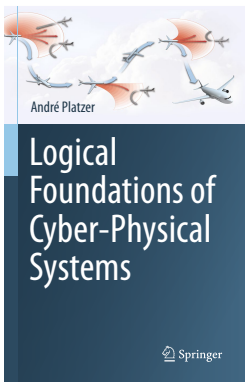


04: Safety & Contracts

Logical Foundations of Cyber-Physical Systems



André Platzer

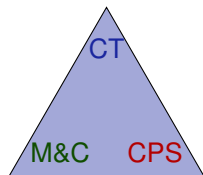


- 1 Learning Objectives
- 2 Quantum the Acrophobic Bouncing Ball
- 3 Contracts for CPS
 - Safety of Robots
 - Safety of Bouncing Balls
- 4 Logical Formulas for Hybrid Programs
- 5 Differential Dynamic Logic
 - Syntax
 - Semantics
 - Notational Convention
- 6 Identifying Requirements of a CPS
- 7 Summary



- 1 Learning Objectives
- 2 Quantum the Acrophobic Bouncing Ball
- 3 Contracts for CPS
 - Safety of Robots
 - Safety of Bouncing Balls
- 4 Logical Formulas for Hybrid Programs
- 5 Differential Dynamic Logic
 - Syntax
 - Semantics
 - Notational Convention
- 6 Identifying Requirements of a CPS
- 7 Summary

rigorous specification
contracts
preconditions
postconditions
differential dynamic logic

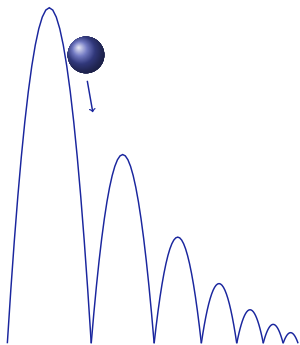


discrete+continuous
analytic specification

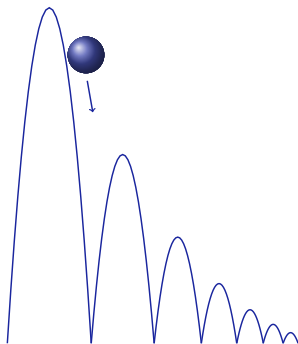
model semantics
reasoning principles



- 1 Learning Objectives
- 2 Quantum the Acrophobic Bouncing Ball
- 3 Contracts for CPS
 - Safety of Robots
 - Safety of Bouncing Balls
- 4 Logical Formulas for Hybrid Programs
- 5 Differential Dynamic Logic
 - Syntax
 - Semantics
 - Notational Convention
- 6 Identifying Requirements of a CPS
- 7 Summary

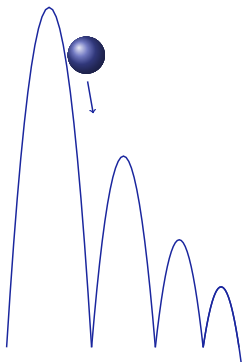


Example (Quantum the Bouncing Ball)



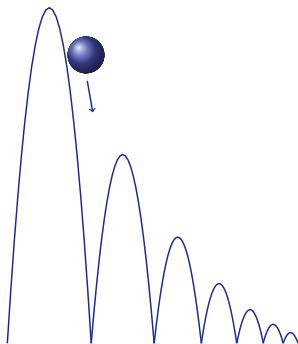
Example (Quantum the Bouncing Ball)

$$\{x' = v, v' = -g\}$$



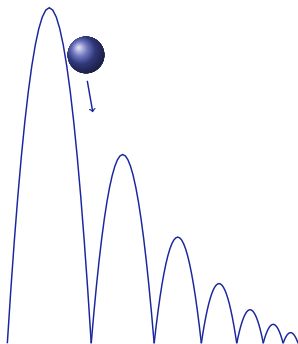
Example (Quantum the Bouncing Ball)

$$\{x' = v, v' = -g\}$$



Example (Quantum the Bouncing Ball)

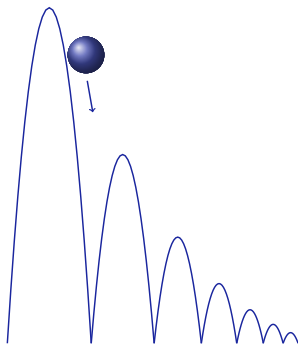
$$\{x' = v, v' = -g \& x \geq 0\}$$



Example (Quantum the Bouncing Ball)

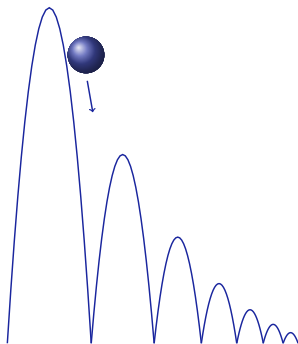
$$\{x' = v, v' = -g \& x \geq 0\};$$

$$\text{if}(x = 0) \ v := -cv$$



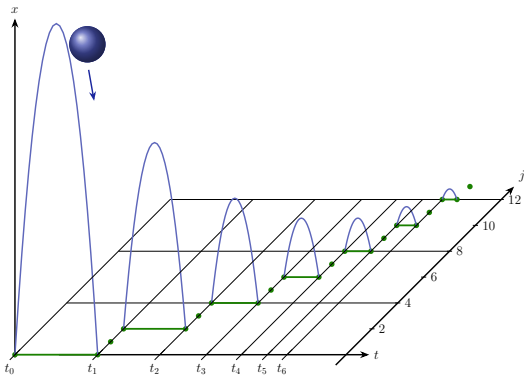
Example (Quantum the Bouncing Ball)

$$\begin{aligned} &(\{x' = v, v' = -g \& x \geq 0\}; \\ &\text{if}(x = 0) \ v := -cv)^* \end{aligned}$$



Example (Quantum the Bouncing Ball)

