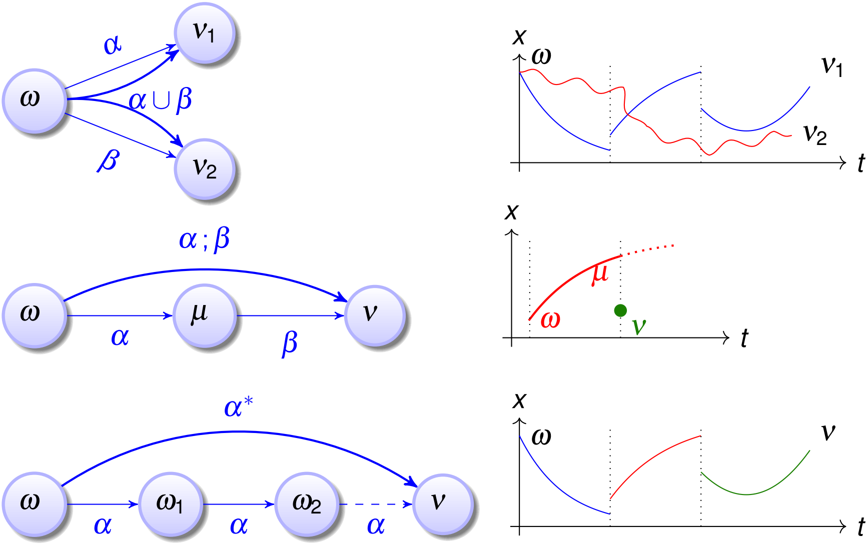


Hybrid Programs: Semantics

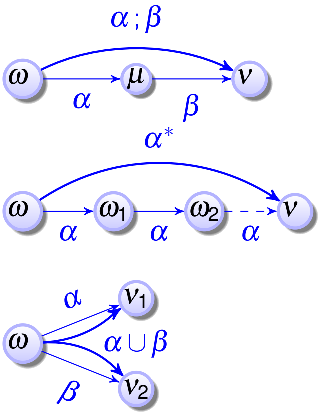




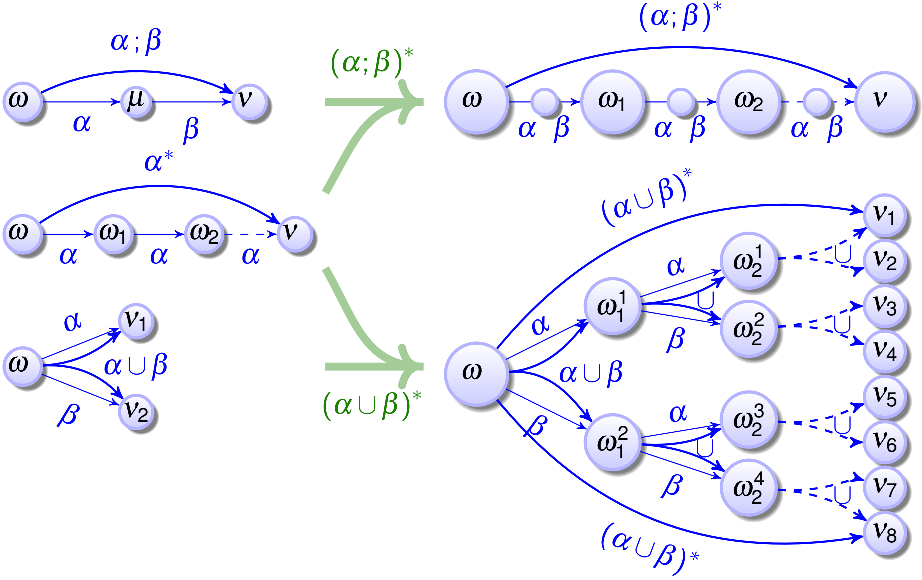
Hybrid Programs: Semantics



Plug-in for Semantics of Composed Hybrid Programs



Plug-in for Semantics of Composed Hybrid Programs



Hybrid Programs: Syntax & Semantics

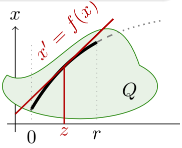
Definition (Syntax of hybrid program α)

