03: Choice & Control

Logical Foundations of Cyber-Physical Systems

André Platzer

Carnegie Mellon University
Computer Science Department
Outline

1. Learning Objectives

2. Gradual Introduction to Hybrid Programs

3. Hybrid Programs
   - Syntax
   - Semantics
   - Notational Convention

4. Examples

5. Summary
Outline

1 Learning Objectives

2 Gradual Introduction to Hybrid Programs

3 Hybrid Programs
   - Syntax
   - Semantics
   - Notational Convention

4 Examples

5 Summary
Learning Objectives

Choice & Control

- nondeterminism
- abstraction
- programming languages for CPS
- semantics
- compositionality

CT

M&C

CPS

models
core principles
discrete+
continuous

operational effect
operational precision

André Platzer (CMU)
Outline

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?
Playing with Acceleration and Braking

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)

\[
a := -2; \quad \{ x' = v, v' = a \}; \\
a := 0.25; \quad \{ x' = v, v' = a \}; \\
a := -2; \quad \{ x' = v, v' = a \}; \\
a := 0.25; \quad \{ x' = v, v' = a \}; \\
a := -2; \quad \{ x' = v, v' = a \}; \\
a := 0.25; \quad \{ x' = v, v' = a \}
\]
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?
Playing with Acceleration and Braking

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)

\[
\begin{align*}
  \text{if}(v < 4) & \quad a := a + 1 \quad \text{else} \quad a := -b; \\
  \{x' = v, \ v' = a\}
\end{align*}
\]

Velocity-dependent control
Example (Speedy the point)

\[
\text{if}(x - m > s) \quad a := a + 1 \quad \text{else} \quad a := -b;
\]

\[
\{ x' = v, \; v' = a \}
\]

Distance-dependent control for obstacle \( m \)
Example (Speedy the point)

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

Velocity and distance-dependent control

**Iterative Design**

Start as simple as possible, then add challenges once basics are correct.
Example (Speedy the point)

\[
\begin{align*}
    \text{if}(x - m > s \land v < 4 \land \text{efficiency}) \quad & a := a + 1 \quad \text{else} \quad a := -b; \\
    \{x' = v, \quad v' = a\}
\end{align*}
\]

Also only accelerate if it’s efficient to do so
Example (Speedy the point)

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

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.

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

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

**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)

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

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.

\[
\forall v < 4; v := v + 1 \quad \text{only runs if} \quad v < 4 \quad \text{initially true}
\]

\[
 v := v + 1; \forall v < 4 \quad \text{only runs if} \quad v < 3 \quad \text{initially true}
\]

**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)

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

Repeated control needs longer programs, e.g., by copy&paste
Example (Speedy the point)

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

Nondeterministic repetition * repeats *any* arbitrary number of times
Outline

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 $\alpha$)

\[
\alpha, \beta ::= x := e \mid ?Q \mid x' = f(x) \& Q \mid \alpha \cup \beta \mid \alpha;\beta \mid \alpha^*
\]
### Definition (Syntax of hybrid program $\alpha$)

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

#### Syntax Elements
- **Discrete Assign**
- **Test Condition**
- **Differential Equation**
- **Nondet. Choice**
- **Seq. Compose**
- **Nondet. Repeat**
Hybrid Programs: Syntax

Definition (Syntax of hybrid program $\alpha$)

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

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

Like regular expressions. Everything nondeterministic.
Hybrid Programs: Semantics

\[ x := e \]

\[ x' = f(x) \& Q \]

?Q

\[ \omega \rightarrow v \]

\[ \omega \rightarrow v \]

\[ \omega \rightarrow v \]
Hybrid Programs: Semantics

\[ x := e \]

\[ x' = f(x) \& Q \]

\[ ?Q \]

\[ \begin{align*}
    & v \quad \text{if } v(x) = \omega[e] \\
    & \omega \quad \text{and } v(z) = \omega(z) \text{ for } z \neq x
\end{align*} \]
Hybrid Programs: Semantics

\( x := e \)

\( x' = f(x) \& Q \)

\(?Q\)
Hybrid Programs: Semantics

\[ x := e \]
\[ x' = f(x) \& Q \]
\[ ?Q \]

\[ v \]
\[ \omega \]

\[ v(x) = \omega[e] \]
\[ \omega \text{ and } v(z) = \omega(z) \text{ for } z \neq x \]

\[ x' = f(x) \& Q \]

\[ \omega \mid Q \]
\[ \omega \not\mid Q \]

Andre Platzer (CMU)
LFCPS/03: Choice & Control
Hybrid Programs: Semantics

\( x := e \)

\( x' = f(x) \& Q \)