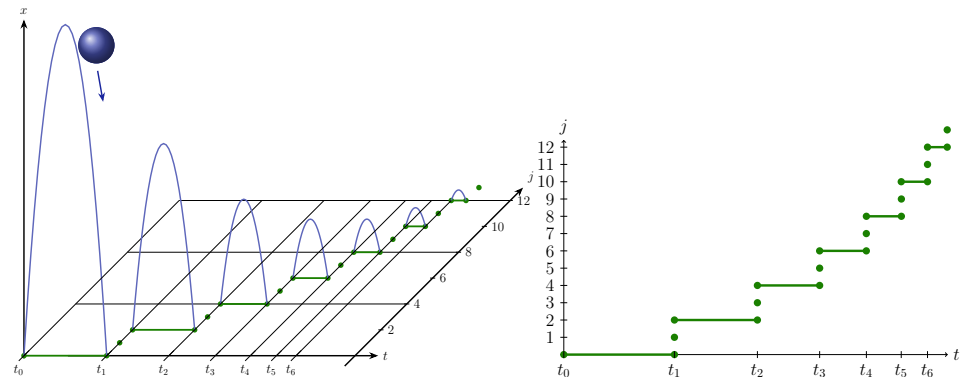
$$\begin{aligned} &(\{x' = v, v' = -g \& x \geq 0\}; \\ &\text{if}(x = 0) \ v := -cv)^* \end{aligned}$$



Example (Quantum the Bouncing Ball)

$$(\{x' = v, v' = -g \& x \geq 0\};$$

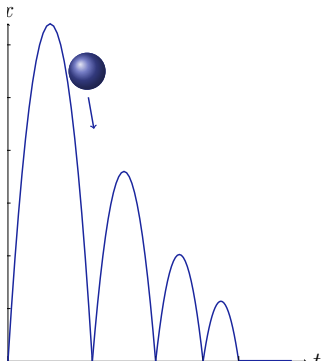
$$\text{if}(x = 0) \ v := -cv)^*$$



Example (Quantum the Bouncing Ball)

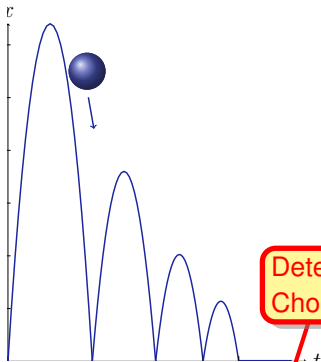
$$(\{x' = v, v' = -g \& x \geq 0\};$$

$$\text{if}(x = 0) \ v := -cv)^*$$



Example (Quantum the Bouncing Ball)

$$\begin{aligned} &(\{x' = v, v' = -g \& x \geq 0\}; \\ &\text{if}(x = 0) \ v := -cv)^* \end{aligned}$$



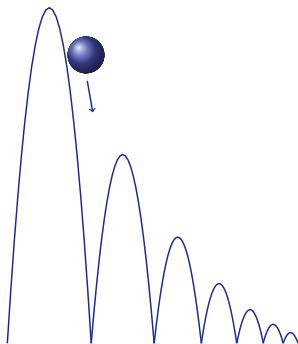
Determ.
Choice

Nondet.
Choice

Example (Quantum the Bouncing Ball)

$$(\{x' = v, v' = -g \ \& \ x \geq 0\};$$

$$\text{if}(x = 0) (v := -cv \cup v := 0))^*$$



Example (Quantum the Bouncing Ball)

$$\begin{aligned} &(\{x' = v, v' = -g \& x \geq 0\}; \\ &\text{if}(x = 0) \ v := -cv)^* \end{aligned}$$



- 1 Learning Objectives
- 2 Quantum the Acrophobic Bouncing Ball
- 3 Contracts for CPS**
 - Safety of Robots
 - Safety of Bouncing Balls
- 4 Logical Formulas for Hybrid Programs
- 5 Differential Dynamic Logic
 - Syntax
 - Semantics
 - Notational Convention
- 6 Identifying Requirements of a CPS
- 7 Summary





Three Laws of Robotics

Isaac Asimov 1942

- 1 A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- 2 A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
- 3 A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

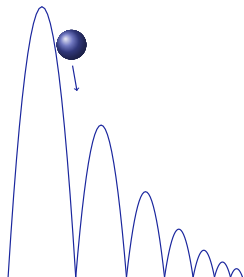


Three Laws of Robotics

Isaac Asimov 1942

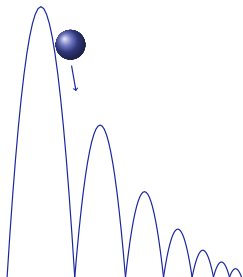
- 1 A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- 2 A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
- 3 A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Three Laws of Robotics are not the answer.
They are the inspiration!



Example (Quantum the Bouncing Ball)

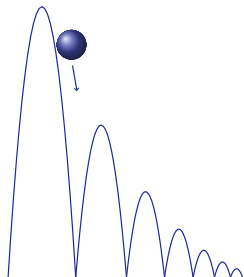
$$\begin{aligned} &(\{x' = v, v' = -g \& x \geq 0\}; \\ &\text{if}(x = 0) v := -cv)^* \end{aligned}$$



Example (Quantum the Bouncing Ball)

ensures($0 \leq x$)

$\{\{x' = v, v' = -g \& x \geq 0\};$
 $\text{if}(x = 0) v := -cv)^*$



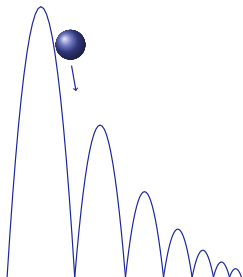
Example (Quantum the Bouncing Ball)

ensures $(0 \leq x)$

ensures $(x \leq H)$

$\{\{x' = v, v' = -g \& x \geq 0\};$

$\text{if}(x = 0) v := -cv)^*$



Example (Quantum the Bouncing Ball)

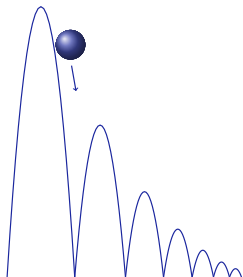
requires($x = H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{\{x' = v, v' = -g \& x \geq 0\};$

$\text{if}(x = 0) v := -cv)^*$



Example (Quantum the Bouncing Ball)

requires($x = H$)

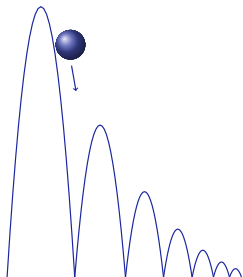
requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{\{x' = v, v' = -g \& x \geq 0\};$

$\text{if}(x = 0) v := -cv)^*$



Example (Quantum the Bouncing Ball)

requires($x = H$)