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

Definition (Semantics of hybrid programs) ($[\![\cdot]\!]$: HP $\to \wp(\mathscr{S} \times \mathscr{S})$)

$$\begin{aligned}
[\![x := e]\!] &= \{(\omega,\nu) \ : \ \nu = \omega \text{ except } \nu[\![x]\!] = \omega[\![e]\!]\} \\
[\![?Q]\!] &= \{(\omega,\omega) \ : \ \omega \models Q\} \\
[\![x' = f(x)]\!] &= \{(\varphi(0),\varphi(r)) \ : \ \varphi \models x' = f(x) \text{ for some duration } r \geq 0\} \\
[\![\alpha \cup \beta]\!] &= [\![\alpha]\!] \cup [\![\beta]\!] \\
[\![\alpha;\beta]\!] &= [\![\alpha]\!] \circ [\![\beta]\!] = \{(\omega,\nu) \ : \ (\omega,\mu) \in [\![\alpha]\!] \text{ and } (\mu,\nu) \in [\![\beta]\!]\} \\
[\![\alpha^*]\!] &= [\![\alpha]\!]^* = \bigcup_{n \in \mathbb{N}} [\![\alpha^n]\!] \qquad \alpha^n \equiv \underbrace{\alpha;\alpha;\alpha;\ldots;\alpha}_{n \text{ times}}
\end{aligned}$$

compositional



Hybrid Programs: Syntax & Semantics

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) $([\![\cdot]\!] : \text{HP} \to \wp(\mathscr{S} \times \mathscr{S}))$

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

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

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

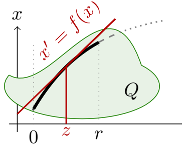
$$[\![\alpha \cup \beta]\!] = [\![\alpha]\!] \cup [\![\beta]\!]$$

$$[\![\alpha ; \beta]\!] = [\![\alpha]\!] \circ [\![\beta]\!]$$

$$[\![\alpha^*]\!] = [\![\alpha]\!]^* = \bigcup_{n \in \mathbb{N}} [\![\alpha^n]\!]$$

compositional

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



Notational Conventions: Names

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

BUT: in CPS applications, our names will follow application
for example: x position, v velocity, and a acceleration variables

Notational Conventions: Precedence

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

Notational Conventions: Precedence

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)

$\forall x\, P \wedge Q \equiv$ $\qquad$ $\forall x\, P \rightarrow Q \equiv$ $\qquad$.

Notational Conventions: Precedence

Convention (Operator Precedence)

1. Unary operators (including *, $\neg$ and $\forall x, \exists x$) bind stronger than binary.
2. $\land$ binds stronger than $\lor$, 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)

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

Notational Conventions: Precedence

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)

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

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)

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

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

Notational Conventions: Precedence

Convention (Operator Precedence)

1. Unary operators (including *, $\neg$ and $\forall x, \exists x$) bind stronger than binary.
2. $\land$ binds stronger than $\lor$, 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)

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

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

Notational Conventions: Precedence

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)

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

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

Notational Conventions: Precedence

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)

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

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

Notational Conventions: Precedence

Convention (Operator Precedence)

1. Unary operators (including *, $\neg$ and $\forall x, \exists x$) bind stronger than binary.
2. $\land$ binds stronger than $\lor$, 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)

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

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

$P \rightarrow Q \rightarrow R \equiv$

Notational Conventions: Precedence

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)

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

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

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

Notational Conventions: Precedence

Convention (Operator Precedence)

1. Unary operators (including *, $\neg$ and $\forall x, \exists x$) bind stronger than binary.
2. $\land$ binds stronger than $\lor$, 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)

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

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

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

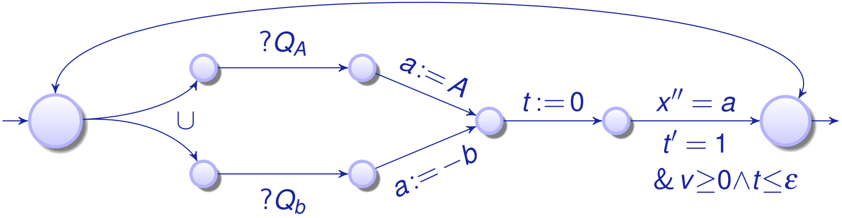
But $\rightarrow, \leftrightarrow$ expect explicit parentheses. Illegal: $P \rightarrow Q \leftrightarrow R$ $\qquad$ $P \leftrightarrow Q \rightarrow R$

Outline

Branching Transition Structure in Hybrid Programs

$$
\begin{aligned}
\text{Robot} &\equiv (\text{ctrl}\,;\text{drive})^* \\
\text{ctrl} &\equiv (?Q_A; a := A) \\
&\quad \cup (?Q_b; a := -b) \\
\text{drive} &\equiv t := 0; \{x' = v, v' = a, t' = 1 \,\&\, v \geq 0 \land t \leq \varepsilon\}
\end{aligned}
$$