\(?Q\)
Hybrid Programs: Semantics

- $x := e$
- $x' = f(x) \& Q$
- $?Q$

Semantics:
- $v$ if $v(x) = \omega[\epsilon]
- \omega$ and $v(z) = \omega(z)$ for $z \neq x$
- $x' = f(x) \& Q$
- $x' = f(x) \& Q$

Andre Platzer (CMU)
Hybrid Programs: Semantics

\[
x := e
\]

\[
x' = f(x) \& Q
\]

\[
?Q \quad \text{if } \omega \models Q
\]

\[
\begin{align*}
\nu &\quad \text{if } \nu(x) = \omega[e] \\
\omega &\quad \text{and } \nu(z) = \omega(z) \text{ for } z \neq x
\end{align*}
\]

\[
x' = f(x) \& Q
\]

\[
\omega \quad \text{no change if } \omega \models Q
\]
Hybrid Programs: Semantics

\( x := e \)

\( x' = f(x) & Q \)

\( ?Q \) if \( \omega \models Q \)

\( \omega \) no change if \( \omega \models Q \)

\( \omega \) otherwise no transition

\( \nu(\omega[e]) \)

\( \nu(z) = \omega(z) \) for \( z \neq x \)

\( t \)

\( 0 \)

\( r \)

\( 0 \)

\( t \)

\( x \)

\( x' = f(x) \& Q \)

\( Q \)

\( Q \)

\( \omega \)

André Platzer (CMU)

LFCPS/03: Choice & Control
Hybrid Programs: Semantics

\( \omega \cup \beta \)

\( \alpha ; \beta \)

\( \alpha^* \)
Hybrid Programs: Semantics

\[ \omega \xrightarrow{\alpha} \nu_1 \]
\[ \omega \xrightarrow{\alpha \cup \beta} \nu_1 \cup \nu_2 \]
\[ \nu_1 \xrightarrow{\alpha \cdot \beta} \nu_2 \]
\[ \omega \xrightarrow{\alpha^*} \nu \]
Hybrid Programs: Semantics

\[
\begin{align*}
    \omega &\xrightarrow{\alpha} v_1 \\
    \omega &\xrightarrow{\alpha \cup \beta} v_1 \\
    \omega &\xrightarrow{\beta} v_2 \\
    \alpha ; \beta &\xrightarrow{} \mu \xrightarrow{} v \\
    \omega &\xrightarrow{\alpha} \mu \\
    \omega &\xrightarrow{\beta} \mu \\
    \omega &\xrightarrow{\alpha^*} \nu \\
    \omega &\xrightarrow{} \nu
\end{align*}
\]
Hybrid Programs: Semantics
Hybrid Programs: Semantics

\[ \alpha \cup \beta \]

\[ \alpha ; \beta \]

\[ \alpha^* \]

\[ \omega \rightarrow \alpha \rightarrow \omega_1 \rightarrow \alpha \rightarrow \omega_2 \rightarrow \alpha \rightarrow \nu \]

\[ \omega \rightarrow \alpha \rightarrow \mu \rightarrow \beta \rightarrow \nu \]

\[ x \rightarrow \omega \rightarrow v_1 \rightarrow v_2 \rightarrow t \]

\[ x \rightarrow \omega \rightarrow v \rightarrow t \]

\[ x \rightarrow \omega \rightarrow v \rightarrow t \]
Plug-in for Semantics of Composed Hybrid Programs

\[ \alpha ; \beta \]

\[ \omega \xrightarrow{\alpha} \mu \xrightarrow{\beta} \nu \]

\[ \alpha^* \]

\[ \omega \xrightarrow{\alpha} \omega_1 \xrightarrow{\alpha} \omega_2 \xrightarrow{\alpha} \nu \]

\[ \omega \xrightarrow{\alpha} v_1 \]

\[ \omega \xrightarrow{\beta} v_2 \]

\[ \omega \xrightarrow{\alpha \cup \beta} v_1 \]

\[ \omega \xrightarrow{\beta} v_2 \]
Hybrid Programs: Syntax & Semantics

**Definition (Syntax of hybrid program $\alpha$)**

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

**Definition (Semantics of hybrid programs) \( \llbracket \cdot \rrbracket : \text{HP} \to \mathcal{P}(\mathcal{S} \times \mathcal{S}) \)**

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

**Compositional**
Definition (Syntax of hybrid program $\alpha$)

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

Definition (Semantics of hybrid programs)  
\[(\cdot) : HP \rightarrow \wp(S \times S)\]

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

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

Andre Platzer (CMU)
### Notational Conventions: Names

**Example (Naming Conventions)**

<table>
<thead>
<tr>
<th>Letters</th>
<th>Convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x, y, z$</td>
<td>variables</td>
</tr>
<tr>
<td>$e, \tilde{e}$</td>
<td>terms</td>
</tr>
<tr>
<td>$P, Q$</td>
<td>formulas</td>
</tr>
<tr>
<td>$\alpha, \beta$</td>
<td>programs</td>
</tr>
<tr>
<td>$c$</td>
<td>constant symbols</td>
</tr>
<tr>
<td>$f, g, h$</td>
<td>function symbols</td>
</tr>
<tr>
<td>$p, q, r$</td>
<td>predicate symbols</td>
</tr>
</tbody>
</table>