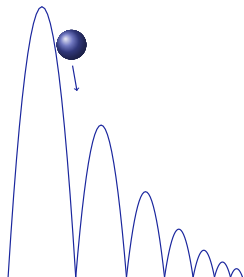
requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{\{x' = v, v' = -g \& x \geq 0\};$

$\text{if}(x = 0) v := -cv)^* \text{@invariant}(x \geq 0)$



Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{\{x' = v, v' = -g \& x \geq 0\};$

$\text{if}(x = 0) v := -cv)^* @\text{invariant}(x \geq 0)$



- 1 Learning Objectives
- 2 Quantum the Acrophobic Bouncing Ball
- 3 Contracts for CPS
 - Safety of Robots
 - Safety of Bouncing Balls
- 4 Logical Formulas for Hybrid Programs**
- 5 Differential Dynamic Logic
 - Syntax
 - Semantics
 - Notational Convention
- 6 Identifying Requirements of a CPS
- 7 Summary

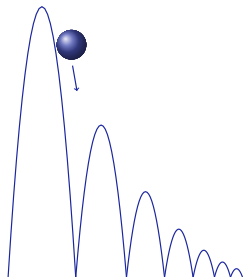
CPS contracts are crucial for CPS safety.

We need to understand CPS programs and contracts and how we can convince ourselves that a CPS program respects its contract.

Contracts are at a disadvantage compared to full logic.

Logic is for Specification and Reasoning

- 1 Specification of a whole CPS program.
- 2 Analytic inspection of its parts.
- 3 Argumentative relations between contracts and program parts.
“Yes, this CPS program meets its contract, and here’s why . . .”



Example (Quantum the Bouncing Ball)

requires($x = H$)

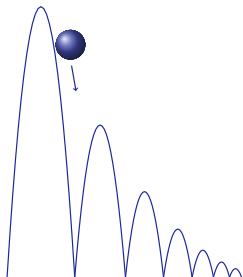
requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{\{x' = v, v' = -g \& x \geq 0\};$

$\text{if}(x = 0) v := -cv)^*$



Example (Quantum the Bouncing Ball)

requires($x = H$)

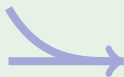
requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

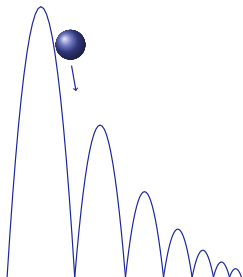
$\{x' = v, v' = -g \ \& \ x \geq 0\};$

$\text{if}(x = 0) \ v := -cv)^*$



Precondition:

$x = H \wedge 0 \leq H$ in FOL



Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{x' = v, v' = -g \ \& \ x \geq 0\};$

$\text{if}(x = 0) \ v := -cv)^*$



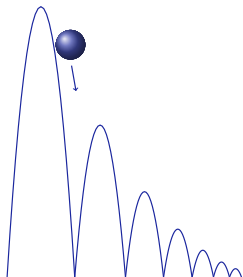
Precondition:

$x = H \wedge 0 \leq H$ in FOL



Postcondition:

$0 \leq x \wedge x \leq H$ in FOL



Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{\{x' = v, v' = -g \ \& \ x \geq 0\};$

$\text{if}(x = 0) \ v := -cv)^*$



Precondition:

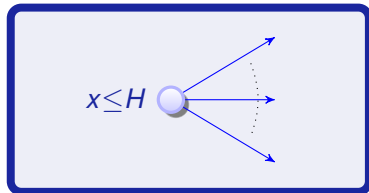
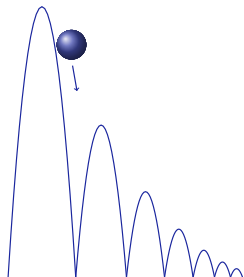
$x = H \wedge 0 \leq H$ in FOL



Postcondition:

$0 \leq x \wedge x \leq H$ in FOL

How to say post is true
after all HP runs?



Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{x' = v, v' = -g \& x \geq 0\};$

$\text{if}(x = 0) v := -cv)^*$

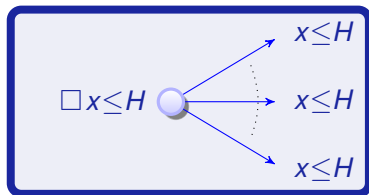
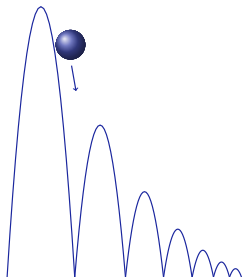


Precondition:

$x = H \wedge 0 \leq H$ in FOL

Postcondition:

$0 \leq x \wedge x \leq H$ in FOL



Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{\{x' = v, v' = -g \& x \geq 0\};$

$\text{if}(x = 0) v := -cv)^*$



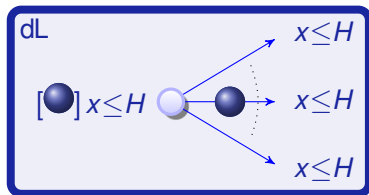
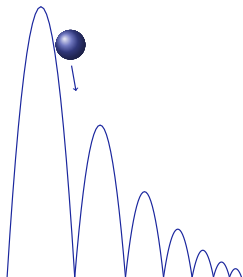
Precondition:

$x = H \wedge 0 \leq H$ in FOL



Postcondition:

$0 \leq x \wedge x \leq H$ in FOL



Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{\{x' = v, v' = -g \& x \geq 0\};$

$\text{if}(x = 0) v := -cv)^*$

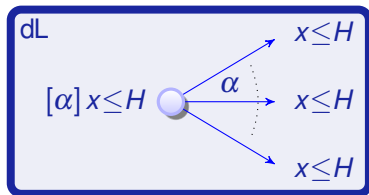
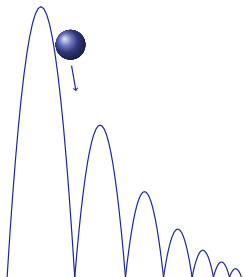


Precondition:

$x = H \wedge 0 \leq H$ in FOL

Postcondition:

$0 \leq x \wedge x \leq H$ in FOL



Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{x' = v, v' = -g \ \& \ x \geq 0\};$