Branching Transition Structure in Hybrid Programs



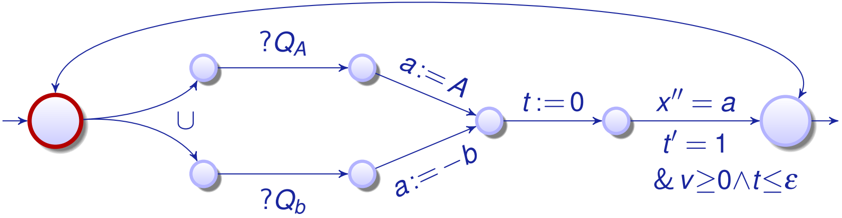
$$\text{Robot} \equiv (\text{ctrl} \,;\text{drive})^*$$

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

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

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

Branching Transition Structure in Hybrid Programs

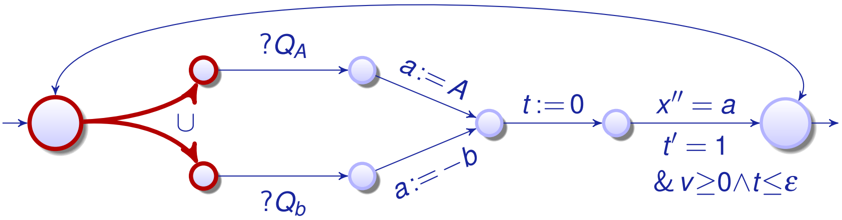


$$\text{Robot} \equiv (\text{ctrl}\,;\text{drive})^*$$

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

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

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



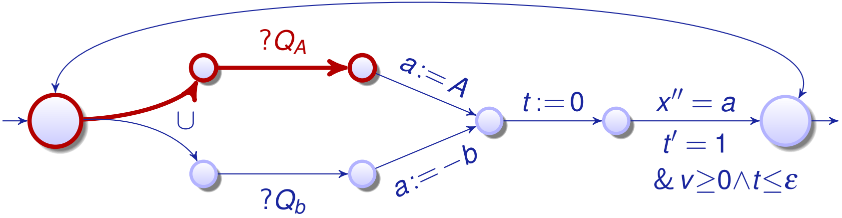
$$\text{Robot} \equiv (\text{ctrl}\,;\text{drive})^*$$

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

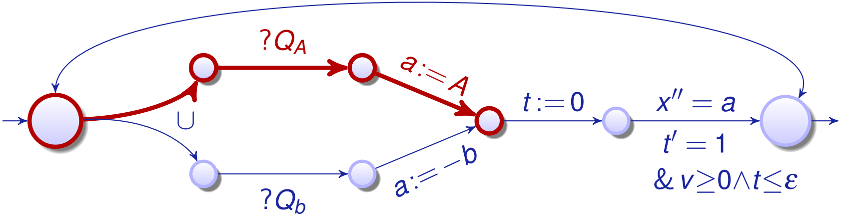
$$\cup\, (?Q_b; a := -b)$$

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

Branching Transition Structure in Hybrid Programs

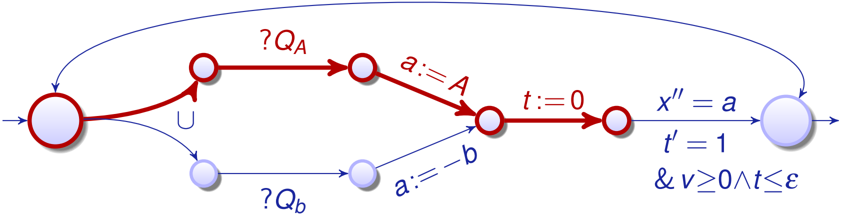


$$\begin{aligned}
\text{Robot} &\equiv (\text{ctrl}\,;\text{drive})^* \\
\text{ctrl} &\equiv (?Q_A; a := A) \\
&\quad \cup (?Q_b; a := -b) \\
\text{drive} &\equiv t := 0; \{x' = v, v' = a, t' = 1 \,\&\, v \geq 0 \wedge t \leq \varepsilon\}
\end{aligned}$$



$$\text{Robot} \equiv (\text{ctrl}\,;\text{drive})^*$$
$$\text{ctrl} \equiv (?Q_A; a := A)$$
$$\cup\,(?Q_b; a := -b)$$
$$\text{drive} \equiv t := 0; \{x' = v, v' = a, t' = 1 \,\&\, v \geq 0 \land t \leq \varepsilon\}$$

