From Idea to Provably Safe Implementation
Modeling, Proving, Simulation, and Synthesis in KeYmaera X

Nathan Fulton Stefan Mitsch André Platzer

http://keymaeraX.org/
Computer Science Department
Carnegie Mellon University, Pittsburgh, PA
# 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
- Differential Invariants
- Example: Elementary Differential Invariants
- Example: Ground Robots

## 5 Synthesize Monitors

## 6 Case Studies

## 7 Summary
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
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots

5 Synthesize Monitors

6 Case Studies

7 Summary
Correctness Questions in Complex 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?
- Code reviews?

When are we done? How many test cases are enough? Did we cover all relevant tests?
## Benefits of Logical Foundations for CPS

### Proofs

<table>
<thead>
<tr>
<th>Safety</th>
<th>Formalize system properties: What is “Safe”? “Reach goal”?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Models</td>
<td>Formalize system models, clarify behavior</td>
</tr>
<tr>
<td>Assumptions</td>
<td>Make assumptions explicit rather than silently</td>
</tr>
<tr>
<td>Constraints</td>
<td>Reveal invariants, switching conditions, operating conditions</td>
</tr>
<tr>
<td>Design</td>
<td>Invariants guide safe controller design</td>
</tr>
<tr>
<td>Constructive</td>
<td>Construct system models along with their proofs</td>
</tr>
</tbody>
</table>

### Byproducts

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Determine design trade-offs &amp; feasibility early</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthesis</td>
<td>Turn high-level models into code &amp; correctness monitors</td>
</tr>
<tr>
<td>Certificate</td>
<td>Proofs as artifacts for certification</td>
</tr>
</tbody>
</table>

### Tools

| KeYmaera X   | aXiomatic Tactical Theorem Prover for CPS |

Nathan Fulton, Stefan Mitsch, André Platzer | KeYmaera X Tutorial: From Idea to Provably Safe Implementation | 3 / 40
KeYmaera X Tutorial: From Idea to Provably Safe Implementation
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Core</td>
<td>Increases trust, modularity, enables experimentation (1677)</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 that follow proof search strategies Constructive tactics that 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>
<tr>
<td>System</td>
<td>LOC Estimate</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Keymaera X</td>
<td>1,677</td>
</tr>
<tr>
<td>Keymaera</td>
<td>65,989</td>
</tr>
<tr>
<td>KeY</td>
<td>51,328</td>
</tr>
<tr>
<td>HOL Light</td>
<td>396</td>
</tr>
<tr>
<td>Isabelle/Pure</td>
<td>8,113</td>
</tr>
<tr>
<td>Nuprl</td>
<td>15,000 + 50,000</td>
</tr>
<tr>
<td>Coq</td>
<td>20,000</td>
</tr>
<tr>
<td>HSolver</td>
<td>20,000</td>
</tr>
<tr>
<td>Flow*</td>
<td>25,000</td>
</tr>
<tr>
<td>PHAVer</td>
<td>30,000</td>
</tr>
<tr>
<td>dReal</td>
<td>50,000 + millions</td>
</tr>
<tr>
<td>SpaceEx</td>
<td>100,000</td>
</tr>
<tr>
<td>HyCreate2</td>
<td>6,081 + user model analysis</td>
</tr>
</tbody>
</table>

Disclaimer: These self-reported estimates of the soundness-critical lines of code + rules are to be taken with a grain of salt. Different languages, capabilities, styles...
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)
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 combine multiple simple dynamical effects.

Tame Parts
Exploiting compositionality tames CPS complexity.
Learning Objectives

Use KeYmaera X to:

1. Model cyber-physical systems
2. Express safety/correctness properties
3. Find bugs in a system design
4. Simulate system models
5. Identify safety constraints
6. Identify system invariants
7. Verify the final system design
8. Write automated proof tactics
9. Prove differential equations
10. Synthesize correct runtime monitors
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
   - 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
   • Differential Invariants
   • Example: Elementary Differential Invariants
   • Example: Ground Robots

5 Synthesize Monitors

6 Case Studies

7 Summary
CPS Analysis

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

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

\[ \begin{align*}
  x &\neq m \\
  b &> 0
\end{align*} \]

\[ \text{init} \rightarrow [s_1 \cdot (\text{if} (\text{SB}(x, m) \land a := -b) ; x' = v, v' = a) \ast] x \neq m \]

\[ \text{post} \]

- Nathan Fulton, Stefan Mitsch, André Platzer
  KeYmaera X Tutorial: From Idea to Provably Safe Implementation
  10 / 40
CPS Analysis

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

\[ [\alpha] \varphi \Rightarrow \varphi \]

\[ x \neq m \]

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

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

\[ \text{post} \]

Nathan Fulton, Stefan Mitsch, André Platzer | KeYmaera X Tutorial: From Idea to Provably Safe Implementation
CPS Analysis

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

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

\[ \Box x \neq m \quad x \neq m \quad x \neq m \]

\[ \text{init} \rightarrow \left[ \left( \text{if } \left( \text{SB}(x, m) \right) \right) \right. \]

\[ a = -b \quad x' = v, \quad v' = a \left] \right. \]

\[ \varphi \]

\[ \alpha \]

\[ t \]

\[ x \]

\[ m \]

\[ \alpha \]

\[ \varphi \]

\[ \alpha \]

\[ \varphi \]

\[ \alpha \]

\[ \varphi \]
CPS Analysis

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

\[ [\alpha] \varphi \]

\[ x \not\equiv m \]

\[ \alpha \]

\[ \varphi \]

\[ x \not\equiv m \]

\[ x \not\equiv m \]

\[ x \not\equiv m \]

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

\[ \text{post} \]

[Diagram with graphs and variables]
Concept (Differential Dynamic Logic) (JAR’08, LICS’12)

\[ [\alpha] \varphi \] 

\[ x \neq m \]

\[ \alpha \]