$\text{if}(x = 0) \ v := -cv)^*$



Precondition:

$x = H \wedge 0 \leq H$ in FOL

Postcondition:

$0 \leq x \wedge x \leq H$ in FOL

$$[(\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*]$$

Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$(\{x' = v, v' = -g \& x \geq 0\};$
 $\text{if}(x = 0) v := -cv)^*$



Precondition:

$x = H \wedge 0 \leq H$ in FOL



Postcondition:

$0 \leq x \wedge x \leq H$ in FOL

$$[(\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](x \leq H)$$

Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$(\{x' = v, v' = -g \& x \geq 0\};$
 $\text{if}(x = 0) v := -cv)^*$



Precondition:

$x = H \wedge 0 \leq H$ in FOL

Postcondition:

$0 \leq x \wedge x \leq H$ in FOL

$$[[\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](0 \leq x)$$

$$[[\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](x \leq H)$$

Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$\{x' = v, v' = -g \& x \geq 0\};$
 $\text{if}(x = 0) v := -cv)^*$



Precondition:

$x = H \wedge 0 \leq H$ in FOL



Postcondition:

$0 \leq x \wedge x \leq H$ in FOL

$$[(\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](0 \leq x)$$

$$[(\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](x \leq H)$$

$$[(\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](0 \leq x \wedge x \leq H)$$

Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$(\{x' = v, v' = -g \& x \geq 0\};$
 $\text{if}(x = 0) v := -cv)^*$



Precondition:

$x = H \wedge 0 \leq H$ in FOL



Postcondition:

$0 \leq x \wedge x \leq H$ in FOL

$$\begin{aligned}
 & [(\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](0 \leq x) \\
 \wedge & [(\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](x \leq H) \\
 \leftrightarrow & [(\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](0 \leq x \wedge x \leq H)
 \end{aligned}$$

Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$(\{x' = v, v' = -g \& x \geq 0\};$
 $\text{if}(x = 0) v := -cv)^*$



Precondition:

$x = H \wedge 0 \leq H$ in FOL



Postcondition:

$0 \leq x \wedge x \leq H$ in FOL

$$[(\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](0 \leq x)$$

Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$(\{x' = v, v' = -g \& x \geq 0\};$
 $\text{if}(x = 0) v := -cv)^*$



Precondition:

$x = H \wedge 0 \leq H$ in FOL



Postcondition:

$0 \leq x \wedge x \leq H$ in FOL

$$x=H \rightarrow [(\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](0 \leq x)$$

Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$(\{x' = v, v' = -g \& x \geq 0\};$
 $\text{if}(x = 0) v := -cv)^*$



Precondition:

$x = H \wedge 0 \leq H$ in FOL



Postcondition:

$0 \leq x \wedge x \leq H$ in FOL



$$0 \leq x \wedge x = H \rightarrow [(\{x' = v, v' = -g \& x \geq 0\}; \text{if}(x=0) v := -cv)^*](0 \leq x)$$

Example (Quantum the Bouncing Ball)

requires($x = H$)

requires($0 \leq H$)

ensures($0 \leq x$)

ensures($x \leq H$)

$(\{x' = v, v' = -g \& x \geq 0\};$
 $\text{if}(x = 0) v := -cv)^*$



Precondition:

$x = H \wedge 0 \leq H$ in FOL



Postcondition:

$0 \leq x \wedge x \leq H$ in FOL



- 1 Learning Objectives
- 2 Quantum the Acrophobic Bouncing Ball
- 3 Contracts for CPS
 - Safety of Robots
 - Safety of Bouncing Balls
- 4 Logical Formulas for Hybrid Programs
- 5 Differential Dynamic Logic**
 - **Syntax**
 - **Semantics**
 - **Notational Convention**
- 6 Identifying Requirements of a CPS
- 7 Summary

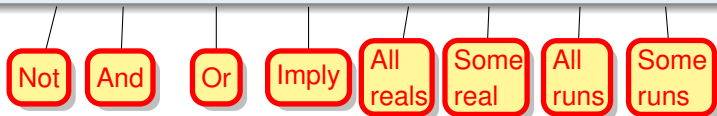
Definition (Syntax of differential dynamic logic)

The *formulas of differential dynamic logic* are defined by the grammar:

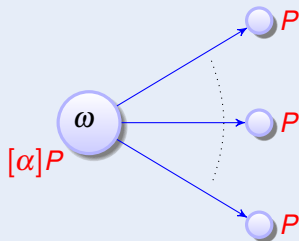
$$P, Q ::= e \geq \tilde{e} \mid \neg P \mid P \wedge Q \mid P \vee Q \mid P \rightarrow Q \mid \forall x P \mid \exists x P \mid [\alpha]P \mid \langle \alpha \rangle P$$

Definition (Syntax of differential dynamic logic)

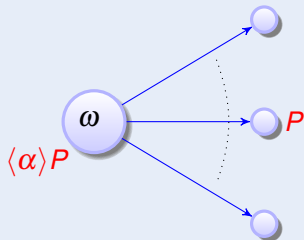
The *formulas of differential dynamic logic* are defined by the grammar:

$$P, Q ::= e \geq \tilde{e} \mid \neg P \mid P \wedge Q \mid P \vee Q \mid P \rightarrow Q \mid \forall x P \mid \exists x P \mid [\alpha]P \mid \langle \alpha \rangle P$$


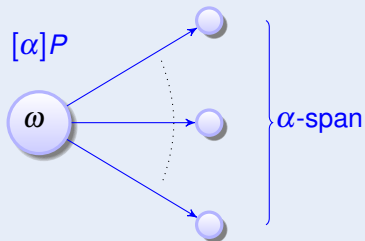
Definition (dL Formulas)



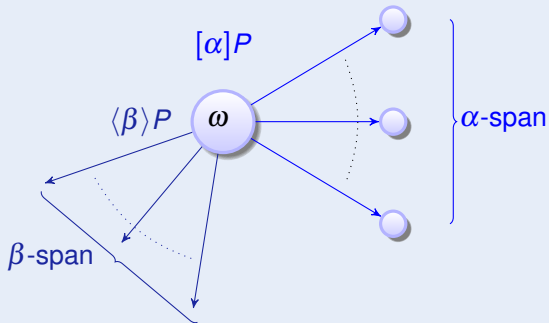
Definition (dL Formulas)



