KeYmaera X Tutorial
Tactics and Proofs for Cyber-Physical Systems

Nathan Fulton  Stefan Mitsch  André Platzer

http://keymaeraX.org/
Computer Science Department
Carnegie Mellon University, Pittsburgh, PA

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
1. KeYmaera X Overview
   - Tutorial Objectives

2. Differential Dynamic Logic for Hybrid Systems
   - Syntax: Notation for Verification Questions
   - Semantics: Meaning of the Syntax
   - Example: Car Control Design
   - Example: Branching Structure

3. Proofs for CPS
   - Compositional Proof Calculus
   - Example: Safe Car Control

4. Differential Invariants for Differential Equations
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots

5. Synthesize Monitors

6. Case Studies

7. Summary
1 KeYmaera X Overview
   - Tutorial Objectives
2 Differential Dynamic Logic for Hybrid Systems
   - Syntax: Notation for Verification Questions
   - Semantics: Meaning of the Syntax
   - Example: Car Control Design
   - Example: Branching Structure
3 Proofs for CPS
   - Compositional Proof Calculus
   - Example: Safe Car Control
4 Differential Invariants for Differential Equations
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots
5 Synthesize Monitors
6 Case Studies
7 Summary
Correctness Questions in Cyber-Physical System Design

Safety  The system must be safe under all circumstances

Liveness  The system must reach a given goal

How do we make cyber-physical systems safe?

Extensive testing?  When are we done? How many test cases are enough? Did we cover all relevant tests?

Code reviews?
### Benefits of Logical Foundations for CPS V & V

<table>
<thead>
<tr>
<th>Proofs</th>
<th>LICS’12, JAR’16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Safety</strong></td>
<td>Formalize system properties: What is “Safe”? “Reach goal”?</td>
</tr>
<tr>
<td><strong>Models</strong></td>
<td>Formalize system models, clarify behavior</td>
</tr>
<tr>
<td><strong>Assumptions</strong></td>
<td>Make assumptions explicit rather than silently</td>
</tr>
<tr>
<td><strong>Predictions</strong></td>
<td>Safety analysis predicts behavior for infinitely many cases</td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
<td>Reveal invariants, switching conditions, operating conditions</td>
</tr>
<tr>
<td><strong>Design</strong></td>
<td>Invariants/proofs guide safe controller design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byproducts</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analysis</strong></td>
<td>Determine design trade-offs &amp; feasibility early</td>
</tr>
<tr>
<td><strong>Synthesis</strong></td>
<td>Turn models into code &amp; safety monitors</td>
</tr>
<tr>
<td><strong>Certificate</strong></td>
<td>Proofs as evidence for certification</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tools</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KeYmaera X</strong></td>
<td>aXiomatic Tactical Theorem Prover for CPS</td>
</tr>
</tbody>
</table>
An axiomatic Tactical Theorem Prover for CPS

KeYmaera X

http://keymaeraX.org/

https://www.keymaeraX.org/CADE-15

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Core</td>
<td>Increases trust, modularity, enables experimentation (1652)</td>
</tr>
<tr>
<td>Tactics</td>
<td>Bridging between small core and powerful reasoning steps (Hilbert) (Sequent)</td>
</tr>
<tr>
<td>Separation</td>
<td>Tactics can make courageous inferences, Core establishes soundness</td>
</tr>
<tr>
<td>Search&amp;Do</td>
<td>Search-based tactics follow proof search strategies, Constructive tactics directly build a proof</td>
</tr>
<tr>
<td>Interaction</td>
<td>Interactive proofs mixed with tactical proofs and proof search</td>
</tr>
<tr>
<td>Extensible</td>
<td>Flexible for new algorithms, new tactics, new logics, new proof rules, new axioms, ...</td>
</tr>
<tr>
<td>Customize</td>
<td>Modular user interface, API</td>
</tr>
</tbody>
</table>

An aXiomatic Tactical Theorem Prover for CPS

<table>
<thead>
<tr>
<th>Tool</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeYmaera X</td>
<td>1652</td>
</tr>
<tr>
<td>KeYmaera</td>
<td>65989</td>
</tr>
<tr>
<td>KeY</td>
<td>51328</td>
</tr>
<tr>
<td>Nuprl</td>
<td>15000 + 50000</td>
</tr>
<tr>
<td>MetaPRL</td>
<td>8196</td>
</tr>
<tr>
<td>Isabelle/Pure</td>
<td>8913</td>
</tr>
<tr>
<td>Coq</td>
<td>16538</td>
</tr>
<tr>
<td>HOL Light</td>
<td>396</td>
</tr>
<tr>
<td>PHAVer</td>
<td>30000</td>
</tr>
<tr>
<td>HSolver</td>
<td>20000</td>
</tr>
<tr>
<td>SpaceEx</td>
<td>100000</td>
</tr>
<tr>
<td>Flow*</td>
<td>25000</td>
</tr>
<tr>
<td>dReal</td>
<td>50000 + millions</td>
</tr>
<tr>
<td>HyCreate2</td>
<td>6081 + user model analysis</td>
</tr>
</tbody>
</table>

Disclaimer: Self-reported estimates of the soundness-critical lines of code + rules
Challenge (Hybrid Systems)

Fixed rule describing state evolution with both

- Discrete dynamics (control decisions)
- Continuous dynamics (differential equations)
CPS Analysis

Challenge (Hybrid Systems)

Fixed rule describing state evolution with both

- Discrete dynamics (control decisions)
- Continuous dynamics (differential equations)
Challenge (Hybrid Systems)

Fixed rule describing state evolution with both

- **Discrete dynamics** (control decisions)
- **Continuous dynamics** (differential equations)
CPS Analysis

Challenge (Hybrid Systems)

Fixed rule describing state evolution with both
- Discrete dynamics (control decisions)
- Continuous dynamics (differential equations)
CPSs are Multi-Dynamical Systems

CPS Dynamics

CPS are characterized by multiple facets of dynamical systems.

CPS Compositions

CPS combines multiple simple dynamical effects.

Descriptive simplification

Tame Parts

Exploiting compositionality tames CPS complexity.

Analytic simplification

Nathan Fulton, Stefan Mitsch, André Platzer KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Learning Objectives

Use KeYmaera X to:

1. Model cyber-physical systems
2. Express safety/correctness properties
3. Find bugs in a system design
4. Identify safety constraints
5. Identify system invariants
6. Verify the final system design
7. Write automated proof tactics
8. Prove differential equations
9. Synthesize correct-by-construction runtime monitors
1. **KeYmaera X Overview**
   - Tutorial Objectives

2. **Differential Dynamic Logic for Hybrid Systems**
   - Syntax: Notation for Verification Questions
   - Semantics: Meaning of the Syntax
   - Example: Car Control Design
   - Example: Branching Structure

3. **Proofs for CPS**
   - Compositional Proof Calculus
   - Example: Safe Car Control

4. **Differential Invariants for Differential Equations**
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots

5. **Synthesize Monitors**

6. **Case Studies**

7. **Summary**
1. KeYmaera X Overview
   - Tutorial Objectives

2. Differential Dynamic Logic for Hybrid Systems
   - Syntax: Notation for Verification Questions
   - Semantics: Meaning of the Syntax
   - Example: Car Control Design
   - Example: Branching Structure

3. Proofs for CPS
   - Compositional Proof Calculus
   - Example: Safe Car Control

4. Differential Invariants for Differential Equations
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots

5. Synthesize Monitors

6. Case Studies

7. Summary
Concept (Differential Dynamic Logic) (JAR’08, LICS’12)

\[ \lbrack \alpha \rbrack \varphi \quad \alpha \quad \varphi \]

\[ x \neq m \land b > 0 \]

\[ \text{init} \rightarrow (\text{if} (SB(x, m)) a := -b \land x' = v, v' = a) \]

\[ x \neq m \]

\[ \text{post} \]

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
CPS Analysis

Concept (Differential Dynamic Logic) (JAR’08, LICS’12)

\[ [\alpha] \varphi \mapsto \varphi \]

\[ \square x \neq m \mapsto x \neq m \]

\[ x \neq m \]

\[ \square x \neq m \mapsto x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]
Concept (Differential Dynamic Logic) (JAR’08, LICS’12)

\[ x \neq m \]

\[ \Box \varphi \quad \alpha \quad \varphi \]

\[ [\alpha] \varphi \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]
CPS Analysis

Concept (Differential Dynamic Logic) (JAR’08, LICS’12)

\[ [\alpha] \varphi \]

\[ x \neq m \]

\[ x' = v, v' = a \]

\[ \text{ODE} \]
CPS Analysis

Concept (Differential Dynamic Logic) (JAR’08, LICS’12)

$$[\alpha] \varphi \quad \alpha \xrightarrow{} \varphi$$

$$x \neq m$$

$$a := -b \quad x' = v, v' = a$$

$$\text{assign}$$

$$\text{ODE}$$
CPS Analysis

Concept (Differential Dynamic Logic) (JAR’08, LICS’12)

\[ [\alpha] \varphi \quad \alpha \quad \varphi \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ (\text{if}(SB(x, m)) a := -b) \quad x' = v, v' = a \]

\[ \text{test} \quad \text{assign} \]

\[ t \]

\[ a \]

\[ m \]

\[ t \]

\[ x \]

\[ t \]

\[ t \]
Concept (Differential Dynamic Logic) (JAR’08, LICS’12)

\[ \alpha \Diamond \varphi \]

\[
\begin{aligned}
&\text{if} (SB(x, m)) a := -b) \; ; \; x' = v, v' = a
\end{aligned}
\]
Concept (Differential Dynamic Logic) (JAR’08,LICS’12)

\[(\text{if}(SB(x, m)) a := -b) ; \ x' = v, v' = a\]
Concept (Differential Dynamic Logic) \[ (JAR’08, LICS’12) \]

\[
[\alpha] \varphi \\
\alpha \rightarrow \varphi
\]

\[
[(\text{if}(\text{SB}(x, m)) \ a := -b) ; \ x' = v, v' = a)^*]x \neq m
\]

\[
\begin{align*}
 & a \\
 & v \\
 & x
\end{align*}
\]

\[
\begin{align*}
 & m \\
 & t
\end{align*}
\]
Concept (Differential Dynamic Logic) (JAR’08, LICS’12)

\[ x \neq m \quad \text{(init)} \]

\[ \text{init} \rightarrow \left[ ((\text{if}(\text{SB}(x, m)) a := -b) ; x' = v, v' = a)^* \right] x \neq m \quad \text{(post)} \]

\[ x \neq m \quad \text{(JAR’08, LICS’12)} \]
**Definition (Hybrid program \( \alpha \))**