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

\[ \text{ODE} \]
CPS Analysis

Concept (Differential Dynamic Logic)

\[ [\alpha] \varphi \]

\[ \alpha \]

\[ \varphi \]

\( x \neq m \)

\( x \neq m \)

\( x \neq m \)

\( x \neq m \)

\( a := -b \)

\( x' = v, v' = a \)

\( x \neq m \)

\( x \neq m \)

\( x \neq m \)

\( x \neq m \)

\( x' = v, v' = a \)

\( x \neq m \)

\( x \neq m \)

\( x \neq m \)

\( x \neq m \)

\( x' = v, v' = a \)

\( x \neq m \)

\( x \neq m \)

\( x \neq m \)

\( x \neq m \)
CPS Analysis

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

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

\( \text{test} \)
\( \text{assign} \)

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

\[ x \neq m \]

\[ [\alpha] x \neq m \]

ODE
CPS Analysis

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

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

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

seq. compose

test assign

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: From Idea to Provably Safe Implementation
Concept (Differential Dynamic Logic) (JAR’08,LICS’12)

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

### CPS Analysis

[Diagram and textual content regarding CPS Analysis, including expressions and logical formulas.]
CPS Analysis

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

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

\[ x \neq m \]

\[ [\alpha] x \neq m \]

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

![Graphical representation of all runs]

\[ \alpha \]

\[ \varphi \]

\[ \varphi \alpha \]
CPS Analysis

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

\[ [\alpha] \varphi \quad \text{init} \]

\[
x \neq m \land b > 0 \rightarrow \left[ \left( (\text{if}(SB(x, m)) \ a := -b) ; \ x' = v, v' = a \right)^* \right] x \neq m
\]

\[ [\alpha] x \neq m \quad \text{post} \]
CPS Analysis

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

\[ [\alpha] \varphi \quad \text{init} \]

\[ x \neq m \land b > 0 \rightarrow \left[ \left( (\neg \text{SB}(x, m) \cup a := -b) ; x' = v, v' = a \right)^* \right] x \neq m \quad \text{post} \]
CPS Analysis

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

\[
[\alpha] \varphi 
\]

\[
\text{test} \quad \text{nondet. choice}
\]

\[
x \neq m \land b > 0 \rightarrow \left[\left((\neg \mathsf{SB}(x, m) \cup a := -b) ; x' = v, v' = a\right)^*\right] x \neq m
\]

\[
\begin{align*}
&\text{init} \\
&\text{post}
\end{align*}
\]
CPS Analysis

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

\[ [\alpha] \varphi \]

\[ x \neq m \]

\[ x \neq m \]

\[ x \neq m \]

\[ \text{init} \]

\[ \text{post} \]

\[ x \neq m \wedge b > 0 \rightarrow [((? \neg SB(x, m) \cup a := -b) ; x' = v, v' = a)^*] x \neq m \]

hybrid program dynamics
Hybrid Programs vs. Hybrid Automata

\[
\begin{align*}
\text{far} &\equiv x' = v, \quad v' = A \land \neg \text{SB}(x, m) \\
\text{brk} &\equiv x' = v, \quad v' = -b \land \text{SB}(x, m) \lor \text{true} \\
\text{cls} &\equiv x' = v, \quad v' = \ldots \land \ldots \\
\text{fsa} &\equiv x' = 0, \quad v' = 0 \land v = 0
\end{align*}
\]
Hybrid Programs vs. Hybrid Automata

\[
\begin{align*}
\text{far} &\equiv x' = v, \quad v' = A \land \neg \text{SB}(x, m) \\
\text{brk} &\equiv x' = v, \quad v' = -b \land \text{SB}(x, m) \lor \text{true} \\
\text{cls} &\equiv x' = v, \quad v' = \ldots \land \ldots \\
\text{fsa} &\equiv x' = 0, \quad v' = 0 \land v = 0
\end{align*}
\]
Hybrid Programs vs. Hybrid Automata

Want: Compositional verification

\[
\begin{align*}
\text{far} & \equiv x' = v, v' = A \land \neg \text{SB}(x, m) \\
\text{brk} & \equiv x' = v, v' = -b \land \text{SB}(x, m) \lor \text{true} \\
\text{cls} & \equiv x' = v, v' = \ldots \land \ldots \\
\text{fsa} & \equiv x' = 0, v' = 0 \land v = 0
\end{align*}
\]
Want: Compositional verification

\[
\begin{align*}
\text{far} & \equiv x' = v, \ v' = A \land \neg \text{SB}(x, m) \\
\text{brk} & \equiv x' = v, \ v' = -b \land \text{SB}(x, m) \lor \text{true} \\
\text{cls} & \equiv x' = v, \ v' = \ldots \land \ldots \\
\text{fsa} & \equiv x' = 0, \ v' = 0 \land v = 0
\end{align*}
\]
Hybrid Programs vs. Hybrid Automata

Want: Compositional verification

\[
\begin{align*}
\text{far} & \equiv x' = v, \quad v' = A \land \neg \text{SB}(x, m) \\
\text{brk} & \equiv x' = v, \quad v' = -b \land \text{SB}(x, m) \lor \text{true} \\
\text{cls} & \equiv x' = v, \quad v' = \ldots \land \ldots \\
\text{fsa} & \equiv x' = 0, \quad v' = 0 \land v = 0
\end{align*}
\]
### Differential Dynamic Logic $dL$: Syntax

**Definition (Hybrid program $\alpha$)**

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

**Definition ($dL$ Formula $P$)**

\[
e \geq \bar{e} \mid \neg P \mid P \land Q \mid \forall x \ P \mid \exists x \ P \mid [\alpha] P \mid \langle \alpha \rangle P
\]
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 ($d\mathcal{L}$ Formula $P$)**

\[ e \geq \bar{e} \mid \neg P \mid P \land Q \mid \forall x \ P \mid \exists x \ P \mid [\alpha]P \mid \langle \alpha \rangle P \]