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

**Convention (Operator Precedence)**

1. Unary operators (including *, ¬ and ∀x, ∃x) bind stronger than binary.
2. ∧ binds stronger than ∨, which binds stronger than →, ↔
3. ; binds stronger than ∪
4. Arithmetic operators +, −, · associate to the left
5. Logical and program operators associate to the right

**Example (Operator Precedence)**

\[
\forall x P \land Q \equiv (\forall x P) \land Q \\
\alpha; \beta \cup \gamma \equiv (\alpha; \beta) \cup \gamma \\
\alpha \cup \beta; \gamma \equiv \alpha \cup (\beta; \gamma) \\
P \to Q \to R \equiv P \to (Q \to R).
\]

But →, ↔ expect explicit parentheses. Illegal: \( P \to Q \leftrightarrow R \) \( P \leftrightarrow Q \to R \)
1. Learning Objectives

2. Gradual Introduction to Hybrid Programs

3. Hybrid Programs
   - Syntax
   - Semantics
   - Notational Convention

4. Examples

5. Summary
Robot ≡ (ctrl ; drive)*

ctrl ≡ (?Q_A; a := A)
∪ (?Q_a; a := -b)

drive ≡ t := 0; {x' = v, v' = a, t' = 1 & v ≥ 0 ∧ t ≤ ε}
Robot ≡ (ctrl ; drive)*

ctrl ≡ (?Q_A; a := A)
    ∪ (?Q_b; a := −b)

drive ≡ t := 0; \{ x' = v, v' = a, t' = 1 & v \geq 0 \land t \leq \epsilon \}
Branching Transition Structure in Hybrid Programs

Robot ≡ (ctrl ; drive)∗

ctrl ≡ (?Q_A; a := A) ∪ (?Q_b; a := −b)

drive ≡ t := 0; \{ x' = v, v' = a, t' = 1 \& v ≥ 0 \& t ≤ ε \}
Robot \equiv (ctrl ; drive)^* \\
ctrl \equiv (?Q_A; a := A) \\
\quad \cup (?Q_b; a := -b) \\
drive \equiv t := 0; \{ x' = v, v' = a, t' = 1 \& v \geq 0 \land t \leq \varepsilon \}
Branching Transition Structure in Hybrid Programs

Robot \equiv (\text{ctrl} \; ; \text{drive})^* \\
\text{ctrl} \equiv (?Q_A; a := A) \\
\cup (\text{ctrl} \; ; a := A) \\
\text{drive} \equiv t := 0; \{ x' = v, v' = a, t' = 1 \& v \geq 0 \land t \leq \epsilon \}
Branching Transition Structure in Hybrid Programs

Robot \equiv (\text{ctrl} ; \text{drive})^* \\
\text{ctrl} \equiv (\text{?}Q_A; a := A) \\
\quad \cup (\text{?}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

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

\text{ctrl} \equiv (\nabla Q_A; a := A)
\quad \cup (\nabla Q_b; a := -b)

\text{drive} \equiv t := 0; \{x' = v, v' = a, t' = 1 \land v \geq 0 \land t \leq \varepsilon\}
Branching Transition Structure in Hybrid Programs

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

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 \}
Robot \equiv (ctrl; drive)^*

ctrl \equiv (?Q_A; a := A)
\quad \cup (?Q_b; a := -b)

drive \equiv t := 0; \{ x' = v, v' = a, t' = 1 \& v \geq 0 \& t \leq \varepsilon \}
Robot \equiv (ctrl \cdot drive)^*

ctrl \equiv (?Q_A; a := A)
\quad \cup (?Q_b; a := -b)

drive \equiv t := 0; \{ x' = v, v' = a, t' = 1 \& v \geq 0 \land t \leq \varepsilon \}
Robot ≡ (ctrl ; drive)*

ctrl ≡ (?QA; a := A)

∪ (?QB; a := −b)
drive ≡ t := 0; \{x' = v, v' = a, t' = 1 & v ≥ 0 \land t ≤ ε\}
Branching Transition Structure in Hybrid Programs

Robot \equiv (\text{ctrl} ; \text{drive})^* \\
\text{ctrl} \equiv (?Q_A; a \equiv A) \\
\quad \cup (?Q_b; a \equiv -b) \\
\text{drive} \equiv t \equiv 0; \{ x' = v, v' = a, t' = 1 \& v \geq 0 \& t \leq \varepsilon \}
Branching Transition Structure in Hybrid Programs