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

**Definition (d\( \mathcal{L} \) Formula \( P \))**

\[
e \geq \tilde{e} \mid \neg P \mid P \land Q \mid \forall x \; P \mid \exists x \; P \mid [\alpha]P \mid \langle \alpha \rangle P
\]
Differential Dynamic Logic $\mathcal{dL}$: Syntax

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

### Definition ($\mathcal{dL}$ Formula $P$)
\[
e \geq \tilde{e} \mid \neg P \mid P \land Q \mid \forall x \ P \mid \exists x \ P \mid \left\llbracket \alpha \right\rrbracket P \mid \left\langle \alpha \right\rangle P
\]

Tableaux’07, JAutomReas’08, LICS’12
## Differential Dynamic Logic $\mathcal{dL}$: Semantics

**Definition (Hybrid program semantics)**  
\[
\begin{align*}
[x := e] & = \{(ω, ν) : ν = ω \text{ except } ν[x] = ω[e]\} \\
[?Q] & = \{(ω, ω) : ω ∈ [Q]\} \\
[x' = f(x)] & = \{(φ(0), φ(r)) : φ \models x' = f(x) \text{ for some duration } r\} \\
[α ∪ β] & = [α] ∪ [β] \\
[α; β] & = [α] ◦ [β] \\
[α^*] & = \bigcup_{n ∈ \mathbb{N}} [α^n]
\end{align*}
\]

**Definition ($\mathcal{dL}$ semantics)**  
\[
\begin{align*}
[e ≥ \tilde{e}] & = \{ω : ω[e] ≥ ω[\tilde{e}]\} \\
[¬P] & = [P]^C \\
[P ∧ Q] & = [P] ∩ [Q] \\
[⟨α⟩P] & = [α] ◦ [P] = \{ω : ν ∈ [P] \text{ for some } ν : (ω, ν) ∈ [α]\} \\
[[α]P] & = [¬⟨α⟩¬P] = \{ω : ν ∈ [P] \text{ for all } ν : (ω, ν) ∈ [α]\} \\
[∃x P] & = \{ω : ω^r_x ∈ [P] \text{ for some } r ∈ \mathbb{R}\}
\end{align*}
\]
Differential Dynamic Logic $d\mathcal{L}$: Semantics

1. $x := e$  
   \[ \omega \xrightarrow{\text{x := e}} \nu \]

2. $x' = f(x) \& Q$  
   \[ \omega \xrightarrow{\text{x' = f(x) \& Q}} \nu \]

3. ?$Q$  
   \[ \omega \rightarrow \quad \text{if } \omega \in \llbracket Q \rrbracket \]

4. $\nu$ if $\nu(x) = \omega [\llbracket e \rrbracket]$ and $\nu(z) = \omega(z)$ for $z \neq x$  
   \[ \omega \xrightarrow{\nu \text{ if } \nu(x) = \omega [\llbracket e \rrbracket] \text{ and } \nu(z) = \omega(z) \text{ for } z \neq x} t \]

5. $0$  
   \[ 0 \rightarrow \quad \text{no change if } \omega \in \llbracket Q \rrbracket \]

6. otherwise no transition  
   \[ 0 \rightarrow \quad \text{no change if } \omega \in \llbracket Q \rrbracket \]

7. $\omega$ \quad \text{no change if } \omega \in \llbracket Q \rrbracket \quad \text{otherwise no transition}  
   \[ \omega \rightarrow \quad \text{no change if } \omega \in \llbracket Q \rrbracket \quad \text{otherwise no transition} \quad t \]
Differential Dynamic Logic $\mathcal{DL}$: Semantics

\[ \omega
\]

\[ \nu_1
\]

\[ \nu_2
\]

\[ \alpha \cup \beta
\]

\[ \alpha
\]

\[ \beta
\]

\[ \alpha ; \beta
\]

\[ \omega
\]

\[ s
\]

\[ \nu
\]

\[ \omega
\]

\[ \nu
\]

\[ \omega
\]

\[ \omega_1
\]

\[ \omega_2
\]

\[ \nu
\]

\[ \alpha^*
\]

\[ t
\]

\[ \nu_1
\]

\[ \nu_2
\]
Differential Dynamic Logic $d\mathcal{L}$: Semantics
Differential Dynamic Logic $\mathcal{DL}$: Semantics
Definition (dŁ Formulas)

\[ \omega \]

\[ [\alpha]P \]

\[ P \]

\[ P \]

\[ P \]
Definition (dŁ Formulas)

\[ \omega \langle \alpha \rangle P \]

Compositional semantics \implies \text{compositional proofs!}
Definition (d\(\mathcal{L}\) Formulas)

\[ [\alpha] P \]

\(\omega\)-span

\(\alpha\)-span
Definition (dŁ Formulas)

\[ [\alpha]P \]
\[ \langle \beta \rangle P \]
\[ \omega \]

\( \alpha \)-span

\( \beta \)-span

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Definition (d$\mathcal{L}$ Formulas)
Definition (dŁ Formulas)

\[ \langle \beta \rangle P \]

\[ [\alpha]P \]

\[ \omega \]

\[ \beta \text{-span} \]

\[ \langle \beta \rangle P \]

\[ [\alpha]P \]

\[ \omega \]

\[ \alpha \text{-span} \]

compositional semantics ⇒ compositional proofs!
Example (Single car $\text{car}_s$)

$x' = v, \ v' = a$
Ex: Car Control

Control decision: accelerate or brake

Example (Single car $car_s$)

$ (a := A \cup a := -b); \quad x' = v, v' = a$
Ex: Car Control

Repeat control decisions

Example (Single car $\text{cars}$)

$$((a := A \cup a := -b); x' = v, v' = a)^*$$
repeat control decisions

example (single car \( car_s \))

\[
\left( a := A \cup a := -b \right); x' = v, v' = a
\]
Ex: Car Control

Velocity bound $v \geq 0$

Example (Single car $car_s$)

$$((a := A \cup a := -b); \quad x' = v, v' = a \& v \geq 0)^*$$
Accelerate not always safe

Example (Single car $cars$)

\[
(( a := A \cup a := -b); \ x' = v, \ v' = a \land v \geq 0)^*
\]
Accelerate condition $?Q$

Example (Single car $car_s$)

$(((?Q; a := A) \cup a := -b); \ x' = v, v' = a \land v \geq 0)^*$
Accelerate condition $Q$ depends on $A$

Example (Single car $\text{car}_s$)

$$(((Q; a := 0) \cup a := -b); x' = v, v' = a \& v \geq 0)^*$$
Ex: Car Control Properties

\[ Q \equiv \]

Example (Single car \(\text{car}_\varepsilon\) time-triggered)

\[
(((?Q; a := A) \cup a := -b); \ t := 0; \ x' = v, v' = a, t' = 1 \& v \geq 0 \& t \leq \varepsilon)\]

Example (\(\triangleright\) Safely stays before traffic light \(m\))

\[
A \geq 0 \& b > 0 \rightarrow [\text{car}_\varepsilon] x \leq m
\]
Ex: Car Control Properties

time-triggered

\[ Q \equiv 2b(m - x) \geq v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v) \]

Example (Single car \(car_\varepsilon\) time-triggered)

\[ (((?Q; a := A) \cup a := -b); \quad t := 0; \quad x' = v, v' = a, \quad t' = 1 \& v \geq 0 \& t \leq \varepsilon)^* \]

Example (_publish Safely stays before traffic light \(m\))

\[ v^2 \leq 2b(m - x) \& A \geq 0 \& b > 0 \rightarrow [\text{car}_\varepsilon] x \leq m \]
\[ Q \equiv 2b(m - x) \geq v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v) \]

Example (Single car \(\text{car}_\varepsilon\) time-triggered)

\[ (((?Q; a := A) \cup a := -b); \ t := 0; \ x' = v, v' = a, t' = 1 \& v \geq 0 \& t \leq \varepsilon)^* \]

Example (Live, can move everywhere)

\[ \varepsilon > 0 \& A > 0 \& b > 0 \rightarrow \forall p \exists m (\langle \text{car}_\varepsilon \rangle x \geq p) \]
Branching Transitions in Hybrid Programs

Robot $\equiv (\text{ctrl} ; \text{drive})^*$
\[
\begin{align*}
\text{ctrl} & \equiv (?m - x \leq SB; a := -b) \\
& \cup (?m - x \geq SB; a := A) \\
\text{drive} & \equiv t := 0; (x' = v, v' = a, t' = 1 \\
& \& v \geq 0 \& t \leq \varepsilon)
\end{align*}
\]
Robot ≡ (ctrl ; drive)*

ctrl ≡ (?m − x ≤ SB; a := −b)
∪ (?m − x ≥ SB; a := A)

drive ≡ t := 0; (x′ = v, v′ = a, t′ = 1
& v ≥ 0 ∧ t ≤ ε)
Robot \equiv (\text{ctrl} ; \text{drive})^*

\text{ctrl} \equiv (?m - x \leq SB; \ a := -b) \cup (?m - x \geq SB; \ a := A)

\text{drive} \equiv t := 0; (x' = v, \ \nu' = a, \ t' = 1 \\
& \& \nu \geq 0 \land t \leq \varepsilon)
Branching Transitions in Hybrid Programs

Robot ≡ (ctrl ; drive)*

ctrl ≡ (?m − x ≤ SB; a := −b)
∪ (?m − x ≥ SB; a := A)

drive ≡ t := 0; (x' = v, v' = a, t' = 1
& v ≥ 0 ∧ t ≤ ε)
Branching Transitions in Hybrid Programs