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

- All Reals
- Some Reals
- All Runs
- Some Runs
CPS Analysis

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

\[ \alpha \varphi \Rightarrow \varphi \]

\[ x \neq m \land b > 0 \rightarrow \left[ \left( (? SB(x, m) \cup a := -b) \ ; \ x' = v, v' = a \right)^* \right] x \neq m \]

hybrid program dynamics
Definition (Hybrid program semantics) 
\( \llbracket \cdot \rrbracket : \text{HP} \to \wp(\mathcal{S} \times \mathcal{S}) \)

- \( \llbracket x := e \rrbracket = \{ (\omega, \nu) : \nu = \omega \text{ except } \llbracket x \rrbracket \nu = \llbracket e \rrbracket \omega \} \)
- \( \llbracket ? Q \rrbracket = \{ (\omega, \omega) : \omega \in \llbracket Q \rrbracket \} \)
- \( \llbracket x' = f(x) \rrbracket = \{ (\varphi(0), \varphi(r)) : \varphi \models x' = f(x) \text{ for some duration } r \} \)
- \( \llbracket \alpha \cup \beta \rrbracket = \llbracket \alpha \rrbracket \cup \llbracket \beta \rrbracket \)
- \( \llbracket \alpha ; \beta \rrbracket = \llbracket \alpha \rrbracket \circ \llbracket \beta \rrbracket \)
- \( \llbracket \alpha^* \rrbracket = \bigcup_{n \in \mathbb{N}} \llbracket \alpha^n \rrbracket \)

Definition (d\(\mathcal{L}\) semantics) 
\( \llbracket \cdot \rrbracket : \text{Fml} \to \wp(\mathcal{S}) \)

- \( \llbracket e \geq \tilde{e} \rrbracket = \{ \omega : \llbracket e \rrbracket \omega \geq \llbracket \tilde{e} \rrbracket \omega \} \)
- \( \llbracket \neg P \rrbracket = \llbracket P \rrbracket^c \)
- \( \llbracket P \land Q \rrbracket = \llbracket P \rrbracket \cap \llbracket Q \rrbracket \)
- \( \llbracket \langle \alpha \rangle P \rrbracket = \llbracket \alpha \rrbracket \circ \llbracket P \rrbracket = \{ \omega : \nu \in \llbracket P \rrbracket \text{ for some } \nu : (\omega, \nu) \in \llbracket \alpha \rrbracket \} \)
- \( \llbracket [\alpha] P \rrbracket = \llbracket \neg \langle \alpha \rangle \neg P \rrbracket = \{ \omega : \nu \in \llbracket P \rrbracket \text{ for all } \nu : (\omega, \nu) \in \llbracket \alpha \rrbracket \} \)
- \( \llbracket \exists x P \rrbracket = \{ \omega : \omega_x^r \in \llbracket P \rrbracket \text{ for some } r \in \mathbb{R} \} \)
Differential Dynamic Logic \( d\mathcal{L} \): Semantics

\[
\omega \xrightarrow{x := e} \nu
\]

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

\[
\omega \xrightarrow{? \mathcal{Q}} \nu \quad \text{if} \quad \omega \in \llbracket \mathcal{Q} \rrbracket
\]

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

\[
\omega \quad \text{no change if} \quad \omega \in \llbracket \mathcal{Q} \rrbracket
\]

\[
\text{otherwise no transition}
\]
Differential Dynamic Logic $\text{dL}$: Semantics
Differential Dynamic Logic $\mathcal{DL}$: Semantics

\[ \omega \xrightarrow{\alpha} \nu_1 \xrightarrow{\beta} \nu_2 \]

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

\[ (\alpha; \beta)^* \]

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

\[ \nu_1 \]

\[ \nu_2 \]

\[ t \]

\[ \nu \]

\[ \omega \]

\[ \nu \]

Nathan Fulton, Stefan Mitsch, André Platzer  KeYmaera X Tutorial: From Idea to Provably Safe Implementation
Definition (\(d\mathcal{L}\) Formulas)

- \([\alpha]P\)
- \(\omega\)
- \(P\)

Compositional semantics \(\Rightarrow\) compositional proofs!
Definition (dL Formulas)

ω

⟨α⟩P

P

α-span

β-span

compositional semantics ⇒ compositional proofs!
Definition (d\(\mathcal{L}\) Formulas)

\[ [\alpha]^P \]

\(\omega\)

\(\alpha\)-span

compositional semantics \(\Rightarrow\) compositional proofs!
Definition (dL Formulas)

\[ [\alpha]P \]

\[ \langle \beta \rangle P \]

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

\[ \alpha\text{-span} \]
Definition (dL Formulas)
Definition (dŁ Formulas)

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

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

Control decision: accelerate or brake

Example (Single car $car_s$)

$\begin{align*}
(a := A \cup a := -b); \quad & x' = v, v' = a
\end{align*}$
Repeat control decisions

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

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

Example (Single car $c_{ars}$)

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

Velocity bound $v \geq 0$

Example (Single car $c$ars)

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

Example (Single car $car_s$)

\[ ((a := A \cup a := -b); \ x' = v, v' = a \& v \geq 0) \]

$\begin{array}{c|c|c|c|c|c|c|c}
 t & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
 a & -2 & -2 & 0 & 0 & 0 & 0 & 0 \\
 v & 6 & 6 & 6 & 6 & 6 & 6 & 6 \\
 \end{array}$
Accelerate condition \( ?Q \)

Example (Single car \( car_s \))