Branching Transition Structure in Hybrid Programs

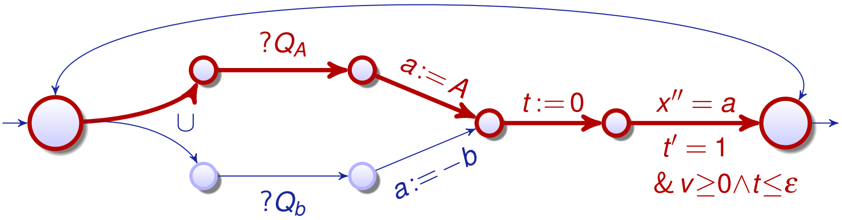


$$\text{Robot} \equiv (\text{ctrl}\,;\text{drive})^*$$

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

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

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

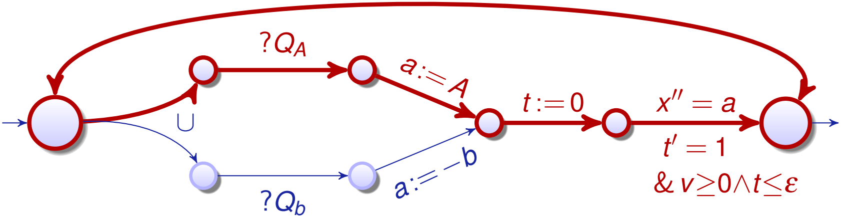


$$\text{Robot} \equiv (\text{ctrl}\,;\text{drive})^*$$

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

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

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

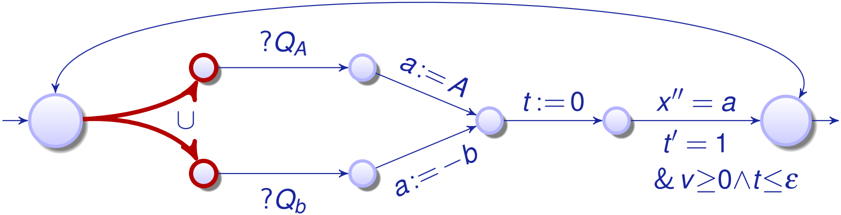


$\text{Robot} \equiv (\text{ctrl}\,;\text{drive})^*$

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

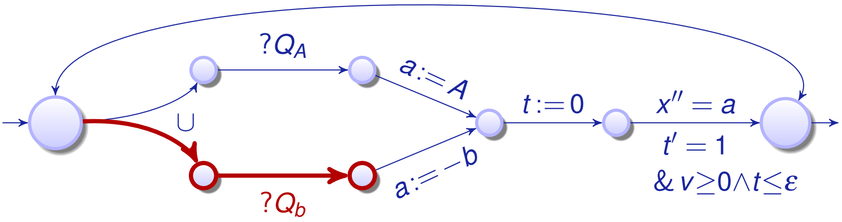
$\cup (?Q_b; a := -b)$

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

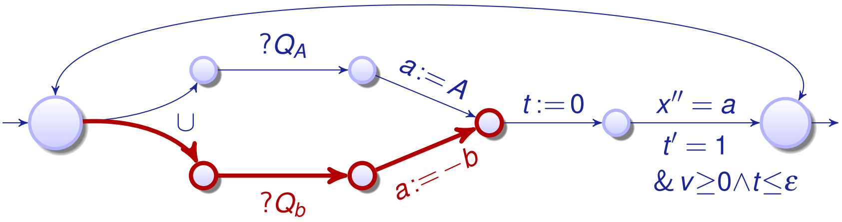


$$\begin{aligned}
\text{Robot} &\equiv (\text{ctrl}\,;\text{drive})^* \\
\text{ctrl} &\equiv (?Q_A; a := A) \\
&\quad \cup (?Q_b; a := -b) \\
\text{drive} &\equiv t := 0; \{x' = v, v' = a, t' = 1 \;\&\; v \geq 0 \land t \leq \varepsilon\}
\end{aligned}$$

Branching Transition Structure in Hybrid Programs



$$\begin{aligned}
\text{Robot} &\equiv (\text{ctrl}\,;\text{drive})^* \\
\text{ctrl} &\equiv (?Q_A; a := A) \\
&\quad \cup (?Q_b; a := -b) \\
\text{drive} &\equiv t := 0; \{x' = v, v' = a, t' = 1 \,\&\, v \geq 0 \wedge t \leq \varepsilon\}
\end{aligned}$$