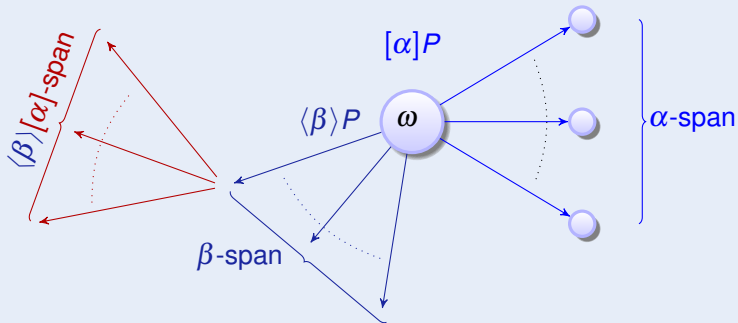
Definition (dL Formulas)



Definition (dL Formulas)



Definition (dL Formulas)



Definition (Syntax of differential dynamic logic)

The *formulas of differential dynamic logic* are defined by the grammar:

$$P, Q ::= e \geq \tilde{e} \mid \neg P \mid P \wedge Q \mid P \vee Q \mid P \rightarrow Q \mid \forall x P \mid \exists x P \mid [\alpha]P \mid \langle \alpha \rangle P$$

Definition (dL semantics)

$$(\llbracket \cdot \rrbracket : \text{Fml} \rightarrow \wp(\mathcal{S}))$$

$$\llbracket e \geq \tilde{e} \rrbracket = \{\omega : \omega[e] \geq \omega[\tilde{e}]\}$$

$$\llbracket \neg P \rrbracket = \llbracket P \rrbracket^c = \mathcal{S} \setminus \llbracket P \rrbracket$$

$$\llbracket P \wedge Q \rrbracket = \llbracket P \rrbracket \cap \llbracket Q \rrbracket$$

$$\llbracket P \vee Q \rrbracket = \llbracket P \rrbracket \cup \llbracket Q \rrbracket$$

$$\llbracket P \rightarrow Q \rrbracket = \llbracket P \rrbracket^c \cup \llbracket Q \rrbracket$$

$$\llbracket \langle \alpha \rangle P \rrbracket = \llbracket \alpha \rrbracket \circ \llbracket P \rrbracket = \{\omega : v \in \llbracket P \rrbracket \text{ for some } v : (\omega, v) \in \llbracket \alpha \rrbracket\}$$

$$\llbracket [\alpha]P \rrbracket = \llbracket \neg \langle \alpha \rangle \neg P \rrbracket = \{\omega : v \in \llbracket P \rrbracket \text{ for all } v : (\omega, v) \in \llbracket \alpha \rrbracket\}$$

$$\llbracket \exists x P \rrbracket = \{\omega : \omega_x^r \in \llbracket P \rrbracket \text{ for some } r \in \mathbb{R}\}$$

$$\llbracket \forall x P \rrbracket = \{\omega : \omega_x^r \in \llbracket P \rrbracket \text{ for all } r \in \mathbb{R}\}$$

$$\omega_x^d(y) = \begin{cases} d & \text{if } y=x \\ \omega(y) & \text{if } y \neq x \end{cases}$$

$\llbracket P \rrbracket$ the set of states in which formula P is true

$\omega \models P$ formula P is true in state ω , alias $\omega \in \llbracket P \rrbracket$

$\models P$ formula P is valid, i.e., true in all states ω , i.e., $\llbracket P \rrbracket = \mathcal{S}$

Definition (dL semantics)

$(\llbracket \cdot \rrbracket : \text{Fml} \rightarrow \wp(\mathcal{S}))$

$$\llbracket e \geq \tilde{e} \rrbracket = \{ \omega : \omega[e] \geq \omega[\tilde{e}] \}$$

$$\llbracket \neg P \rrbracket = \llbracket P \rrbracket^c = \mathcal{S} \setminus \llbracket P \rrbracket$$

$$\llbracket P \wedge Q \rrbracket = \llbracket P \rrbracket \cap \llbracket Q \rrbracket$$

$$\llbracket P \vee Q \rrbracket = \llbracket P \rrbracket \cup \llbracket Q \rrbracket$$

$$\llbracket P \rightarrow Q \rrbracket = \llbracket P \rrbracket^c \cup \llbracket Q \rrbracket$$

$$\llbracket \langle \alpha \rangle P \rrbracket = \llbracket \alpha \rrbracket \circ \llbracket P \rrbracket = \{ \omega : v \in \llbracket P \rrbracket \text{ for some } v : (\omega, v) \in \llbracket \alpha \rrbracket \}$$

$$\llbracket [\alpha] P \rrbracket = \llbracket \neg \langle \alpha \rangle \neg P \rrbracket = \{ \omega : v \in \llbracket P \rrbracket \text{ for all } v : (\omega, v) \in \llbracket \alpha \rrbracket \}$$

$$\llbracket \exists x P \rrbracket = \{ \omega : \omega_x^r \in \llbracket P \rrbracket \text{ for some } r \in \mathbb{R} \}$$

$$\llbracket \forall x P \rrbracket = \{ \omega : \omega_x^r \in \llbracket P \rrbracket \text{ for all } r \in \mathbb{R} \}$$

$\llbracket P \rrbracket$ the set of states in which formula P is true

$\omega \models P$ formula P is true in state ω , alias $\omega \in \llbracket P \rrbracket$

$\models P$ formula P is valid, i.e., true in all states ω , i.e., $\llbracket P \rrbracket = \mathcal{S}$

$$\exists d [x := 1; x' = d] x \geq 0 \quad \text{and} \quad [x := x + 1; x' = d] x \geq 0 \quad \text{and} \quad \langle x' = d \rangle x \geq 0$$

Definition (dL semantics)