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

Example (Single car $c ar_s$)

$$(((?Q; a := 0) \cup a := -b); \ x' = v, v' = a \& v \geq 0)$$

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

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

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

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

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

![Diagram](image)
Ex: Car Control Properties

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

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

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

Example (Live, can move everywhere)

\[ \varepsilon > 0 \& A > 0 \& b > 0 \rightarrow \forall p \exists m \langle car_\varepsilon \rangle x \geq p \]
Robot ≡ (ctrl ; drive)*

\[
\begin{align*}
\text{ctrl} & \equiv (\exists m - x \leq SB; a := -b) \\
& \quad \cup (\exists m - x \geq SB; a := A)
\end{align*}
\]

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

\text{drive} \equiv t := 0 ; (x' = v, v' = a, t' = 1
\land v \geq 0 \land t \leq \varepsilon)
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})^*

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

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

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

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

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

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

Robot \equiv (ctrl ; 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' = \nu, \nu' = a, t' = 1 \\
& \text{& } \nu \geq 0 \land t \leq \varepsilon)
\end{align*}
Robot $\equiv (\text{ctrl} ; \text{drive})^*$

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

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

if(Q)α else β ≡ (?Q; α) ∪ (?¬Q; β)

while(Q)α ≡

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

while\(Q\) \(\alpha\) \(\equiv\) \((?Q; \alpha)^*; \neg Q\)

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

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

\(\cup\) (?\(m - x \geq SB\); \(a := A\))

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

\& \(v \geq 0 \land t \leq \varepsilon\))
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
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots

5. Synthesize Monitors

6. Case Studies

7. Summary
Differential Dynamic Logic: Axiomatization

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

\[ ?Q \models (Q \implies P) \]

\[ x' = f(x) \models \forall t \geq 0 \left[ x := y(t) \right] P \quad (y'(t) = f(y)) \]

\[ \cup \models [\alpha \cup \beta] P \iff [\alpha]P \land [\beta]P \]

\[ ; \models [\alpha; \beta] P \iff [\alpha][\beta]P \]

\[ ^* \models [\alpha^*]P \iff P \land [\alpha][\alpha^*]P \]

\[ K \models [\alpha](P \implies Q) \implies ([\alpha]P \implies [\alpha]Q) \]

\[ L \models [\alpha^*](P \implies [\alpha]P) \implies (P \implies [\alpha^*]P) \]

\[ C \models [\alpha^*]\forall v > 0 (P(v) \implies \langle\alpha\rangle P(v-1)) \implies \forall v (P(v) \implies \langle\alpha^*\rangle \exists v \leq 0 P(v)) \]
Proofs for Hybrid Systems

compositional semantics $\Rightarrow$ 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

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

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

\[ P \quad P \rightarrow [\alpha]P \]

\[ [\alpha^*]P \]

Nathan Fulton, Stefan Mitsch, André Platzer  KeYmaera X Tutorial: From Idea to Provably Safe Implementation
Example Proof: Safe Braking

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

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

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 (bottom) to premises with remaining subgoals (top) until no more subgoals (\( \ast \))
Example Proof: Safe Braking

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

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

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 (bottom) to premises with remaining subgoals (top) until no more subgoals (*)
Example Proof: Safe Braking

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

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 (bottom) to premises with remaining subgoals (top) until no more subgoals (*)

\[
\begin{align*}
\Gamma & \vdash [a := -b][x' = v, v' = a]J(x, v) \\
\Gamma & \vdash [x' = v, v' = -b]J(x, v) \\
\Gamma & \vdash [x' = v, v' = a]J(x, v)
\end{align*}
\]
Example Proof: Safe Braking

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

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

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

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

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

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 (bottom) to premises with remaining subgoals (top) until no more subgoals (\( \ast \))
Example Proof: Safe Braking

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

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

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

\( \frac{\vdash}{\vdash} \text{sequent} \)

\( \frac{\vdash}{\vdash} \text{antecedent} \)

\( \frac{\vdash}{\vdash} \text{succedent} \)
\( J(x, v) \equiv v^2 \leq 2b(m - x) \)

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

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

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

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

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

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

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

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

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

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

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

Nathan Fulton, Stefan Mitsch, André Platzer  KeYmaera X Tutorial: From Idea to Provably Safe Implementation 22 / 40
Example Proof: Safe Accelerating

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\[
\begin{align*}
J(x, v) &\vdash [a := -b][x'' = a \ldots]J(x, v) \land [?\neg \text{SB}; a := A][x'' = a \ldots]J(x, v) \\
\text{[U]} &
J(x, v) \vdash [a := -b \cup ?\neg \text{SB}; a := A][x'' = a, t' = 1 \& t \leq \varepsilon]J(x, v) \\
\text{[i]} &
J(x, v) \vdash [(a := -b \cup ?\neg \text{SB}; a := A); x'' = a, t' = 1 \& t \leq \varepsilon]J(x, v) \\
\text{loop} &
J(x, v) \vdash [((a := -b \cup ?\neg \text{SB}; a := A); x'' = a, t' = 1 \& t \leq \varepsilon)^*]J(x, v)
\end{align*}
\]
\[ J(x, v) \equiv v^2 \leq 2b(m - x) \]
\[ SB \equiv 2b(m - x) < v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v) \]

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

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

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

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

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

\[
\begin{align*}
\text{QE} & \quad \text{previous proofs for braking and acceleration} \\
J(x, v) \vdash [a := -b][x'' = a \ldots]J(x, v) & \land [\neg SB; a := A][x'' = a \ldots]J(x, v) \\
[\cup] & \quad J(x, v) \vdash [a := -b \cup \neg SB; a := A][x'' = a, t' = 1 & t \leq \varepsilon]J(x, v) \\
[\vdash] & \quad J(x, v) \vdash [(a := -b \cup \neg SB; a := A); x'' = a, t' = 1 & t \leq \varepsilon]J(x, v) \\
\text{loop} & \quad J(x, v) \vdash [((a := -b \cup \neg SB; a := A); x'' = a, t' = 1 & t \leq \varepsilon)^*]J(x, v)
\end{align*}
\]

1. Proof is essentially 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 pieces is often effective, but not necessary.
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
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots

5. Synthesize Monitors

6. Case Studies

7. Summary
Differential Dynamic Logic: Axiomatization

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

\[ ? Q ] P \leftrightarrow (Q \rightarrow P) \]

\[ x' = f(x) ] P \leftrightarrow \forall t \geq 0 [ x := y(t) ] P \quad (y'(t) = f(y)) \]

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

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

\[ [\alpha^* ] P \leftrightarrow P \land [\alpha ][\alpha^* ] P \]

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

I \[ [\alpha^* ](P \rightarrow [\alpha ] P) \rightarrow (P \rightarrow [\alpha^* ] P) \]

C \[ [\alpha^* ] \forall v > 0 (P(v) \rightarrow \langle \alpha \rangle P(v-1)) \rightarrow \forall v (P(v) \rightarrow \langle \alpha^* \rangle \exists v \leq 0 P(v)) \]

LICS'12, CADE'15
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, CADE’15

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: From Idea to Provably Safe Implementation
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, CADE’15

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: From Idea to Provably Safe Implementation

24 / 40
Differential Invariants for Differential Equations

\[ x' = f(x) \]

Logic
Provability theory
Math
Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15
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, CADE’15
Differential Invariants for Differential Equations

Differential Invariant

\[ x' = f(x) \]

Differential Cut

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

Differential Ghost

\[ \text{Logic} \]

Provability theory

\[ \text{Math} \]

Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15
Differential Invariants for Differential Equations

**Differential Invariant**

\[
\frac{dx}{dt} = f(x) \quad \Rightarrow \quad x = \int f(x) \, dt
\]

**Differential Cut**

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

**Differential Ghost**

\[
\text{Logic Provability theory}
\]

\[
\text{Math Characteristic PDE}
\]
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, CADE'15
Differential Invariants for Differential Equations

Differential Invariant

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

Differential Cut

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

Differential Ghost

Logic
Provability theory

Math
Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15

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

\[ \begin{align*}
    x' &= f(x) \\
    y' &= g(x, y)
\end{align*} \]

Logic

Provability theory

Math

Characteristic PDE

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, CADE’15
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, CADE'15

Nathan Fulton, Stefan Mitsch, André Platzer KeYmaera X Tutorial: From Idea to Provably Safe Implementation
Differential Invariant

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

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

Differential Cut

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

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

Differential Cut

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

Differential Ghost

\( F \leftrightarrow \exists y \ G \)
\[
\frac{G \vdash [x' = f(x), y' = g(x, y) \& Q]G}{F \vdash [x' = f(x) \& Q]F}
\]
**Differential Invariant**

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

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

**Differential Cut**

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

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

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

\[
\omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y \land (\omega \geq 0 \land d \geq 0)] \omega^2 x^2 + y^2 \leq c^2
\]
\[
\begin{align*}
\omega \geq 0 & \land d \geq 0 \vdash 2\omega^2 xy + 2y(-\omega^2 x - 2d\omega y) \leq 0 \\
\omega \geq 0 & \land d \geq 0 \vdash [x' := y][y' := -\omega^2 x - 2d\omega y]2\omega^2 xx' + 2yy' \leq 0 \\
\omega^2 x^2 + y^2 \leq c^2 & \vdash [x' = y, y' = -\omega^2 x - 2d\omega y \land (\omega \geq 0 \land d \geq 0)]\omega^2 x^2 + y^2 \leq c^2
\end{align*}
\]
\[
\begin{align*}
\omega \geq 0 \land d \geq 0 \vdash & \quad 2\omega^2 xy + 2y(-\omega^2 x - 2d\omega y) \leq 0 \\
\omega \geq 0 \land d \geq 0 \vdash & \quad [x' := y][y' := -\omega^2 x - 2d\omega y]2\omega^2 xx' + 2yy' \leq 0 \\
\omega^2 x^2 + y^2 \leq c^2 \vdash & \quad [x' = y, y' = -\omega^2 x - 2d\omega y \land (\omega \geq 0 \land d \geq 0)]\omega^2 x^2 + y^2 \leq c^2
\end{align*}
\]
\begin{align*}
\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
\end{align*}
\[ x^2 + x^3 - y^2 - c = 0 \rightarrow [x' = -2y, y' = -2x - 3x^2] \]
Differential Invariants for Differential Equations

\[
\begin{align*}
    x' &= 2x^4y + 4x^2y^3 - 6x^2y, \\
    y' &= -4x^3y^2 - 2xy^4 + 6xy^2
\end{align*}
\]

\[x^4y^2 + x^2y^4 - 3x^2y^2 \leq c\]
Motion in 2D Plane: Don’t Collide with Obstacles
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: From Idea to Provably Safe Implementation
Motion in 2D Plane: Don’t Collide with Obstacles

$c = (x, y)$

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

\[ p = (x, y) \]

\[ c \]

\[ r \]

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

$c = (x, y)$

$p = (x, y)$

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

$c = (x, y)$

$p = (x, y)$

How to always get such motion collision-free?
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
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

$c = (x, y)$

How to always get such motion collision-free?