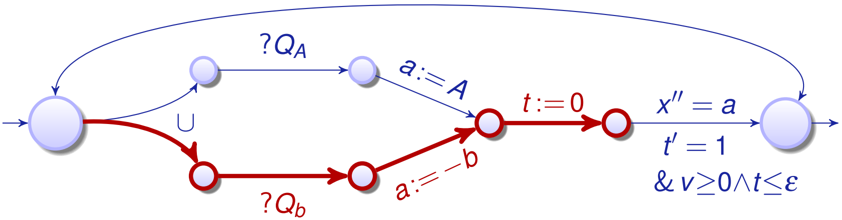
$$\text{Robot} \equiv (\text{ctrl}\,;\text{drive})^*$$

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

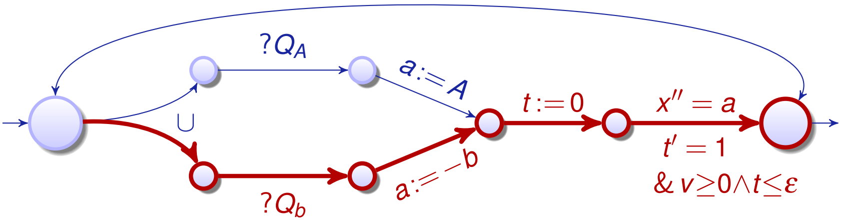
$$\cup\, (?Q_b;\, a := -b)$$

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

Branching Transition Structure in Hybrid Programs



$$
\begin{aligned}
\text{Robot} &\equiv (\text{ctrl}\,;\text{drive})^*\\
\text{ctrl} &\equiv (?Q_A; a:=A)\\
&\quad \cup (?Q_b; a:=-b)\\
\text{drive} &\equiv t:=0; \{x'=v, v'=a, t'=1 \,\&\, v\geq 0 \land t\leq \varepsilon\}
\end{aligned}
$$



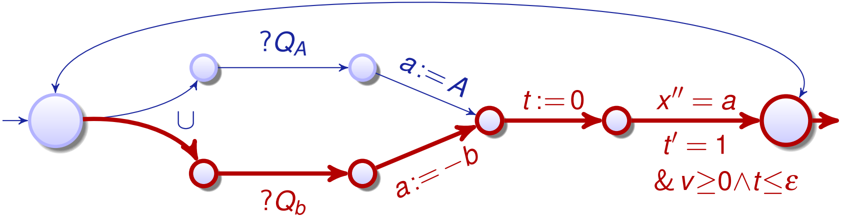
$$\text{Robot} \equiv (\text{ctrl}\,;\text{drive})^*$$

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

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

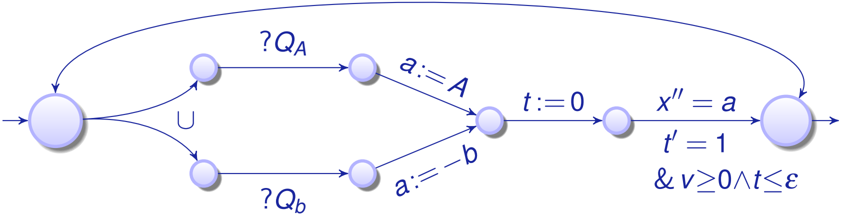
$$\text{drive} \equiv t := 0; \{x' = v, v' = a, t' = 1 \,\&\, v \geq 0 \land t \leq \varepsilon\}$$

Branching Transition Structure in Hybrid Programs



$$
\begin{aligned}
\text{Robot} &\equiv (\text{ctrl}\,;\text{drive})^* \\
\text{ctrl} &\equiv (?Q_A; a := A) \\
&\quad \cup (?Q_b; a := -b) \\
\text{drive} &\equiv t := 0; \{x' = v, v' = a, t' = 1 \,\&\, v \geq 0 \wedge t \leq \varepsilon\}
\end{aligned}
$$