($\llbracket \cdot \rrbracket : \text{Fml} \rightarrow \wp(\mathcal{S})$)

$$\llbracket e \geq \tilde{e} \rrbracket = \{ \omega : \omega[e] \geq \omega[\tilde{e}] \}$$

$$\llbracket \neg P \rrbracket = \llbracket P \rrbracket^c = \mathcal{S} \setminus \llbracket P \rrbracket$$

$$\llbracket P \wedge Q \rrbracket = \llbracket P \rrbracket \cap \llbracket Q \rrbracket$$

$$\llbracket P \vee Q \rrbracket = \llbracket P \rrbracket \cup \llbracket Q \rrbracket$$

$$\llbracket P \rightarrow Q \rrbracket = \llbracket P \rrbracket^c \cup \llbracket Q \rrbracket$$

$$\llbracket \langle \alpha \rangle P \rrbracket = \llbracket \alpha \rrbracket \circ \llbracket P \rrbracket = \{ \omega : v \in \llbracket P \rrbracket \text{ for some } v : (\omega, v) \in \llbracket \alpha \rrbracket \}$$

$$\llbracket [\alpha] P \rrbracket = \llbracket \neg \langle \alpha \rangle \neg P \rrbracket = \{ \omega : v \in \llbracket P \rrbracket \text{ for all } v : (\omega, v) \in \llbracket \alpha \rrbracket \}$$

$$\llbracket \exists x P \rrbracket = \{ \omega : \omega_x^r \in \llbracket P \rrbracket \text{ for some } r \in \mathbb{R} \}$$

$$\llbracket \forall x P \rrbracket = \{ \omega : \omega_x^r \in \llbracket P \rrbracket \text{ for all } r \in \mathbb{R} \}$$

$\llbracket P \rrbracket$ the set of states in which formula P is true

$\omega \models P$ formula P is true in state ω , alias $\omega \in \llbracket P \rrbracket$

$\models P$ formula P is valid, i.e., true in all states ω , i.e., $\llbracket P \rrbracket = \mathcal{S}$

$\vDash \exists d [x := 1; x' = d] x \geq 0$ and $\not\vDash [x := x + 1; x' = d] x \geq 0$ and $\not\vDash \langle x' = d \rangle x \geq 0$

Definition (dL semantics)

$(\llbracket \cdot \rrbracket : \text{Fml} \rightarrow \wp(\mathcal{S}))$

$$\llbracket e \geq \tilde{e} \rrbracket = \{ \omega : \omega[e] \geq \omega[\tilde{e}] \}$$

$$\llbracket \neg P \rrbracket = \llbracket P \rrbracket^c = \mathcal{S} \setminus \llbracket P \rrbracket$$

$$\llbracket P \wedge Q \rrbracket = \llbracket P \rrbracket \cap \llbracket Q \rrbracket$$

$$\llbracket P \vee Q \rrbracket = \llbracket P \rrbracket \cup \llbracket Q \rrbracket$$

$$\llbracket P \rightarrow Q \rrbracket = \llbracket P \rrbracket^c \cup \llbracket Q \rrbracket$$

$$\llbracket \langle \alpha \rangle P \rrbracket = \llbracket \alpha \rrbracket \circ \llbracket P \rrbracket = \{ \omega : v \in \llbracket P \rrbracket \text{ for some } v : (\omega, v) \in \llbracket \alpha \rrbracket \}$$

$$\llbracket [\alpha] P \rrbracket = \llbracket \neg \langle \alpha \rangle \neg P \rrbracket = \{ \omega : v \in \llbracket P \rrbracket \text{ for all } v : (\omega, v) \in \llbracket \alpha \rrbracket \}$$

$$\llbracket \exists x P \rrbracket = \{ \omega : \omega_x^r \in \llbracket P \rrbracket \text{ for some } r \in \mathbb{R} \}$$

$$\llbracket \forall x P \rrbracket = \{ \omega : \omega_x^r \in \llbracket P \rrbracket \text{ for all } r \in \mathbb{R} \}$$

Convention (Operator Precedence)

- 1 Unary operators (e.g., $*$, \neg , $\forall x, \exists x, [\alpha], \langle \alpha \rangle$) 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

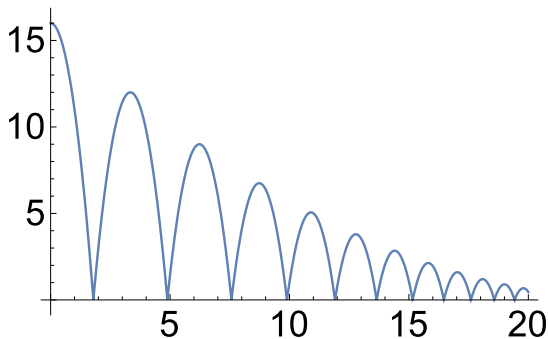
Example (Operator Precedence)

$$\begin{aligned}
 [\alpha]P \wedge Q &\equiv ([\alpha]P) \wedge Q & \forall x P \wedge Q &\equiv (\forall x P) \wedge Q & \forall x P \rightarrow Q &\equiv (\forall x P) \rightarrow Q \\
 \alpha; \beta \cup \gamma &\equiv (\alpha; \beta) \cup \gamma & \alpha \cup \beta; \gamma &\equiv \alpha \cup (\beta; \gamma) & \alpha; \beta^* &\equiv \alpha; (\beta^*) \\
 P \rightarrow Q \rightarrow R &\equiv P \rightarrow (Q \rightarrow R).
 \end{aligned}$$

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



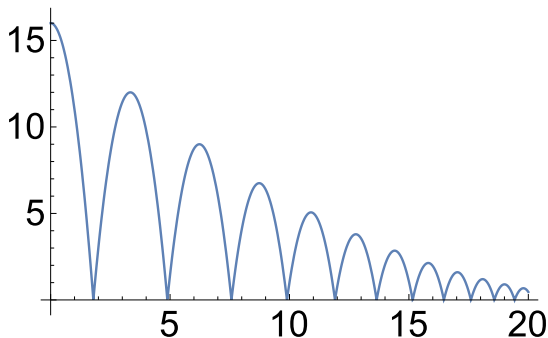
- 1 Learning Objectives
- 2 Quantum the Acrophobic Bouncing Ball
- 3 Contracts for CPS
 - Safety of Robots
 - Safety of Bouncing Balls
- 4 Logical Formulas for Hybrid Programs
- 5 Differential Dynamic Logic
 - Syntax
 - Semantics
 - Notational Convention
- 6 Identifying Requirements of a CPS**
- 7 Summary



Example (▶ Bouncing Ball)