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: From Idea to Provably Safe Implementation
Motion in 2D Plane: Don’t Collide with Obstacles

$c = (x, y)$

$p = (x, y)$

How to always get such motion collision-free?
2D Planar Car Model

State

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

\[
d_x = \cos(\theta), \quad d_y = \sin(\theta)
\]

\[
d_x' = \cos(\theta)', \quad d_y' = -\sin(\theta)
\]

\[
\theta' = -\omega
\]

\[\text{trajectory (length } v \text{ after time } \varepsilon)\]

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: From Idea to Provably Safe Implementation
2D Planar Car Model

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

Translational velocity $x' = v$
Rotational velocity $\theta' = \omega$

Differential Axiomatization

New variables for (undecidable) transcendental functions

\begin{align*}
dx &= \cos(\theta), \\
dy &= \sin(\theta) \\
d'x &= \cos(\theta') \\
d'y &= -\sin(\theta') \\
\theta' &= -\omega
\end{align*}

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
  
  \[
  d_x = \cos(\theta), \quad d_y = \sin(\theta)
  \]

---

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

$$dx = \cos(\theta), \quad dy = \sin(\theta)$$

- Trajectory (length $v\varepsilon$)

![Diagram with labels: p after time $\varepsilon$, trajectory (length $v\varepsilon$), c, p, d, r, $\omega\varepsilon$.]
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
  
  \[
  \begin{align*}
  d_x &= \cos(\theta), \quad d_y = \sin(\theta) \\
  d'_x &= \cos(\theta)' = -\sin(\theta)\theta' = -\omega d_y
  \end{align*}
  \]

$p$ 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

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

\[
\begin{align*}
\left( \begin{array}{l}
  a := -b \\
  \cup a := A; \quad \omega := *; \quad r := *; \quad \lnot r \neq 0 \land r \omega = v; \quad m := *; \quad ?Q \\
  t := 0; \\
  \{ p' = vd, \quad v' = a, \quad d' = \omega d^\perp, \quad \omega' = \frac{a}{r}, \quad t' = 1 \land v \geq 0 \land t \leq \varepsilon \} \right)^*
\end{array} \right)
\]
2D Car Model as Hybrid Program

Example: 2D Car with Brake, Accelerate, Steer

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

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

- Stay on the circle: $\|d\| = 1$
Differential Invariants:

- Stay on the circle: \( \| d \| = 1 \)
Intuition for Car’s Differential Invariants

Differential Invariants:
- Stay on the circle: $\|d\| = 1$
- Stay close to position: $\|p - \text{old}(p)\| \leq vt - \frac{a}{2}t^2$
Differential Invariants:

- Stay on the circle: $\|d\| = 1$
- Stay close to position: $\|p - old(p)\| \leq vt - \frac{a}{2} t^2$

$\{ \text{need both} \}$
Intuition for Car’s Differential Invariants

Differential Invariants:

- Stay on the circle: \( \|d\| = 1 \)
- Stay close to position: \( \|p - old(p)\|_{\infty} \leq vt - \frac{a}{2}t^2 \)

need both

supremum norm overapproximates circle with box (easier arithmetic)
Intuition for Car’s Differential Invariants

Differential Invariants:

- Stay on the circle: \(|d| = 1\)
- Stay close to position: \(||p - old(p)||_\infty \leq vt - \frac{a}{2}t^2\)  
  - supremum norm overapproximates circle with box (easier arithmetic)
- Velocity follows acceleration: \(v = old(v) + at\)
Interpretation of Safety Constraints

How much Safety Distance?

- Depends on reaction time and speed
- \[ 2b \|p - m\|_\infty \geq v^2 + (A + b)(A\varepsilon^2 + 2\varepsilon v) \]
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
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots

5 Synthesize Monitors

6 Case Studies

7 Summary
Real CPS

Verification Results

Proof
Reachability Analysis

safe
Real CPS

Model $\alpha^*$

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

$v := v + 1$

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

$x' = v$

Proof

Reachability Analysis

Verification Results

Verification results about models only apply if CPS fits to the model; Verifiably correct runtime model validation.
Formal Verification in CPS Development

Real CPS

Model

Abstract

Proof

Reachability Analysis

Verification Results

Challenge

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

Verifiably correct runtime model validation

\[
v := v + 1
\]

\[
x' = v
\]
ModelPlex ensures that verification results about models apply to CPS implementations.

Diagram:

- Model $\alpha$ from $i-1$ to $i$.
- Control safe? from $i$ to $i+1$.
- Plant until next cycle? from $i+1$.

Diagram elements:
- Model adequate?
- Turn
- Predict

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.

Nathan Fulton, Stefan Mitsch, André Platzer
KeYmaera X Tutorial: From Idea to Provably Safe Implementation
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”

Diagram showing an airplane with labels for Sensors, Controller, and Actuators.
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 $\alpha$?

Model $\alpha$:

- $i - 1$ to $i$ through $\alpha$

Logical $d_L$:

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

Semantical:

- $\langle \alpha \rangle (x = x +)$

Arithmetical:

- $d_L$ proof check at runtime (efficient)

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: From Idea to Provably Safe Implementation

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

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

reachability relation of $\alpha$
When are two states linked through a run of model $\alpha$?

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

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

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

Lemma: exists a run of $\alpha$ to a state where $x = x^+$
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**
When are two states linked through a run of model $\alpha$?

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

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

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

**Offline**

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

---

KeYmaera X Tutorial: From Idea to Provably Safe Implementation

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

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

**Offline**

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

\[ \Downarrow \text{Lemma} \]

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

\[ \Downarrow \text{\( \mathcal{L} \) proof} \]

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

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

RV'14, FMSD'16

Nathan Fulton, Stefan Mitsch, André Platzer | KeYmaera X Tutorial: From Idea to Provably Safe Implementation
Logic reduces CPS safety to runtime monitor with offline proof

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

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

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
Logic reduces CPS safety to runtime monitor with offline proof

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

\[
\text{Model } \alpha \rightarrow \nu
\]

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

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

\[
\text{Arithmetical: } (\omega, \nu) \models F(x^-, x^+) \text{ check at runtime (efficient)}
\]

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

**dL proof** \( A \rightarrow [\alpha]S \)

**Offline**

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

**Semantical:**

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

\( \uparrow \) **Lemma**

**Logical dL:**

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

\( \uparrow \) **dL proof**

**Arithmetical:**

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

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

RV’14, FMSD’16

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: From Idea to Provably Safe Implementation
Logic reduces CPS safety to runtime monitor with offline proof

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

Init $\omega \in [A]$ Safe $\nu \in [S]$

Model $\alpha$ $\in$ $\omega \cap \nu$

$\checkmark$ Lemma

$\checkmark$ dC proof

check at runtime (efficient)

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

逻辑减少CPS安全性到运行时监视器，具有离线证明。

**Logic** reduces **CPS** safety to **run**time **monitor** with **offline** proof.

\[
\text{Model } \alpha \rightarrow [\alpha] S
\]

**Offline**

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

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

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

\checkmark check at runtime (efficient)

RV’14, FMSD’16
Logical Reductions for Model Safety Transfer

Logic reduces CPS safety to runtime monitor with offline proof

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

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
Logic reduces CPS safety to runtime monitor with offline proof

Not initial state. Model repeats...