Branching Transition Structure in Hybrid Programs



$$\text{if}(Q)\,\alpha\,\text{else}\,\beta \equiv$$

$$\text{while}(Q)\,\alpha \equiv$$

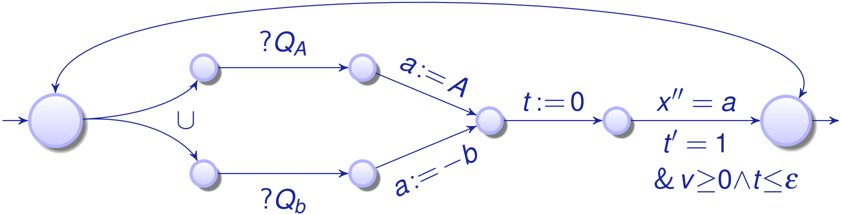
$$\text{Robot} \equiv (\text{ctrl}\,;\text{drive})^*$$

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

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

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

Branching Transition Structure in Hybrid Programs



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

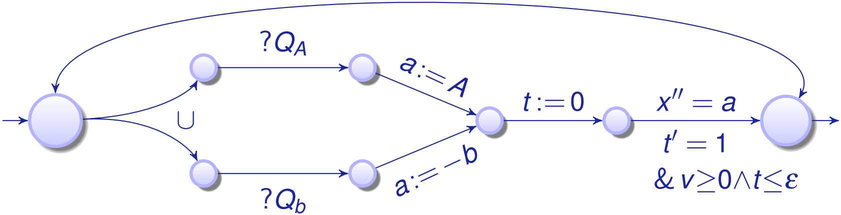
$$\text{while}(Q)\,\alpha \equiv$$

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

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

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

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



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

$$\text{while}(Q)\,\alpha \equiv (?Q;\alpha)^*;?\neg Q$$

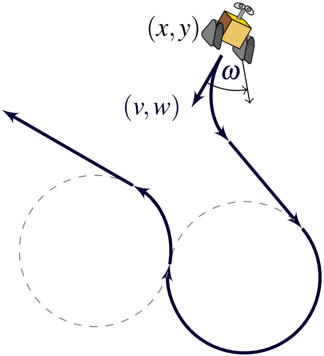
$$\text{Robot} \equiv (\text{ctrl};\text{drive})^*$$

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

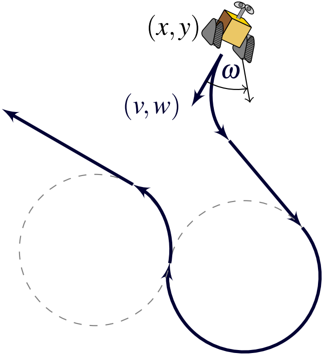
$$\cup (?Q_b; a := -b)$$

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

Runaround Robot with Dubins Paths



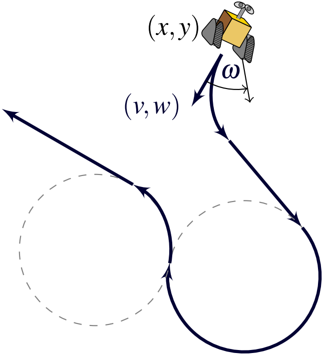
Runaround Robot with Dubins Paths



Example (Runaround Robot)

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

Runaround Robot with Dubins Paths



Example (Runaround Robot)

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

A Matter of Choice

Example (Recall Point Mass Motion)

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

A Matter of Choice: Remove Brakes

Example (Recall Point Mass Motion)

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

A Matter of Choice: Accidentally Discarded Runs!

Example (Recall Point Mass Motion)

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

Now it's a bad model!

The HP assumes the test $v < 4$ passes at the beginning of each loop iteration (so after each ODE). Other choices for backtracking are no longer available.

Don't let your controller discard important cases!

Outline

Hybrid Programs: Syntax & Semantics

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) $([\![\cdot]\!] : \mathrm{HP} \to \wp(\mathcal{S} \times \mathcal{S}))$

$$\begin{aligned}
[\![x := e]\!] &= \{(\omega, \nu) \ : \ \nu = \omega \text{ except } \nu[\![x]\!] = \omega[\![e]\!]\} \\
[\![?Q]\!] &= \{(\omega, \omega) \ : \ \omega \models Q\} \\
[\![x' = f(x)]\!] &= \{(\varphi(0), \varphi(r)) \ : \ \varphi \models x' = f(x) \text{ for some duration } r \geq 0\} \\
[\![\alpha \cup \beta]\!] &= [\![\alpha]\!] \cup [\![\beta]\!] \\
[\![\alpha;\beta]\!] &= [\![\alpha]\!] \circ [\![\beta]\!] \\
[\![\alpha^*]\!] &= [\![\alpha]\!]^* = \bigcup_{n \in \mathbb{N}} [\![\alpha^n]\!]
\end{aligned}$$

compositional