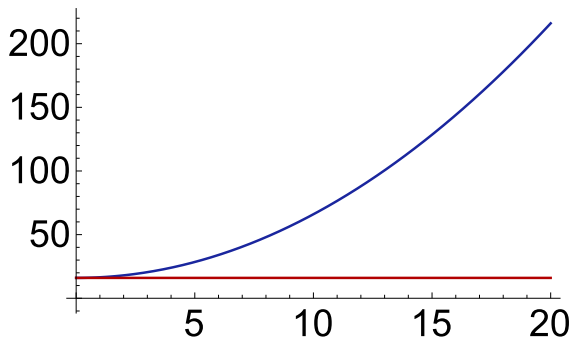
$$(\{x' = v, v' = -g \& x \geq 0\};$$

$$\text{if}(x = 0) v := -cv)^*$$



Example (▶ Bouncing Ball)

$$H = x \geq 0 \quad \rightarrow \quad [(\{x' = v, v' = -g \& x \geq 0\}; \\ \text{if}(x = 0) v := -cv)^*] \quad 0 \leq x \leq H$$



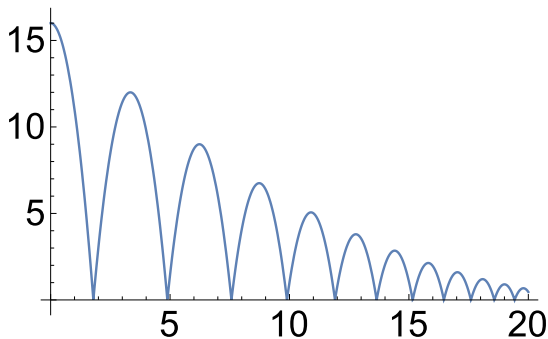
Not if $g < 0$ in anti-gravity

Example (▶ Bouncing Ball)

$$H = x \geq 0$$

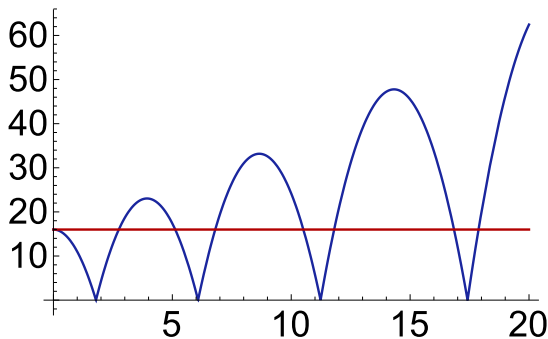
~~$$\rightarrow [(\{x' = v, v' = -g \& x \geq 0\};$$

$$\text{if}(x = 0) v := -cv)^*] 0 \leq x \leq H$$~~



Example (▶ Bouncing Ball)

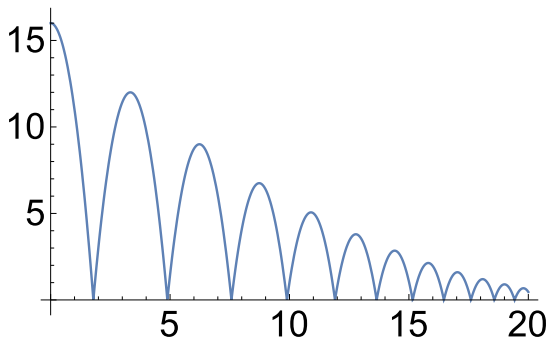
$$H = x \geq 0 \wedge g > 0 \rightarrow [(\{x' = v, v' = -g \wedge x \geq 0\}; \\ \text{if}(x = 0) v := -cv)^*] 0 \leq x \leq H$$



Not if $c > 1$ for anti-damping

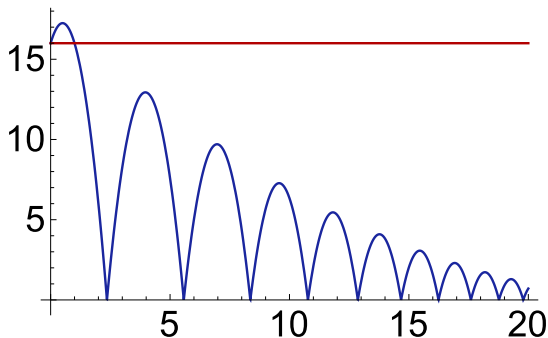
Example (▶ Bouncing Ball)

$$H = x \geq 0 \wedge g > 0 \rightarrow \left[\left(\{x' = v, v' = -g \ \& \ x \geq 0\}; \right. \right. \\ \left. \left. \text{if } (x = 0) \ v := -cv \right)^* \right] \ 0 \leq x \leq H$$



Example (▶ Bouncing Ball)

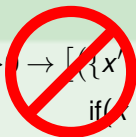
$$1 \geq c \geq 0 \wedge H = x \geq 0 \wedge g > 0 \rightarrow [(\{x' = v, v' = -g \wedge x \geq 0\}; \\ \text{if}(x = 0) v := -cv)^*] 0 \leq x \leq H$$

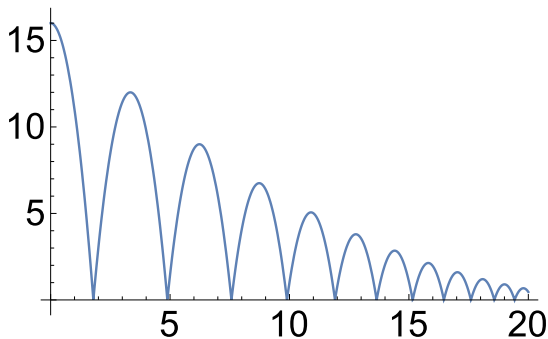


Not if $v > 0$ initial climbing

Example (▶ Bouncing Ball)

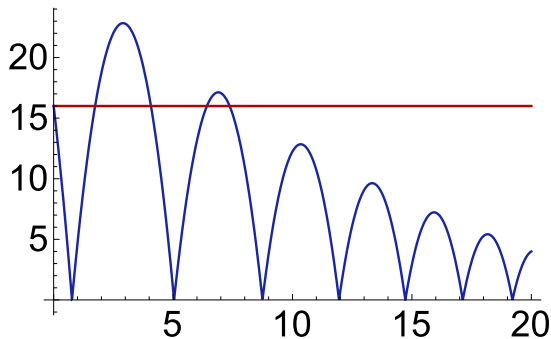
$$1 \geq c \geq 0 \wedge H = x \geq 0 \wedge g > 0 \rightarrow \left[\left(\{x' = v, v' = -g \ \& \ x \geq 0\}; \right. \right. \\ \left. \left. \text{if } (x = 0) \ v := -cv \right)^* \right] 0 \leq x \leq H$$