\( A \rightarrow [\alpha]S \)

\( \omega \) \( \in [A] \) \( \Rightarrow \) \( \omega \) \( \in [\alpha]S \)

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**
Logic reduces CPS safety to runtime monitor with offline proof

\[ A \rightarrow [\alpha^*]S \]

**Offline**

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

**Semantical:**

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

\[ \uparrow \text{Lemma} \]

**Logical dL:**

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

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

**Arithmetical:**

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

\[ \left\uparrow \text{check at runtime (efficient)} \right. \]

*Nathan Fulton, Stefan Mitsch, André Platzer*  
*KeYmaera X Tutorial: From Idea to Provably Safe Implementation*
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
   - Differential Invariants
   - Example: Elementary Differential Invariants
   - Example: Ground Robots

5. Synthesize Monitors

6. Case Studies

7. Summary
Verified CPS Applications
Verified CPS Applications

FM’11, LMCS’12, ICCPS’12, ITSC’11, ITSC’13, IJCAR’12
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
   ● 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

REST-API
- Proof View
- Tactics
- Models
- Proof Log
- start/stop/pause/resume
- controls
- observes
- stores

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

HyDRA Server
- Tactical Prover
- Proof Tree
- Proof Strategies
- dL Tactics
- Combinators
- Wrappers for Kernel Primitives
- manages
- uses
- executes
- combines
- Scheduler executes tactics on tools/ CPU cores

KeYmaera X Kernel (soundness-critical, Scala)
- Axiomatic Core
  - Axioms
  - Proof Certificates
  - Uniform Substitution
  - Bound Renaming
  - Propositional Sequent Calculus with Skolemization
- Real Quantifier Elimination
- Differential Equation Solving
- ...

Nathan Fulton, Stefan Mitsch, André Platzer | KeYmaera X Tutorial: From Idea to Provably Safe Implementation | 39 / 40
KeYmaera X Theorem Prover for Hybrid Systems

**differential dynamic logic**

\[ \mathcal{dL} = DL + HP \]

- 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

KeYmaera X

KeYmaera X Tutorial: From Idea to Provably Safe Implementation

Nathan Fulton, Stefan Mitsch, André Platzer
Outline

8 Soundness and Completeness

9 More on Differential Invariants
   • Assuming Differential Invariants
   • Example: Differential Cut
   • Differential Calculus
   • Differential Invariants
   • Examples
<table>
<thead>
<tr>
<th>Theorem (Sound &amp; Complete)</th>
<th>(J.Autom.Reas. 2008, LICS’12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dL calculus is a sound &amp; complete axiomatization of hybrid systems relative to either differential equations or discrete dynamics.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corollary (Complete Proof-theoretical Alignment &amp; Bridging)</th>
</tr>
</thead>
<tbody>
<tr>
<td>proving continuous = proving hybrid = proving discrete</td>
</tr>
</tbody>
</table>
Theorem (Sound & Complete) (J.Autom.Reas. 2008, LICS’12)

\(d\mathcal{L}\) calculus is a sound & complete axiomatization of hybrid systems relative to either differential equations or discrete dynamics.

Corollary (Complete Proof-theoretical Alignment & Bridging)

proving continuous = proving hybrid = proving discrete
Theorem (Sound & Complete) (J. Autom. Reas. 2008, LICS’12)

\[ \mathcal{L} \text{ calculus is a sound & complete axiomatization of hybrid systems relative to either differential equations or discrete dynamics.} \]

Corollary (Complete Proof-theoretical Alignment & Bridging)

proving continuous = proving hybrid = proving discrete
Outline

8 Soundness and Completeness

9 More on Differential Invariants
- Assuming Differential Invariants
- Example: Differential Cut
- Differential Calculus
- Differential Invariants
- Examples
Assuming Differential Invariance

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

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

(loop)

\[ J \vdash [\alpha]J \]
\[ \frac{J \vdash [\alpha^*]J}{J \vdash [\alpha^*]J} \]
Assuming Differential Invariance

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

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

Example (Restrictions)

\[ v^2 - 2v + 1 = 0 \vdash [v' = w, w' = -v]v^2 - 2v + 1 = 0 \]
Assuming Differential Invariance

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

Example (Restrictions)

\[ v^2 - 2v + 1 = 0 \vdash [v' := w][w' := -v]2vv' - 2v' = 0 \]
\[ v^2 - 2v + 1 = 0 \vdash [v' = w, w' = -v]v^2 - 2v + 1 = 0 \]
Assuming Differential Invariance

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

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

Example (Restrictions)

\[
\begin{align*}
\nu^2 - 2\nu + 1 = 0 & \vdash 2\nu w - 2w = 0 \\
\nu^2 - 2\nu + 1 = 0 & \vdash [\nu' := w][w' := -\nu]2\nu\nu' - 2\nu' = 0 \\
\nu^2 - 2\nu + 1 = 0 & \vdash [\nu' = w, w' = -\nu]\nu^2 - 2\nu + 1 = 0
\end{align*}
\]
Assuming Differential Invariance

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

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

Example (Restrictions)

\[ v^2 - 2v + 1 = 0 \vdash 2vw - 2w = 0 \]
\[ v^2 - 2v + 1 = 0 \vdash [v' := w][w' := -v]2vv' - 2v' = 0 \]
\[ v^2 - 2v + 1 = 0 \vdash [v' = w, w' = -v]v^2 - 2v + 1 = 0 \]
Assuming Differential Invariance