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

\[
\text{ctrl} \equiv (? Q_A; a := A) \\
\cup (? Q_b; a := -b)
\]

\[
\text{drive} \equiv t := 0; \{ x' = v, v' = a, t' = 1 \& v \geq 0 \& t \leq \epsilon \} 
\]
Robot $\equiv (ctrl \; ; \; drive)^*$

\begin{align*}
ctrl & \equiv (?Q_A; \; a := A) \\
& \quad \cup (?Q_b; \; a := -b)
\end{align*}

\begin{align*}
drive & \equiv t := 0; \{ x' = v, v' = a, t' = 1 \; \& \; v \geq 0 \; \& \; t \leq \epsilon \} 
\end{align*}
Branching Transition Structure in Hybrid Programs

if \((Q)\alpha\) else \(\beta\) ≡

while \((Q)\alpha\) ≡

Robot ≡ (ctrl ; drive)∗

ctrl ≡ (?\(Q_A\); \(a := A\))

∪ (?\(Q_b\); \(a := -b\))

drive ≡ \(t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0 \land t \leq \varepsilon\}\)
Branching Transition Structure in Hybrid Programs

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\}

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

while(Q) \alpha \equiv

\begin{align*}
Q_A & \rightarrow a := A \\
Q_b & \rightarrow a := -b \\
t & := 0 \\
x'' & := a \\
t' & := 1 \\
& \land v \geq 0 \land t \leq \varepsilon
\end{align*}
Branching Transition Structure in Hybrid Programs

Robot ≡ (ctrl ; drive)*

\[
\text{ctrl} \equiv (?Q_A; \ a := A) \\
\quad \cup \ (?Q_b; \ a := -b)
\]

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

if(\(Q\)) \(\alpha\) else \(\beta\) ≡ (?\(Q\); \(\alpha\)) \(\cup\) (?\(\neg Q\); \(\beta\))

while(\(Q\)) \(\alpha\) ≡ (?\(Q\); \(\alpha\))^*; ?\(\neg Q\)
Runaround Robot with Dubins Paths

\[
Q \equiv \frac{1}{2} \left( (x + w \omega - o x)^2 + (y - v \omega - o y)^2 \right) \neq v^2 + w^2 \\
Q_0 \equiv (o x - x) w \neq (o y - y)v
\]

Obstacle not on tangential circle
Obstacle not on ray

Example (Runaround Robot)

\[
\omega \begin{aligned}
&:= -1 \cup \omega := 1 \cup \omega := 0 \\
\{ \begin{array}{l}
x' = v, \\
y' = w, \\
v' = \omega w, \\
w' = -\omega v
\end{array} \end{aligned}
\]

\[
\omega \begin{aligned}
&:= -1 \cup \omega := 1 \cup \omega := 0 \\
\{ \begin{array}{l}
x' = v, \\
y' = w, \\
v' = \omega w, \\
w' = -\omega v
\end{array} \end{aligned}
\]
Example (Runaround Robot)

\[
\left( (\omega := -1 \cup \omega := 1 \cup \omega := 0); \right. \\
\left. \{ x' = v, y' = w, v' = \omega w, w' = -\omega v \} \right)^* 
\]
Example (Runaround Robot)

\[((\unit{?Q}{-1}; \omega := -1 \cup \unit{?Q}{1}; \omega := 1 \cup \unit{?Q}{0}; \omega := 0); \\
\{x' = v, y' = w, v' = \omega w, w' = -\omega v\})^\ast\]
Example (Speedy the point)

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

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

$(\forall v < 4; a := a + 1 \cup a := -b)$;
\[\{x' = v, v' = a\} ;\]
Example (Speedy the point)

\[
\begin{align*}
?v &< 4; a := a + 1; \\
\{x' = v, v' = a\}; \\
?v &< 4; a := a + 1; \\
\{x' = v, v' = a\}; \\
?v &< 4; a := a + 1; \\
\{x' = v, v' = a\}
\end{align*}
\]
Example (Speedy the point)

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

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!
Hybrid Programs: Syntax & Semantics

Definition (Syntax of hybrid program \( \alpha \))

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

Definition (Semantics of hybrid programs) \( ([\cdot] : HP \rightarrow \wp(\mathcal{S} \times \mathcal{S})) \)

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

compositional
André Platzer.

*Logical Foundations of Cyber-Physical Systems.*
URL: http://www.springer.com/978-3-319-63587-3, doi:10.1007/978-3-319-63588-0.

André Platzer.

*Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics.*
doi:10.1007/978-3-642-14509-4.

André Platzer.

Logics of dynamical systems.

André Platzer.

Differential dynamic logic for hybrid systems.
André Platzer.
A complete uniform substitution calculus for differential dynamic logic.