Robot \equiv (\text{ctrl} ; \text{drive})^*
\begin{align*}
\text{ctrl} & \equiv (?m - x \leq SB; a := -b) \\
& \quad \cup (?m - x \geq SB; a := A)
\end{align*}
\begin{align*}
\text{drive} & \equiv t := 0; (x' = v, v' = a, t' = 1) \\
& \quad \quad \quad \& v \geq 0 \land t \leq \varepsilon)
Branching Transitions in Hybrid Programs

Robot ≡ (ctrl ; drive)*

ctrl ≡ (?m – x ≤ SB; a := –b)
∪ (?m – x ≥ SB; a := A)

drive ≡ t := 0; (x′ = v, v′ = a, t′ = 1
& v ≥ 0 ∧ t ≤ ε)
Branching Transitions in Hybrid Programs

Robot ≡ (ctrl ; drive)∗

ctrl ≡ (?m−x ≤ SB; a := −b)
∪ (?m−x ≥ SB; a := A)

drive ≡ t := 0; (x′ = v, v′ = a, t′ = 1
& v ≥ 0 ∧ t ≤ ε)
Branching Transitions in Hybrid Programs

Robot ≡ (ctrl ; drive)∗

ctrl ≡ (?m − x ≤ SB; a := −b)
∪ (?m − x ≥ SB; a := A)

drive ≡ t := 0; (x′ = v, v′ = a, t′ = 1
& v ≥ 0 ∧ t ≤ ε)
Robot ≡ (ctrl ; drive)∗

ctrl ≡ (?m − x ≤ SB; a := −b)
∪ (?m − x ≥ SB; a := A)

drive ≡ t := 0; (x′ = ν, ν′ = a, t′ = 1
& ν ≥ 0 ∧ t ≤ ε)
Branching Transitions in Hybrid Programs

\[ m - x \leq SB \quad \text{or} \quad m - x \geq SB \]

\[ a := -b \quad \text{or} \quad a := A \quad \text{if} \quad (Q) \alpha \quad \text{else} \quad \beta \equiv (\ ? Q ; \alpha) \quad \cup \quad (\ ? \neg Q ; \beta) \]

\[ t := 0 \quad \text{and} \quad t' = 1 \quad \text{if} \quad (Q) \alpha \quad \text{else} \quad \beta \]

Robot \equiv (ctrl ; drive)^*

ctrl \equiv (?m - x \leq SB; a := -b) \quad \cup \quad (?m - x \geq SB; a := A)

drive \equiv t := 0; (x' = v, v' = a, t' = 1 \quad \text{and} \quad v \geq 0 \land t \leq \epsilon)
Branching Transitions in Hybrid Programs

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

\text{ctrl} \equiv (?m - x \leq SB; a := -b)
\bigcup (?m - x \geq SB; a := A)

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

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

ctrl \equiv (?m - x \leq SB; \ a := -b) \\
\quad \cup (?m - x \geq SB; \ a := A)

drive \equiv t := 0; (x' = v, v' = a, t' = 1 \\
\quad \quad \quad \quad \quad \quad \ & v \geq 0 \land t \leq \varepsilon)
Branching Transitions in Hybrid Programs

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

\text{ctrl} \equiv (?m - x \leq SB; a := -b)
\cup (?m - x \geq SB; a := A)

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

Robot ≡ (ctrl ; drive)^*  
ctrl ≡ (?m – x ≤ SB; a := –b)  
∪ (?m – x ≥ SB; a := A)  
drive ≡ t := 0; (x' = v, v' = a, t' = 1  
& v ≥ 0 & t ≤ ε)
Robot ≡ (ctrl ; drive)*

\[ \text{ctrl} \equiv (?m - x \leq SB; a := -b) \]
\[ \cup (?m - x \geq SB; a := A) \]
\[ \text{drive} \equiv t := 0; (x' = v, v' = a, t' = 1 \]
\[ & v \geq 0 \land t \leq \varepsilon) \]
Branching Transitions in Hybrid Programs

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

while$(Q)\alpha \equiv$

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

ctrl $\equiv (?m - x \leq \text{SB}; a := -b)$

$\cup (\ ?m - x \geq \text{SB}; a := A)$

drive $\equiv t := 0; (x' = v, v' = a, t' = 1$

$\& v \geq 0 \& t \leq \varepsilon)$
Branching Transitions in Hybrid Programs

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

while \((Q) \alpha \equiv \)

Robot \equiv (ctrl ; drive)^*

ctrl \equiv (?m – x \leq \text{SB}; a := –b)
\text{ } \cup (?m – x \geq \text{SB}; a := A)

drive \equiv t := 0; (x' = \nu, \nu' = a, t' = 1
\text{ } \& \nu \geq 0 \land t \leq \varepsilon)
Branching Transitions in Hybrid Programs

\[
\begin{align*}
\text{if}(Q) \alpha \text{ else } \beta & \equiv (?Q; \alpha) \cup (\neg Q; \beta) \\
\text{while}(Q) \alpha & \equiv (?Q; \alpha)^*; ?\neg Q
\end{align*}
\]

Robot \equiv (ctrl ; drive)^*

ctrl \equiv (?m - x \leq SB; a := -b) \\
\cup (?m - x \geq SB; a := A)

drive \equiv t := 0; (x' = v, v' = a, t' = 1 \\
& \quad \& v \geq 0 \land t \leq \varepsilon)
Outline

1. KeYmaera X Overview
   - Tutorial Objectives

2. Differential Dynamic Logic for Hybrid Systems
   - Syntax: Notation for Verification Questions
   - Semantics: Meaning of the Syntax
   - Example: Car Control Design
   - Example: Branching Structure

3. Proofs for CPS
   - Compositional Proof Calculus
   - Example: Safe Car Control

4. Differential Invariants for Differential Equations
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots

5. Synthesize Monitors

6. Case Studies

7. Summary
Differential Dynamic Logic: Axiomatization

\[x := e \] \( P(x) \iff P(e)\)

\[? \] \( [? Q] P \iff (Q \to P)\)

\[\prime \] \( [x' = f(x)] P \iff \forall t \geq 0 \; [x := y(t)] P \quad \text{(} y'(t) = f(y)\text{)}\)

\[\cup \] \( [\alpha \cup \beta] P \iff [\alpha] P \land [\beta] P\)

\[;\] \( [\alpha; \beta] P \iff [\alpha][\beta] P\)

\[\ast \] \( [\alpha^\ast] P \iff P \land [\alpha][\alpha^\ast] P\)

\(K\) \( [\alpha](P \to Q) \to ([\alpha]P \to [\alpha]Q)\)

\(L\) \( [\alpha^\ast](P \to [\alpha]P) \to (P \to [\alpha^\ast]P)\)

\(C\) \( [\alpha^\ast]\forall v > 0 \; (P(v) \to \langle \alpha \rangle P(v-1)) \to \forall v \; (P(v) \to \langle \alpha^\ast \rangle \exists v \leq 0 \; P(v))\)

equations of truth
Proofs for Hybrid Systems

compositional semantics ⇒ compositional rules!
\[ [\alpha \cup \beta] P \leftrightarrow [\alpha] P \land [\beta] P \]
\[ [\alpha \cup \beta]P \leftrightarrow [\alpha]P \land [\beta]P \]

\[ [\alpha; \beta]P \leftrightarrow [\alpha][\beta]P \]
Proofs for Hybrid Systems

\[ [\alpha \cup \beta]P \leftrightarrow [\alpha]P \land [\beta]P \]

\[ [\alpha; \beta]P \leftrightarrow [\alpha][\beta]P \]

\[ P \quad P \rightarrow [\alpha]P \quad \frac{}{[\alpha^*]P} \]

Nathan Fulton, Stefan Mitsch, André Platzer KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Example Proof: Safe Braking

\[ J(x, v) \equiv x \leq m \]

\[ ; \quad J(x, v) \vdash [a := -b; (x' = v, v' = a)]J(x, v) \]

1. \( \Gamma \vdash \Delta \) shape of conjecture to prove
2. \( \Gamma \) is list of available assumptions
3. \( \Delta \) needs to be proved from assumptions \( \Gamma \)
4. Proof reduces desired conclusion (at the bottom) to premises with remaining subgoals (top) until no more subgoals (*)
Example Proof: Safe Braking

\[ J(x, v) \equiv x \leq m \]

\[ \begin{align*}
[=] & \quad J(x, v) \vdash [a := -b][x' = v, v' = a]J(x, v) \\
[;] & \quad J(x, v) \vdash [a := -b; (x' = v, v' = a)]J(x, v)
\end{align*} \]

1. \( \Gamma \vdash \Delta \) shape of conjecture to prove
2. \( \Gamma \) is list of available assumptions
3. \( \Delta \) needs to be proved from assumptions \( \Gamma \)
4. Proof reduces desired conclusion (at the bottom) to premises with remaining subgoals (top) until no more subgoals (\(*\)
Example Proof: Safe Braking

\[ J(x, v) \equiv x \leq m \]

\[\begin{align*}
[1] & \quad J(x, v) \vdash [x' = v, v' = -b]J(x, v) \\
[\colonequals] & \quad J(x, v) \vdash [a := -b][x' = v, v' = a]J(x, v) \\
[;] & \quad J(x, v) \vdash [a := -b; (x' = v, v' = a)]J(x, v)
\end{align*}\]