Example (▶ Bouncing Ball)

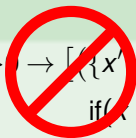
$$v \leq 0 \wedge 1 \geq c \geq 0 \wedge H = x \geq 0 \wedge g > 0 \rightarrow [(\{x' = v, v' = -g \& x \geq 0\}; \\ \text{if}(x = 0) v := -cv)^*] 0 \leq x \leq H$$

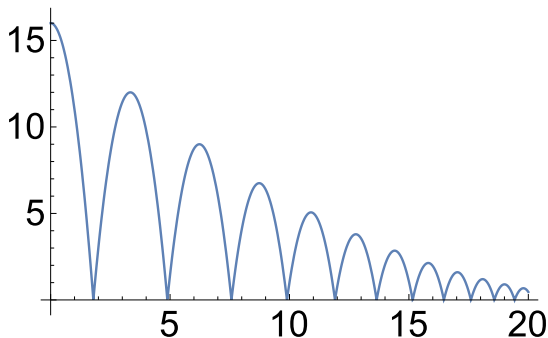


Not if $v \ll 0$ initial dribbling

Example (▶ Bouncing Ball)

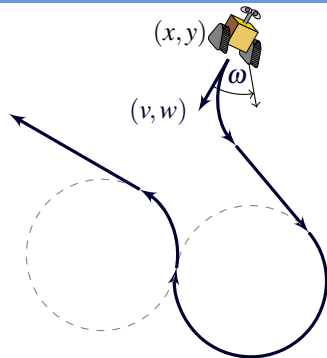
$$v \leq 0 \wedge 1 \geq c \geq 0 \wedge H = x \geq 0 \wedge g > 0 \rightarrow \left[\left(\{x' = v, v' = -g \ \& \ x \geq 0\}; \right. \right. \\ \left. \left. \text{if } (x = 0) \ v := -cv \right)^* \right] \ 0 \leq x \leq H$$

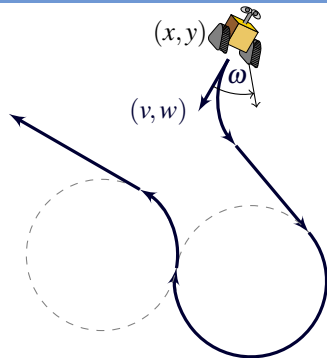




Example (▶ Bouncing Ball)

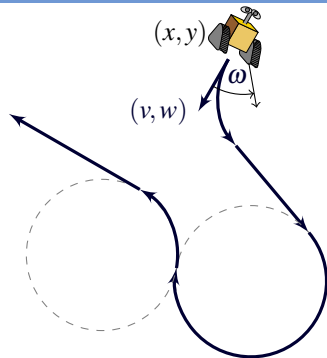
$$v=0 \wedge 1 \geq c \geq 0 \wedge H=x \geq 0 \wedge g > 0 \rightarrow [(\{x' = v, v' = -g \& x \geq 0\}; \\ \text{if}(x = 0) v := -cv)^*] 0 \leq x \leq H$$





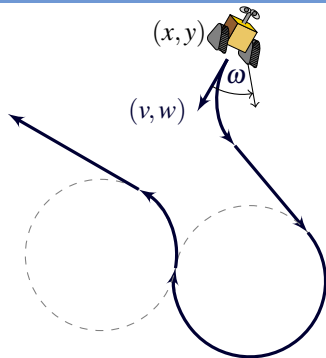
Example (Runaround Robot)

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



Example (Runaround Robot)

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



Example (Runaround Robot)

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



- 1 Learning Objectives
- 2 Quantum the Acrophobic Bouncing Ball
- 3 Contracts for CPS
 - Safety of Robots
 - Safety of Bouncing Balls
- 4 Logical Formulas for Hybrid Programs
- 5 Differential Dynamic Logic
 - Syntax
 - Semantics
 - Notational Convention
- 6 Identifying Requirements of a CPS
- 7 Summary

Definition (Hybrid program α)

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

Definition (dL Formula P)

$$P, Q ::= e \geq \tilde{e} \mid \neg P \mid P \wedge Q \mid \forall x P \mid \exists x P \mid [\alpha]P \mid \langle \alpha \rangle P$$

Discrete
Assign

Test
Condition

Differential
Equation

Nondet.
Choice

Seq.
Compose

Nondet.
Repeat

Definition (Hybrid program α)

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

Definition (dL Formula P)

$$P, Q ::= e \geq \tilde{e} \mid \neg P \mid P \wedge Q \mid \forall x P \mid \exists x P \mid [\alpha]P \mid \langle \alpha \rangle P$$

All
Reals

Some
Reals

All
Runs

Some
Runs

Definition (Hybrid program semantics)

 $([\cdot] : \text{HP} \rightarrow \wp(\mathcal{S} \times \mathcal{S}))$

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

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

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

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

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

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

compositional semantics

Definition (dL semantics)

 $([\cdot] : \text{Fml} \rightarrow \wp(\mathcal{S}))$

$$[e \geq \tilde{e}] = \{\omega : \omega[e] \geq \omega[\tilde{e}]\}$$

$$[\neg P] = [P]^c$$

$$[P \wedge Q] = [P] \cap [Q]$$

$$[\langle \alpha \rangle P] = [\alpha] \circ [P] = \{\omega : \nu \in [P] \text{ for some } \nu : (\omega, \nu) \in [\alpha]\}$$

$$[[\alpha]P] = [\neg \langle \alpha \rangle \neg P] = \{\omega : \nu \in [P] \text{ for all } \nu : (\omega, \nu) \in [\alpha]\}$$

$$[\exists x P] = \{\omega : \omega'_x \in [P] \text{ for some } r \in \mathbb{R}\}$$



André Platzer.

Logical Foundations of Cyber-Physical Systems.

Springer, Cham, 2018.

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.

Springer, Heidelberg, 2010.

URL: <http://www.springer.com/978-3-642-14508-7>,
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](https://doi.org/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).