1. \( \Gamma \vdash \Delta \) shape of conjecture to prove
2. \( \Gamma \) is list of available assumptions
3. \( \Delta \) needs to be proved from assumptions \( \Gamma \)
4. Proof reduces desired conclusion (at the bottom) to premises with remaining subgoals (top) until no more subgoals (\( \ast \))
Example Proof: Safe Braking

\[ J(x, v) \equiv x \leq m \]

\[
\begin{align*}
\Gamma &\vdash \Delta \quad \text{shape of conjecture to prove} \\
\Gamma &\vdash \text{antecedent} \\
\Delta &\vdash \text{succedent} \\
\text{Proof reduces desired conclusion (at the bottom)} &\text{to premises with remaining subgoals (top) until no more subgoals (*)}
\end{align*}
\]
Example Proof: Safe Braking

\[ J(x, v) \equiv x \leq m \]

\[
\begin{align*}
\text{QE}&: J(x, v) \vdash \forall t \geq 0 \left( -\frac{b}{2} t^2 + vt + x \leq m \right) \\
[=]:& J(x, v) \vdash \forall t \geq 0 \left[ x := -\frac{b}{2} t^2 + vt + x \right] J(x, v) \\
[\l']:& J(x, v) \vdash \left[ x' = v, v' = -b \right] J(x, v) \\
[=]:& J(x, v) \vdash \left[ a := -b \right] \left[ x' = v, v' = a \right] J(x, v) \\
[;]:& J(x, v) \vdash \left[ a := -b; (x' = v, v' = a) \right] J(x, v)
\end{align*}
\]

\begin{enumerate}
\item \( \Gamma \vdash \Delta \) shape of conjecture to prove (sequent)
\item \( \Gamma \) is list of available assumptions (antecedent)
\item \( \Delta \) needs to be proved from assumptions \( \Gamma \) (succeedent)
\item Proof reduces desired conclusion (at the bottom) to premises with remaining subgoals (top) until no more subgoals (\( * \))
\end{enumerate}
Example Proof: Safe Braking

\[ J(x, v) \equiv x \leq m \]

\[
\begin{align*}
J(x, v) & \vdash v^2 \leq 2b(m - x) \\
\text{QE} & \\
J(x, v) & \vdash \forall t \geq 0 \left( -\frac{b}{2} t^2 + vt + x \leq m \right) \\
[=] & \\
J(x, v) & \vdash \forall t \geq 0 \left[ x := -\frac{b}{2} t^2 + vt + x \right] J(x, v) \\
['] & \\
J(x, v) & \vdash [x' = v, v' = -b] J(x, v) \\
[=] & \\
J(x, v) & \vdash [a := -b] \left[ x' = v, v' = a \right] J(x, v) \\
[;] & \\
J(x, v) & \vdash \left[ a := -b; (x' = v, v' = a) \right] J(x, v)
\end{align*}
\]

1. \( \Gamma \vdash \Delta \) shape of conjecture to prove (sequent)
2. \( \Gamma \) is list of available assumptions (antecedent)
3. \( \Delta \) needs to be proved from assumptions \( \Gamma \) (succeedent)
4. Proof reduces desired conclusion (at the bottom) to premises with remaining subgoals (top) until no more subgoals (*)
Example Proof: Safe Braking

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]

\[ J(x, v) \vdash v^2 \leq 2b(m - x) \]

\[ \text{QE} \quad J(x, v) \vdash \forall t \geq 0 \left( -\frac{b}{2} t^2 + vt + x \leq m \right) \]

\[ [:=] \quad J(x, v) \vdash \forall t \geq 0 \left[ x := -\frac{b}{2} t^2 + vt + x \right] J(x, v) \]

\[ ['] \quad J(x, v) \vdash [x' = v, v' = -b] J(x, v) \]

\[ [:=] \quad J(x, v) \vdash [a := -b][x' = v, v' = a] J(x, v) \]

\[ [;] \quad J(x, v) \vdash [a := -b; (x' = v, v' = a)] J(x, v) \]
Example Proof: Safe Accelerating

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]

\[ J(x, v) \vdash [\neg \text{SB}; a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v) \]
Example Proof: Safe Accelerating

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]

\[
\begin{align*}
[?] & \quad J(x, v) \vdash [? \neg SB][a := A; (x' = v, \nu' = a, t' = 1 \& t \leq \varepsilon)]J(x, v) \\
[;] & \quad J(x, v) \vdash [? \neg SB; a := A; (x' = v, \nu' = a, t' = 1 \& t \leq \varepsilon)]J(x, v)
\end{align*}
\]
Example Proof: Safe Accelerating

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]

\[
\begin{align*}
\text{[\!\!] J(x, v) &\vdash \neg \text{SB} \rightarrow [a := A; (x' = v, \nu' = a, t' = 1 \& t \leq \varepsilon)]J(x, v) \\
\text{[?]} J(x, v) &\vdash [\neg \text{SB}][a := A; (x' = v, \nu' = a, t' = 1 \& t \leq \varepsilon)]J(x, v) \\
\text{[\!\!] J(x, v) &\vdash [\neg \text{SB}; a := A; (x' = v, \nu' = a, t' = 1 \& t \leq \varepsilon)]J(x, v)
\end{align*}
\]
Example Proof: Safe Accelerating

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]

\[ \vdash \neg SB \rightarrow [a := A][x' = v, v' = a, t' = 1 & t \leq \varepsilon]J(x, v) \]

\[ \vdash \neg SB \rightarrow [a := A; (x' = v, v' = a, t' = 1 & t \leq \varepsilon)]J(x, v) \]

\[ \text{[?] } \vdash [\neg SB][a := A; (x' = v, v' = a, t' = 1 & t \leq \varepsilon)]J(x, v) \]

\[ \vdash [\neg SB; a := A; (x' = v, v' = a, t' = 1 & t \leq \varepsilon)]J(x, v) \]
\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]

\[ \quad \vdash \neg \text{SB} \rightarrow \left[ x' = v, v' = A, t' = 1 \& t \leq \varepsilon \right] J(x, v) \]

\[ \quad 
\begin{align*}
\begin{array}{c}
\vdash J(x, v) \rightarrow \neg \text{SB} \\
\vdash J(x, v) \rightarrow [a := A][x' = v, v' = a, t' = 1 \& t \leq \varepsilon] J(x, v) \\
\vdash J(x, v) \rightarrow [a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v) \\
\vdash J(x, v) \rightarrow [? \neg \text{SB}][a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v) \\
\vdash J(x, v) \rightarrow [? \neg \text{SB}; a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v)
\end{array}
\end{align*}
\]
Example Proof: Safe Accelerating

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]

\[
\begin{align*}
&[=] J(x, v) \vdash \neg \text{SB} \rightarrow \forall t \geq 0 (t \leq \varepsilon \rightarrow [x := \frac{A}{2}t^2 + vt + x] J(x, v)) \\
&['] J(x, v) \vdash \neg \text{SB} \rightarrow [x' = v, v' = A, t' = 1 \& t \leq \varepsilon] J(x, v) \\
&[=] J(x, v) \vdash \neg \text{SB} \rightarrow [a := A][x' = v, v' = a, t' = 1 \& t \leq \varepsilon] J(x, v) \\
&[;] J(x, v) \vdash \neg \text{SB} \rightarrow [a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v) \\
&[?] J(x, v) \vdash [\neg \text{SB}; a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v) \\
&[;] J(x, v) \vdash [\neg \text{SB}; a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v)
\end{align*}
\]
Example Proof: Safe Accelerating

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]

\[
\begin{align*}
J(x, v) &\vdash \neg SB \rightarrow \forall t \geq 0 (t \leq \varepsilon \rightarrow J(\frac{A}{2} t^2 + vt + x, At + v)) \\
[=] &\vdash J(x, v) \vdash \neg SB \rightarrow [x := \frac{A}{2} t^2 + vt + x] J(x, v) \\
[\text{[\text{\textquoteleft\text{\textquoteright}}]} &\vdash J(x, v) \vdash \neg SB \rightarrow [x' = v, v' = A, t' = 1 \& t \leq \varepsilon] J(x, v) \\
[=] &\vdash J(x, v) \vdash \neg SB \rightarrow [a := A][x' = v, v' = a, t' = 1 \& t \leq \varepsilon] J(x, v) \\
[\text{[\text{\textquoteleft\text{\textquoteright}}]} &\vdash J(x, v) \vdash [?\neg SB][a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v) \\
[?] &\vdash J(x, v) \vdash [\neg SB; a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v)
\end{align*}
\]
Example Proof: Safe Accelerating

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]

\[ \text{QE} \quad J(x, v) \vdash \neg \text{SB} \rightarrow \forall t \geq 0 \ (t \leq \varepsilon \rightarrow (At + v)^2 \leq 2b(m - \frac{A}{2} t^2 - vt - x)) \]
\[ J(x, v) \vdash \neg \text{SB} \rightarrow \forall t \geq 0 \ (t \leq \varepsilon \rightarrow J(\frac{A}{2} t^2 + vt + x, At + v)) \]
\[ [::] \quad J(x, v) \vdash \neg \text{SB} \rightarrow \forall t \geq 0 \ (t \leq \varepsilon \rightarrow [x := \frac{A}{2} t^2 + vt + x] J(x, v)) \]
\[ ['] \quad J(x, v) \vdash \neg \text{SB} \rightarrow [x' = v, v' = A, t' = 1 \& t \leq \varepsilon] J(x, v) \]
\[ [::] \quad J(x, v) \vdash \neg \text{SB} \rightarrow [a := A][x' = v, v' = a, t' = 1 \& t \leq \varepsilon] J(x, v) \]
\[ [:] \quad J(x, v) \vdash \neg \text{SB} \rightarrow [a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v) \]
\[ [?] \quad J(x, v) \vdash [? \neg \text{SB}][a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v) \]
\[ [:] \quad J(x, v) \vdash [? \neg \text{SB}; a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v) \]
\[
J(x, v) \equiv v^2 \leq 2b(m - x)
\]

\[
J(x, v) \models \neg SB \rightarrow (A\varepsilon + v)^2 \leq 2b(m - \frac{A}{2}\varepsilon^2 - v\varepsilon - x)
\]

\[
Q.E.
\]

\[
J(x, v) \models \neg SB \rightarrow \forall t \geq 0 \left( t \leq \varepsilon \rightarrow (At + v)^2 \leq 2b(m - \frac{A}{2}t^2 - vt - x) \right)
\]

\[
J(x, v) \models \neg SB \rightarrow \forall t \geq 0 \left( t \leq \varepsilon \rightarrow J \left( \frac{A}{2}t^2 + vt + x, At + v \right) \right)
\]

\[
[:=] J(x, v) \models \neg SB \rightarrow \forall t \geq 0 \left( t \leq \varepsilon \rightarrow [x := \frac{A}{2}t^2 + vt + x] J(x, v) \right)
\]

\[
[\prime] J(x, v) \models \neg SB \rightarrow [x' = v, v' = A, t' = 1 \& t \leq \varepsilon] J(x, v)
\]

\[
[:=] J(x, v) \models \neg SB \rightarrow [a := A][x' = v, v' = a, t' = 1 \& t \leq \varepsilon] J(x, v)
\]

\[
[\prime] J(x, v) \models \neg SB \rightarrow [a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v)
\]

\[
[?] J(x, v) \models [\neg SB][a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v)
\]

\[
[\prime] J(x, v) \models [\neg SB; a := A; (x' = v, v' = a, t' = 1 \& t \leq \varepsilon)] J(x, v)
\]

Nathan Fulton, Stefan Mitsch, André Platzer
KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
FM’16 20 / 41
\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]
\[ SB \equiv 2b(m - x) < v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v) \]
\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]
\[ SB \equiv 2b(m - x) < v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v) \]

\[ \text{ind} J(x, v) \vdash [(a := -b \cup \neg SB; a := A); x'' = a, t' = 1 & t \leq \varepsilon]^* J(x, v) \]
Example Proof: Safe Driving

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]
\[ SB \equiv 2b(m - x) < v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v) \]

\[
\begin{align*}
\text{[;]} & \quad J(x, v) \vdash [(a := -b \cup \neg SB; a := A); x'' = a, t' = 1 & t \leq \varepsilon] J(x, v) \\
\text{ind} & \quad J(x, v) \vdash [((a := -b \cup \neg SB; a := A); x'' = a, t' = 1 & t \leq \varepsilon)]^* J(x, v)
\end{align*}
\]
Example Proof: Safe Driving

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]
\[ SB \equiv 2b(m - x) < v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v) \]

\[
\begin{align*}
[\cup] & \quad J(x, v) \vdash [a := -b \cup ?\neg SB; a := A][x'' = a, \ t' = 1 & t \leq \varepsilon] J(x, v) \\
[\downarrow] & \quad J(x, v) \vdash [(a := -b \cup ?\neg SB; a := A); x'' = a, \ t' = 1 & t \leq \varepsilon] J(x, v) \\
\text{ind} & \quad J(x, v) \vdash [((a := -b \cup ?\neg SB; a := A); x'' = a, \ t' = 1 & t \leq \varepsilon)^*] J(x, v)
\end{align*}
\]
$J(x, v) \equiv v^2 \leq 2b(m - x)$

$SB \equiv 2b(m - x) < v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v)$

\[
J(x, v) \vdash [a := -b][x'' = a \ldots]J(x, v) \land [\neg SB; a := A][x'' = a \ldots]J(x, v)
\]

\[
J(x, v) \vdash [a := -b \lor \neg SB; a := A][x'' = a, t' = 1 \land t \leq \varepsilon]J(x, v)
\]

\[
[a := -b \lor \neg SB; a := A]; x'' = a, t' = 1 \land t \leq \varepsilon]J(x, v)
\]

\[
\text{ind}\ J(x, v) \vdash [((a := -b \lor \neg SB; a := A); x'' = a, t' = 1 \land t \leq \varepsilon)^*]J(x, v)
\]
Example Proof: Safe Driving

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]

\[ SB \equiv 2b(m - x) < v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v) \]

\[
\begin{align*}
J(x, v) \vdash [a := -b][x'' = a \ldots]J(x, v) & \land [\neg SB; a := A][x'' = a \ldots]J(x, v) \\
J(x, v) \vdash [a := -b \cup \neg SB; a := A][x'' = a, t' = 1 & t \leq \varepsilon]J(x, v) \\
J(x, v) \vdash [(a := -b \cup \neg SB; a := A); x'' = a, t' = 1 & t \leq \varepsilon]J(x, v) \\
\text{ind} J(x, v) \vdash [((a := -b \cup \neg SB; a := A); x'' = a, t' = 1 & t \leq \varepsilon)^*]J(x, v)
\end{align*}
\]
Example Proof: Safe Driving

\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]
\[ SB \equiv 2b(m - x) < v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v) \]

1. Proof is deterministic “follow your nose”.
2. Synthesize invariant \( J(x, v) \) and parameter constraint \( SB \).
3. \( J(x, v) \) is a predicate symbol to prove only once and instantiate later.
4. First looking at proofs of smaller pieces is often effective.
Differential Dynamic Logic: Axiomatization

\[
\begin{align*}
[\mathit{:=}] & \quad [x := e]P(x) \leftrightarrow P(e) \\
[?] & \quad [?Q]P \leftrightarrow (Q \rightarrow P) \\
[\prime] & \quad [x' = f(x)]P \leftrightarrow \forall t \geq 0 [x := y(t)]P \quad (y'(t) = f(y)) \\
[\cup] & \quad [\alpha \cup \beta]P \leftrightarrow [\alpha]P \land [\beta]P \\
[;] & \quad [\alpha; \beta]P \leftrightarrow [\alpha][\beta]P \\
[\ast] & \quad [\alpha^*]P \leftrightarrow P \land [\alpha][\alpha^*]P \\
K & \quad [\alpha](P \rightarrow Q) \rightarrow ([\alpha]P \rightarrow [\alpha]Q) \\
I & \quad [\alpha^*](P \rightarrow [\alpha]P) \rightarrow (P \rightarrow [\alpha^*]P) \\
C & \quad [\alpha^*] \forall v > 0 (P(v) \rightarrow \langle \alpha \rangle P(v-1)) \rightarrow \forall v (P(v) \rightarrow \langle \alpha^* \rangle \exists v \leq 0 P(v))
\end{align*}
\]
Differential Invariants for Differential Equations

Differential Invariant

\[ \frac{dx}{dt} = f(x) \]

\[ \frac{dy}{dt} = g(x, y) \]

\[ \text{inv} \]

Differential Cut

\[ DI \leq, \land, \lor \]

\[ DI >, \land, \lor \]

\[ DI \geq, =, \land, \lor \]

\[ DI =, \land, \lor \]

Differential Ghost

\[ x' = f(x) \]

Logic

Provability theory

Math

Characteristic PDE

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, JAR’16

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems

FM’16 22 / 41
Differential Invariants for Differential Equations

Differential Invariant

\[ x' = f(x) \]

Differential Cut

Differential Ghost

Logic

Provability theory

Math

Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, JAR'16

Nathan Fulton, Stefan Mitsch, André Platzer
Differential Invariants for Differential Equations

\[ x' = f(x) \]

\[ y' = g(x, y) \]

Logic Provability theory
Math Characteristic PDE

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, JAR’16
Differential Invariants for Differential Equations

\[ \frac{dx}{dt} = f(x) \]

\[ \frac{dy}{dt} = g(x, y) \]

**Logic**

Provability theory

**Math**

Characteristic PDE

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, JAR’16
Differential Invariants for Differential Equations

\[ \frac{dx}{dt} = f(x) \]

\[ \frac{dy}{dt} = g(x, y) \]

\[ \text{DI} \]

\[ \text{DI}_\geq \]

\[ \text{DI}_= \]

\[ \text{DI}_> \]

\[ \text{DI}_{\geq} \]

\[ \text{DI}_{=} \]

\[ \text{DI}_{>} \]

\[ \text{DI}_{\geq} \]

\[ \text{DI}_{=} \]

\[ \text{DI}_{>} \]

\[ \text{Logic} \]

\[ \text{Provability theory} \]

\[ \text{Math} \]

\[ \text{Characteristic PDE} \]

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, JAR’16

Nathan Fulton, Stefan Mitsch, André Platzer  KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems  FM’16  22 / 41
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

\[ x' = f(x) \]

$0 \quad t$

Logic
Provability theory

Math
Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, JAR'16

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Differential Invariants for Differential Equations

Differential Invariant 

Differential Cut 

Differential Ghost 

0 

x

\[ x' = f(x) \]

Logic 

Provability theory 

Math 

Characteristic PDE 

JLogComput'10, CAV'08, FMSD'09, LMCS’12, LICS’12, ITP’12, JAR’16
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

Logic
Provability theory

Math
Characteristic PDE

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, JAR’16
Differential Invariants for Differential Equations

Differential Invariant

\[ \frac{dy}{dt} = g(x,y) \]

Differential Cut

\[ \frac{dx}{dt} = f(x) \]

Differential Ghost

\[ y' = g(x,y), \quad x' = f(x) \]

Logic

Provability theory

Math

Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, JAR'16

Nathan Fulton, Stefan Mitsch, André Platzer KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

Logic
Provability theory

Math
Characteristic PDE

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, JAR’16

Nathan Fulton, Stefan Mitsch, André Platzer
Differential Invariant

\[ Q \vdash [x' := f(x)](F)' \]
\[ F \vdash [x' = f(x) \& Q]F \]
Differential Invariants for Differential Equations

**Differential Invariant**

\[ Q \vdash [x' := f(x)](F)' \]

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

**Differential Cut**

\[ F \vdash [x' = f(x) \& Q]C \quad F \vdash [x' = f(x) \& Q \land C]F \]

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

---

JLogComput'10, LMCS'12, LICS'12, JAR'16

Nathan Fulton, Stefan Mitsch, André Platzer KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Differential Invariants for Differential Equations

**Differential Invariant**

\[
\frac{Q \vdash [x' := f(x)](F)'}{F \vdash [x' = f(x) & Q]F}
\]

**Differential Cut**

\[
\frac{F \vdash [x' = f(x) & Q]C \quad F \vdash [x' = f(x) & Q \land C]F}{F \vdash [x' = f(x) & Q]F}
\]

**Differential Ghost**

\[
F \iff \exists y \ G \quad G \vdash [x' = f(x), y' = g(x, y) & Q]G
\]

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

JLogComput’10, LMCS’12, LICS’12, JAR’16
Differential Invariants for Differential Equations

Differential Invariant

\[
\frac{Q \vdash [x' := f(x)](F)'}{F \vdash [x' = f(x) & Q]F}
\]

Differential Cut

\[
\frac{F \vdash [x' = f(x) & Q]C \quad F \vdash [x' = f(x) & Q \land C]F}{F \vdash [x' = f(x) & Q]F}
\]

Differential Ghost

\[
F \leftrightarrow \exists y \ G \quad G \vdash [x' = f(x), y' = g(x, y) & Q]G
\]

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

if new \( y' = g(x, y) \) has a global solution

JLogComput’10, LMCS’12, LICS’12, JAR’16
\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y & \omega \geq 0 \land d \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]
\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y \land \omega \geq 0 \land d \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]
\[
\omega \geq 0 \land d \geq 0 \vdash [x' := y] [y' := -\omega^2 x - 2d \omega y] 2\omega^2 xx' + 2yy' \leq 0
\]
\[
\omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d \omega y \land \omega \geq 0 \land d \geq 0] \omega^2 x^2 + y^2 \leq c^2
\]
\[ \omega \geq 0 \land d \geq 0 \vdash 2 \omega^2 xy + 2y(- \omega^2 x - 2d \omega y) \leq 0 \]
\[ \omega \geq 0 \land d \geq 0 \vdash [x' := y][y' := -\omega^2 x - 2d \omega y] 2\omega^2 xx' + 2yy' \leq 0 \]
\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d \omega y \land \omega \geq 0 \land d \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]
\[ \omega \geq 0 \land d \geq 0 \vdash 2 \omega^2 xy + 2y(-\omega^2 x - 2d\omega y) \leq 0 \]
\[ \omega \geq 0 \land d \geq 0 \vdash [x' := y][y' := -\omega^2 x - 2d\omega y] 2\omega^2 xx' + 2yy' \leq 0 \]
\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y \land \omega \geq 0 \land d \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]
Differential Invariants for Differential Equations

\[ \omega \geq 0 \wedge d \geq 0 \vdash 2\omega^2 x y + 2y(-\omega^2 x - 2d\omega y) \leq 0 \]

\[ \omega \geq 0 \wedge d \geq 0 \vdash [x' := y][y' := -\omega^2 x - 2d\omega y] 2\omega^2 xx' + 2yy' \leq 0 \]

\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y \& \omega \geq 0 \wedge d \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]

damped oscillator
\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d \omega y, d' = 7 \& \omega \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]
\[
\omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d \omega y, d' = 7 \land \omega \geq 0] \omega^2 x^2 + y^2 \leq c^2
\]
\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d \omega y, d' = 7 \land \omega \geq 0 \land d \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]

increasingly damped oscillator
\[
\begin{align*}
\omega^2 x^2 + y^2 \leq c^2 &\vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 \land \omega \geq 0 \land \omega \geq 0] \omega^2 x^2 + y^2 \leq c^2 \\
\omega^2 x^2 + y^2 \leq c^2 &\vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 \land \omega \geq 0] \omega^2 x^2 + y^2 \leq c^2 \\
\end{align*}
\]

\[d \geq 0 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 \land \omega \geq 0] d \geq 0\]

increasingly damped oscillator
\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 & \omega \geq 0 \land d \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]

\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 & \omega \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]

\[
\omega \geq 0 \vdash [d' := 7] d' \geq 0
\]

\[
d \geq 0 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 & \omega \geq 0] d \geq 0
\]

increasingly damped oscillator
\[
\begin{align*}
\omega^2 x^2 + y^2 \leq c^2 &\implies [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 &\omega \geq 0 \land d \geq 0] \omega^2 x^2 + y^2 \leq c^2 \\
\omega^2 x^2 + y^2 \leq c^2 &\implies [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 &\omega \geq 0] \omega^2 x^2 + y^2 \leq c^2
\end{align*}
\]

\[
\begin{align*}
\omega \geq 0 &\implies 7 \geq 0 \\
\omega \geq 0 &\implies [d' := 7] d' \geq 0 \\
d \geq 0 &\implies [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 &\omega \geq 0] d \geq 0
\end{align*}
\]
\[
\begin{align*}
\omega^2 x^2 + y^2 &\leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 &\land \omega \geq 0 \land d \geq 0] \omega^2 x^2 + y^2 &\leq c^2 \\
\omega^2 x^2 + y^2 &\leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 &\land \omega \geq 0] \omega^2 x^2 + y^2 &\leq c^2 \\
\end{align*}
\]

\[\begin{array}{c}
\omega \geq 0 \vdash 7 \geq 0 \\
\omega \geq 0 \vdash [d' := 7] d' \geq 0 \\
d \geq 0 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 &\land \omega \geq 0] d' \geq 0
\end{array}\]

increasingly damped oscillator
\[ \omega \geq 0 \land d \geq 0 \vdash [x' := y][y' := -\omega^2 x - 2d \omega y] \ 2\omega^2 xx' + 2yy' \leq 0 \]

\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d \omega y, d' = 7 & \omega \geq 0 \land d \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]

\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d \omega y, d' = 7 & \omega \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]

\[ * \]

\[ \omega \geq 0 \vdash 7 \geq 0 \]

\[ \omega \geq 0 \vdash [d' := 7] \ d' \geq 0 \]

\[ d \geq 0 \vdash [x' = y, y' = -\omega^2 x - 2d \omega y, d' = 7 & \omega \geq 0] \ d \geq 0 \]

increasingly damped oscillator
\[ \omega \geq 0 \land d \geq 0 \vdash 2\omega^2 xy + 2y(-\omega^2 x - 2d\omega y) \leq 0 \]
\[ \omega \geq 0 \land d \geq 0 \vdash [x' := y][y' := -\omega^2 x - 2d\omega y] 2\omega^2 xx' + 2y y' \leq 0 \]
\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 \land \omega \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]
\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 \land \omega \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]

\[ \omega \geq 0 \vdash 7 \geq 0 \]
\[ \omega \geq 0 \vdash [d' := 7] d' \geq 0 \]
\[ d \geq 0 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 \land \omega \geq 0] d \geq 0 \]

increasingly damped oscillator
\[ \omega \geq 0 \land d \geq 0 \vdash 2\omega^2 xy + 2y(-\omega^2 x - 2d\omega y) \leq 0 \]

\[ \omega \geq 0 \land d \geq 0 \vdash [x' := y][y' := -\omega^2 x - 2d\omega y] 2\omega^2 xx' + 2yy' \leq 0 \]

\[ \omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 \land \omega \geq 0] \omega^2 x^2 + y^2 \leq c^2 \]

\[ \omega \geq 0 \vdash 7 \geq 0 \]

\[ \omega \geq 0 \vdash [d' := 7] d' \geq 0 \]

\[ d \geq 0 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 \land \omega \geq 0] d \geq 0 \]

increasingly damped oscillator
Differential Cuts for Differential Equations

\[
\omega \geq 0 \land d \geq 0 \vdash 2\omega^2 xy + 2y(-\omega^2 x - 2d\omega y) \leq 0
\]

\[
\omega \geq 0 \land d \geq 0 \vdash [x' := y][y' := -\omega^2 x - 2d\omega y] 2\omega^2 xx' + 2yy' \leq 0
\]

\[
\omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 \land \omega \geq 0] \omega^2 x^2 + y^2 \leq c^2
\]

\[
\omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 \land \omega \geq 0] \omega^2 x^2 + y^2 \leq c^2
\]

\[
\omega \geq 0 \vdash 7 \geq 0
\]

\[
\omega \geq 0 \vdash [d' := 7] d' \geq 0
\]

\[
d \geq 0 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y, d' = 7 \land \omega \geq 0] d \geq 0
\]

Could repeatedly diffcut in formulas to help the proof
Motion in 2D Plane: Don’t Collide with Obstacles
Motion in 2D Plane: Don’t Collide with Obstacles

\[ p = (x, y) \]
Motion in 2D Plane: Don’t Collide with Obstacles

$c = (x, y)$

$p = (x, y)$
Motion in 2D Plane: Don’t Collide with Obstacles

\[ p = (x, y) \]

How to always get such motion collision-free?

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Motion in 2D Plane: Don’t Collide with Obstacles

$c = (x, y)$

$p = (x, y)$

How to always get such motion collision-free?

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Motion in 2D Plane: Don’t Collide with Obstacles

$c = (x, y)$

How to always get such motion collision-free?

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Motion in 2D Plane: Don’t Collide with Obstacles

$p = (x, y)$

Nathan Fulton, Stefan Mitsch, André Platzer
KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Motion in 2D Plane: Don’t Collide with Obstacles

$c = (x, y)$

How to always get such motion collision-free?

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Motion in 2D Plane: Don’t Collide with Obstacles

\[ p = (x, y) \]

How to always get such motion collision-free?
Motion in 2D Plane: Don’t Collide with Obstacles

How to always get such motion collision-free?

$p = (x, y)$

$c$

$r$
2D Planar Car Model

**State**

- Position $p$
- Curve center $c$
- Curve radius $r$

Differential Axiomatization

New variables for (undecidable) transcendental functions:

$$\begin{align*}
d & x = \cos(\theta), \\
d & y = \sin(\theta), \\
d' & x = \cos(\theta)', \\
d' & y = -\sin(\theta) \\
\theta' &= -\omega \\
d & y = \tilde{p} \text{ after time } \varepsilon \\
\end{align*}$$
2D Planar Car Model

### State
- Position \( p \)
- Curve center \( c \)
- Curve radius \( r \)
- Orientation \( d \)

\[
\begin{align*}
\text{Translational velocity} \quad x' &= v \\
\text{Rotational velocity} \quad \theta' &= \omega
\end{align*}
\]

\[
\begin{align*}
dx &= \cos(\theta), \\
dy &= \sin(\theta)
\end{align*}
\]

\[
\begin{align*}
dx' &= \cos(\theta') = -\sin(\theta), \\
\theta' &= -\omega
\end{align*}
\]

\[
\tilde{p} \text{ after time } \varepsilon \text{ trajectory (length } v \varepsilon)\]
2D Planar Car Model

State
- Position $p$
- Curve center $c$
- Curve radius $r$
- Orientation $d$

Differential Axiomatization
- New variables for (undecidable) transcendental functions
  \[ d_x = \cos(\theta), \quad d_y = \sin(\theta) \]

\[ \sin \theta = d_y, \quad d_x = \cos \theta \]
2D Planar Car Model

State
- Position \( p \)
- Curve center \( c \)
- Curve radius \( r \)
- Orientation \( d \)
- Translational velocity \( x' = v \)
- Rotational velocity \( \theta' = \omega \)
- Control cycle duration \( \varepsilon \)

Differential Axiomatization
- New variables for (undecidable) transcendental functions

\[
\begin{align*}
  d_x &= \cos(\theta), \\
  d_y &= \sin(\theta)
\end{align*}
\]

\( \tilde{p} \) after time \( \varepsilon \)
trajectory (length \( v\varepsilon \))
2D Planar Car Model

State
- Position $p$
- Curve center $c$
- Curve radius $r$
- Orientation $d$
- Translational velocity $x' = v$
- Rotational velocity $\theta' = \omega$
- Control cycle duration $\varepsilon$

Differential Axiomatization
- New variables for (undecidable) transcendental functions

$$d_x = \cos(\theta), \quad d_y = \sin(\theta)$$

$$d'_x = \cos(\theta)' = - \sin(\theta)\theta' = -\omega d_y$$

$\tilde{p}$ after time $\varepsilon$

trajectory (length $v\varepsilon$)
Example: 2D Car with Brake, Accelerate, Steer

\[
t := 0; \\
\{ p' = vd, \ v' = a, \ d' = \omega d^\perp, \ \omega' = \frac{a}{r}, \ t' = 1 & v \geq 0 \land t \leq \varepsilon \}
\]
Example: 2D Car with Brake, Accelerate, Steer

\[
(a := -b \\
\cup a := A; \omega := *; r := *; \ ?r \neq 0 \land r \omega = v; m := *; \ ?Q); \\
t := 0; \\
\{p' = vd, v' = a, \ d' = \omega d^\perp, \ \omega' = \frac{a}{r}, \ t' = 1 \land v \geq 0 \land t \leq \varepsilon\}
\]
Example: 2D Car with Brake, Accelerate, Steer

\[
\begin{align*}
    & ( a := -b \\
    & \cup a := A; \omega := \ast; r := \ast; ?r \neq 0 \land r\omega = v; m := \ast; ?Q) \\
    & t := 0; \\
    & \{ p' = vd, v' = a, d' = \omega d^\perp, \omega' = \frac{a}{r}, t' = 1 \land v \geq 0 \land t \leq \varepsilon \}^*
\end{align*}
\]
Example: 2D Car with Brake, Accelerate, Steer

\[ Q \equiv 2b\|p - m\| \geq v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v) \]

\[
\left( \begin{array}{c}
  a := -b \\
  \cup a := A; \quad \omega := \ast; \quad r := \ast; \quad ?r \neq 0 \wedge r\omega = v; \quad m := \ast; \quad ?Q) \\
  t := 0; \\
  \{ p' = vd, \quad v' = a, \quad d' = \omega d^\perp, \quad \omega' = \frac{a}{r}, \quad t' = 1 \& v \geq 0 \wedge t \leq \varepsilon \} \right)^* 
\]
Intuition for Car’s Differential Invariants

Differential Invariants:

- Time goes forward: $t \geq 0$
Differential Invariants:

- Time goes forward: $t \geq 0$
- Velocity follows acceleration: $v = \text{old}(v) + at$
Differential Invariants:

- Time goes forward: $t \geq 0$
- Velocity follows acceleration: $v = \text{old}(v) + at$
- Stay on the circle: $\|d\| = 1$
Differential Invariants:
- Time goes forward: $t \geq 0$
- Velocity follows acceleration: $v = \text{old}(v) + at$
- Stay on the circle: $\|d\| = 1$
Differential Invariants:

- Time goes forward: \( t \geq 0 \)
- Velocity follows acceleration: \( v = \text{old}(v) + at \)
- Stay on the circle: \( \|d\| = 1 \)
- Stay close to position: \( \|p - \text{old}(p)\| \leq vt - \frac{a}{2}t^2 \)
Intuition for Car’s Differential Invariants

Differential Invariants:
- Time goes forward: $t \geq 0$
- Velocity follows acceleration: $v = \text{old}(v) + at$
- Stay on the circle: $\|d\| = 1$
- Stay close to position: $\|p - \text{old}(p)\| \leq vt - \frac{a}{2}t^2$

\{ need both \}
Differential Invariants:

- Time goes forward: $t \geq 0$
- Velocity follows acceleration: $v = \text{old}(v) + at$
- Stay on the circle: $\|d\| = 1$
- Stay close to position: $\|p - \text{old}(p)\|_{\infty} \leq vt - \frac{a}{2}t^2$

Supremum norm overapproximates circle with box (easier arithmetic)

need both
Outline

1. KeYmaera X Overview
   - Tutorial Objectives

2. Differential Dynamic Logic for Hybrid Systems
   - Syntax: Notation for Verification Questions
   - Semantics: Meaning of the Syntax
   - Example: Car Control Design
   - Example: Branching Structure

3. Proofs for CPS
   - Compositional Proof Calculus
   - Example: Safe Car Control

4. Differential Invariants for Differential Equations
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots

5. Synthesize Monitors

6. Case Studies

7. Summary
Real CPS

Verification Results

safe

Proof

Reachability Analysis

...
Real CPS

Model $\alpha^*$

Abstract

Control $\alpha_{\text{ctrl}}$

\[ \nu := \nu + 1 \]

Plant $\alpha_{\text{plant}}$

\[ x' = \nu \]

Verification Results

Proof

Reachability Analysis

\[ \text{safe} \]

\[ \text{safe} \]
Formal Verification in CPS Development

Real CPS

Model

Challenge

Verification results about models only apply if CPS fits to the model

Verifiably correct runtime model validation

Plant $\alpha_{\text{plant}}$

$x' = \nu$

$sense$

$\nu := \nu + 1$

$act$

$\leadsto$

safe

Verification Results

Reachability Analysis

Nathan Fulton, Stefan Mitsch, André Platzer KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems FM’16 30 / 41
ModelPlex ensures that verification results about models apply to CPS implementations.
ModelPlex ensures that verification results about models apply to CPS implementations.

**Contributions**

- Verification results about models transfer to CPS when validating model compliance.
- Compliance with model is characterizable in logic.
- Compliance formula transformed by proof to executable monitor.
- Correct-by-construction provably correct runtime model validation.

Model adequate?  control safe?  until next cycle?
ModelPlex at Runtime

“Simplex for Models”
ModelPlex at Runtime

**Compliance Monitor** Checks CPS for compliance with model at runtime
- Model Monitor: model adequate?
- Controller Monitor: control safe?
- Prediction Monitor: until next cycle?

**Fallback** Safe action, executed when monitor is not satisfied (veto)

**Challenge** What conditions do the monitors need to check to be safe?
When are two states linked through a run of model $\alpha$?

In

$i - 1$  $\xrightarrow{\text{Model } \alpha}$  $i$
When are two states linked through a run of model $\alpha$?
When are two states linked through a run of model $\alpha$?

A prior state characterized by $x^-$

A posterior state characterized by $x^+$

Semantical: $(\omega, \nu) \in \llbracket \alpha \rrbracket$  
reachability relation of $\alpha$
When are two states linked through a run of model $\alpha$?

![Diagram showing two states linked by $\alpha$]

**Offline**

**Semantical:**

$$(\omega, \nu) \in [\alpha]$$

**Logical dL:**

$$(\omega, \nu) \models \langle \alpha \rangle (x = x^+)$$

**Lemma:**

exists a run of $\alpha$ to a state where $x = x^+$

---

RV’14, FMSD’16

Nathan Fulton, Stefan Mitsch, André Platzer
KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
When are two states linked through a run of model $\alpha$?

A prior state characterized by $x^-$

Model $\alpha$ to $\nu$

A posterior state characterized by $x^+$

Semantical: $(\omega, \nu) \in \lbrack \alpha \rbrack$

Logical $d\mathcal{L}$: $(\omega, \nu) \models \langle \alpha \rangle (x = x^+)$

Arithmetical: $(\omega, \nu) \models F(x^-, x^+)$

Offline

Lemma

d$\mathcal{L}$ proof

exists a run of $\alpha$ to a state where $x = x^+$

check at runtime (efficient)

RV’14, FMSD’16
When are two states linked through a run of model $\alpha$?

- A prior state characterized by $x^-$
- A posterior state characterized by $x^+$

**Offline**

**Semantical:** $(\omega, \nu) \in [\alpha]$

**Logical $d\mathcal{L}$:** $(\omega, \nu) \models \langle \alpha \rangle (x = x^+)$

**Arithmetical:** $(\omega, \nu) \models F(x^-, x^+)$

**Lemma**

- Exists a run of $\alpha$ to a state where $x = x^+$

**$d\mathcal{L}$ proof**

- Check at runtime (efficient)

---

RV'14, FMSD'16

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Logic reduces CPS safety to runtime monitor with offline proof.

\[ \text{dL proof} \quad A \rightarrow [\alpha]S \]

Semantical: \((\omega, \nu) \in \llbracket \alpha \rrbracket\)

\[ \uparrow \quad \text{Lemma} \]

Logical dL: \((\omega, \nu) \models \langle \alpha \rangle (x = x^+)\)

\[ \uparrow \quad \text{dL proof} \]

Arithmetical: \((\omega, \nu) \models F(x^-, x^+)\)

\[ \text{check at runtime (efficient)} \]
Logic reduces CPS safety to runtime monitor with offline proof

$d\mathcal{L}$ proof  \[ A \rightarrow [\alpha]S \]

Offline

Semantical:  \((\omega, \nu) \in [\alpha]\)  

Logical $d\mathcal{L}$:  \((\omega, \nu) \models \langle \alpha \rangle(x = x^+)\)  

Arithmetical:  \((\omega, \nu) \models F(x^-, x^+)\)  

check at runtime (efficient)
Logic reduces CPS safety to runtime monitor with offline proof

Semantical: \((\omega, \nu) \in [\alpha]\)  

Logical \(\mathcal{L}\): \((\omega, \nu) \models \langle \alpha \rangle (x = x^+)\)  

Arithmetical: \((\omega, \nu) \models F(x^-, x^+)\)

\(\omega \xrightarrow{\alpha} \nu\) in Offline  

Init \(\omega \in [A]\)  

Safe \(\nu \in [S]\)

\(\text{check at runtime (efficient)}\)

RV’14, FMSD’16

Nathan Fulton, Stefan Mitsch, André Platzer KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Logic reduces CPS safety to runtime monitor with offline proof

\[ A \to [\alpha]S \]

**Offline**

Init: \( \omega \in \llbracket A \rrbracket \)

Safe: \( \nu \in \llbracket S \rrbracket \)

**Semantical:**

\( (\omega, \nu) \in \llbracket \alpha \rrbracket \)

\( \uparrow \) Lemma

**Logical \( d\mathcal{L} \):**

\( (\omega, \nu) \models \langle \alpha \rangle (x = x^+) \)

\( \uparrow \) \( d\mathcal{L} \) proof

**Arithmetical:**

\( (\omega, \nu) \models F(x^-, x^+) \)

\( \leftarrow \) check at runtime (efficient)

RV'14, FMSD'16
Logic reduces **CPS safety** to runtime monitor with offline proof.

\[ A \rightarrow [\alpha]S \]

**Offline**

- **Init**: \( \omega \in [A] \)
- **Safe**: \( \nu \in [S] \)

**Semantical**: \( (\omega, \nu) \in [\alpha] \)

**Logical \( d\mathcal{L} \)**: \( (\omega, \nu) \models \langle \alpha \rangle (x = x^+) \)

**Arithmetical**: \( (\omega, \nu) \models F(x^-, x^+) \) - check at runtime (efficient)

---

**RV’14, FMSD’16**

Nathan Fulton, Stefan Mitsch, André Platzer KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Logic reduces CPS safety to runtime monitor with offline proof.

\[
\begin{align*}
\text{dL proof: } & A \rightarrow [\alpha]S \\
\text{Offline: } & \omega \in [A] \quad \text{Init} \\
\text{Safe: } & \nu \in [S] \\
\text{Semantical: } & (\omega, \nu) \in [\alpha] \\
\text{Logical dL: } & (\omega, \nu) \models \langle \alpha \rangle(x = x^+) \\
\text{Arithmetical: } & (\omega, \nu) \models F(x^-, x^+) \\
\end{align*}
\]

Lemma: \((\omega, \nu) \models \langle \alpha \rangle(x = x^+)\) can be checked at runtime (efficient)
Logic reduces CPS safety to runtime monitor with offline proof.

\[ \omega \vdash (\alpha) (x = x^+) \]

\[ (\omega, \nu) \models F(x^-, x^+) \]

Check at runtime (efficient)
Logic reduces CPS safety to runtime monitor with offline proof

\[
\begin{align*}
\text{dL proof} & : A \rightarrow [\alpha]S \\
\text{Offline} & \\
\text{Init} & : \omega \in [A] \\
\text{Safe} & : \nu \in [S] \\
\text{Semantical:} & \quad (\omega, \nu) \in [\alpha] \\
& \quad \uparrow \text{Lemma} \\
\text{Logical dL:} & \quad (\omega, \nu) \models \langle\alpha\rangle(x = x^+) \\
& \quad \uparrow \text{dL proof} \\
\text{Arithmetical:} & \quad (\omega, \nu) \models F(x^-, x^+) \\
\end{align*}
\]

\textit{check at runtime (efficient)}

RV'14, FMSD'16
Logic reduces CPS safety to runtime monitor with offline proof

Not initial state. Model repeats...

dL proof $A \rightarrow \{\alpha\}S$

Offline

Init $\omega \in \{A\}$  Safe $\nu \in \{S\}$

Semantical: $(\omega, \nu) \in \{\alpha\}$

Logical dL: $(\omega, \nu) \models \langle \alpha \rangle (x = x^+)$

Arithmetical: $(\omega, \nu) \models F(x^-, x^+)$

check at runtime (efficient)

RV’14, FMSD’16
Logic reduces CPS safety to runtime monitor with offline proof

$d\mathcal{L}$ proof: $A \rightarrow [\alpha^*]S$

Offline

Semantical: $(\omega, \nu) \in [\alpha^*]

Logical $d\mathcal{L}$: $(\omega, \nu) \models \langle \alpha^* \rangle (x = x^+)$

Arithmetical: $(\omega, \nu) \models F(x^-, x^+)$

Init $\omega \in [A]$

Model $\alpha^*$

Safe $\nu \in [S]$

$\bigcap$

$\equiv$

Lemma

check at runtime (efficient)

RV’14, FMSD’16
ModelPlex ensures that verification results about models apply to CPS implementations.
1. KeYmaera X Overview
   - Tutorial Objectives

2. Differential Dynamic Logic for Hybrid Systems
   - Syntax: Notation for Verification Questions
   - Semantics: Meaning of the Syntax
   - Example: Car Control Design
   - Example: Branching Structure

3. Proofs for CPS
   - Compositional Proof Calculus
   - Example: Safe Car Control

4. Differential Invariants for Differential Equations
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots

5. Synthesize Monitors

6. Case Studies

7. Summary
Verified CPS Applications

Nathan Fulton, Stefan Mitsch, André Platzer
KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Verified CPS Applications

FM'11, LMCS'12, ICCPS'12, ITSC'11, ITSC'13, IJCAR'12
Verified CPS Applications

HSCC'13, RSS'13, CADE'12

Nathan Fulton, Stefan Mitsch, André Platzer
KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
Airborne Collision Avoidance System ACAS X: Verify

- Developed by the FAA to replace current TCAS in aircraft
- Approximately optimizes Markov Decision Process on a grid
- Advisory from lookup tables with numerous 5D interpolation regions

1. Identified safe region for each advisory symbolically
2. Proved safety for hybrid systems flight model in KeYmaera X

TACAS'15, EMSOFT'15, STTT'16
ACAS X table comparison shows safe advisory in 97.7% of the 648,591,384,375 states compared (15,160,434,734 counterexamples).

ACAS X issues DNC advisory, which induces collision unless corrected.
Airborne Collision Avoidance System ACAS X: Refine

- Conservative, so too many counterexamples
- Settle for: safe for a little while, with safe future advisory possibility
- Safeable advisory: a subsequent advisory can safely avoid collision

1. Identified safeable region for each advisory symbolically
2. Proved safety for hybrid systems flight model in KeYmaera X
ACAS X table comparison shows safeable advisory in more of the 648,591,384,375 states compared (31 to 899 $10^6$ counterexamples).

ACAS X issues Maintain advisory instead of CL1500
ACAS X table comparison shows safeable advisory in more of the 648,591,384,375 states compared (31 to 899 $10^6$ counterexamples).

Safe Version: Action Issued = CL1500
Followed by Most Extreme Up/Down-sense Available

ACAS X issues Maintain advisory instead of CL1500
1 KeYmaera X Overview
   - Tutorial Objectives
2 Differential Dynamic Logic for Hybrid Systems
   - Syntax: Notation for Verification Questions
   - Semantics: Meaning of the Syntax
   - Example: Car Control Design
   - Example: Branching Structure
3 Proofs for CPS
   - Compositional Proof Calculus
   - Example: Safe Car Control
4 Differential Invariants for Differential Equations
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots
5 Synthesize Monitors
6 Case Studies
7 Summary
KeYmaera X Tool Architecture

KeYmaera X Web UI (JavaScript)
- Simplified Proof Tree View

User Interface
- Proof View
- Tactics
- Models
- Proof Log

REST-API
- Proof Tree Simplification
- Searching
- Execution
- Proof Storing

Scala-API
- Proof Tree
- Proof Strategies
- dL Tactics
- Combinators
- Wrappers for Kernel Primitives

HyDRA Server
- Tactical Prover
- manages
- uses
- executes
- executes tactics on tools/ CPU cores

Axiomatic Core
- Proof Certificates
- Uniform Substitution
- Bound Renaming
- Propositional Sequent Calculus with Skolemization

KeYmaera X Kernel (soundness-critical, Scala)
- Real Quantifier Elimination
- Differential Equation Solving

Nathan Fulton, Stefan Mitsch, André Platzer
KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
KeYmaera X Theorem Prover for Hybrid Systems

- Multi-dynamical systems
- Compositional
- Logic & proofs for CPS
- Small soundness core
- Proof by pointing
- Interactive proof clicking
- Tactical proof programming
- Proof search tactics
- Flexible + modular API

FCPS course
STTT’16 tutorial
Book

KeYmaera X Tutorial: Tactics & Proofs for Cyber-Physical Systems
André Platzer.
Logics of dynamical systems.
In LICS [17], pages 13–24.

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


Stefan Mitsch and André Platzer.
ModelPlex: Verified runtime validation of verified cyber-physical system models.
Special issue of selected papers from RV’14.

Nathan Fulton and André Platzer.
A logic of proofs for differential dynamic logic: Toward independently checkable proof certificates for dynamic logics.
doi:10.1145/2854065.2854078.

André Platzer.
Logic & proofs for cyber-physical systems.
doi:10.1007/978-3-319-40229-1_3.

André Platzer.
Differential dynamic logic for hybrid systems.

André Platzer.
Differential game logic.
André Platzer.
The complete proof theory of hybrid systems.
In LICS [17], pages 541–550.
doi:10.1109/LICS.2012.64.

André Platzer.
Differential-algebraic dynamic logic for differential-algebraic programs.

André Platzer and Edmund M. Clarke.
Computing differential invariants of hybrid systems as fixedpoints.
Special issue for selected papers from CAV’08.

André Platzer.
The structure of differential invariants and differential cut elimination.
André Platzer.
A differential operator approach to equational differential invariants.
doi:10.1007/978-3-642-32347-8_3.

Jean-Baptiste Jeannin, Khalil Ghorbal, Yanni Kouskoulas, Ryan Gardner, Aurora Schmidt, Erik Zawadzki, and André Platzer.
A formally verified hybrid system for the next-generation airborne collision avoidance system.
doi:10.1007/978-3-662-46681-0_2.

Jean-Baptiste Jeannin, Khalil Ghorbal, Yanni Kouskoulas, Ryan Gardner, Aurora Schmidt, Erik Zawadzki, and André Platzer.
Formal verification of ACAS X, an industrial airborne collision avoidance system.