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

Example (Restrictions are unsound!)

\[ (\text{unsound}) \]
\[ v^2 - 2v + 1 = 0 \vdash 2vw - 2w = 0 \]
\[ v^2 - 2v + 1 = 0 \vdash [v' = w][w' = -v]2vv' - 2v' = 0 \]
\[ v^2 - 2v + 1 = 0 \vdash [v' = w, w' = -v]v^2 - 2v + 1 = 0 \]
DC  $x^3 \geq -1 \land y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]x^3 \geq -1$
Differential Cuts

\[
\text{DC } x^3 \geq -1 \land y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2] x^3 \geq -1
\]

\[
\text{DI } y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2] y^5 \geq 0
\]
\[ DC \quad x^3 \geq -1 \land y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]x^3 \geq -1 \]

\[ [\vdash x' = (x - 2)^4 + y^5, y' = y^2]y^5 \geq 0 \]

\[ y^5 \geq 0 \vdash [x = (x - 2)^4 + y^5, y' = y^2]y^5 \geq 0 \]
Ex: Differential Cuts

\[ \begin{aligned}
\text{DC} & \quad x^3 \geq -1 \land y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]x^3 \geq -1 \\
\text{QE} & \quad \vdash 5y^4y^2 \geq 0 \\
[':=] & \quad \vdash [x' := (x - 2)^4 + y^5][y' := y^2]5y^4y' \geq 0 \\
\text{DI} & \quad y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]y^5 \geq 0
\end{aligned} \]
DC  $x^3 \geq -1 \land y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]x^3 \geq -1$

*  

QE  

\[ \vdash 5y^4y^2 \geq 0 \]

[':=]  

\[ \vdash [x' = (x - 2)^4 + y^5][y' = y^2]5y^4y' \geq 0 \]

DI  

\[ y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]y^5 \geq 0 \]
Ex: 

**Differential Cuts**

\[
\begin{align*}
\text{DI} & \quad x^3 \geq -1 \vdash [x' = (x - 2)^4 + y^5, y' = y^2 \land y^5 \geq 0]x^3 \geq -1 \\
\text{DC} & \quad x^3 \geq -1 \land y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]x^3 \geq -1
\end{align*}
\]

\[
\begin{align*}
* & \quad \vdash 5y^4y^2 \geq 0 \\
[:=] & \quad \vdash [x' := (x - 2)^4 + y^5][y' := y^2]5y^4y' \geq 0 \\
\text{DI} & \quad y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]y^5 \geq 0
\end{align*}
\]
Ex: Differential Cuts

\[ y^5 \geq 0 \vdash [x' := (x - 2)^4 + y^5][y' := y^2]2x^2x' \geq 0 \]

**DI**

\[ x^3 \geq -1 \vdash [x' = (x - 2)^4 + y^5, y' = y^2 \& y^5 \geq 0]x^3 \geq -1 \]

**DC**

\[ x^3 \geq -1 \land y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]x^3 \geq -1 \]

\[
\begin{align*}
\text{QE} & \quad \vdash 5y^4y^2 \geq 0 \\
\text{[':=]} & \quad \vdash [x' := (x - 2)^4 + y^5][y' := y^2]5y^4y' \geq 0 \\
\text{DI} & \quad y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]y^5 \geq 0
\end{align*}
\]

Nathan Fulton, Stefan Mitsch, André Platzer | KeYmaera X Tutorial: From Idea to Provably Safe Implementation
Ex: Differential Cuts

\[ y^5 \geq 0 \vdash 2x^2((x - 2)^4 + y^5) \geq 0 \]

\[ [\mathbf{\prime}:=] \]
\[ y^5 \geq 0 \vdash [x' := (x - 2)^4 + y^5][y' := y^2]2x^2x' \geq 0 \]

\[ \mathbf{DI} \]
\[ x^3 \geq -1 \vdash [x' = (x - 2)^4 + y^5, y' = y^2 \& y^5 \geq 0]x^3 \geq -1 \]

\[ \mathbf{DC} \]
\[ x^3 \geq -1 \land y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]x^3 \geq -1 \]

\[ \star \]

\[ \mathbf{QE} \]
\[ \vdash 5y^4y^2 \geq 0 \]

\[ [\mathbf{\prime}:=] \]
\[ \vdash [x' := (x - 2)^4 + y^5][y' := y^2]5y^4y' \geq 0 \]

\[ \mathbf{DI} \]
\[ y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]y^5 \geq 0 \]
Ex: Differential Cuts

* 

**QE**

\[ y^5 \geq 0 \vdash 2x^2((x - 2)^4 + y^5) \geq 0 \]

\[ [\cdot :=] \]

\[ y^5 \geq 0 \vdash [x' := (x - 2)^4 + y^5][y' := y^2]2x^2x' \geq 0 \]

**DI**

\[ x^3 \geq -1 \vdash [x' = (x - 2)^4 + y^5, y' = y^2 \& y^5 \geq 0]x^3 \geq -1 \]

**DC**

\[ x^3 \geq -1 \& y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]x^3 \geq -1 \]

* 

**QE**

\[ \vdash 5y^4y^2 \geq 0 \]

\[ [\cdot :=] \]

\[ \vdash [x' := (x - 2)^4 + y^5][y' := y^2]5y^4y' \geq 0 \]

**DI**

\[ y^5 \geq 0 \vdash [x' = (x - 2)^4 + y^5, y' = y^2]y^5 \geq 0 \]
Differential Equation Axioms & Differential Axioms

\[ \text{DW} \ [x' = f(x) \& Q]Q \]

\[ \text{DC} \ ( [x' = f(x) \& Q]P \iff [x' = f(x) \& Q \land r(x)]P) \]
\[ \iff [x' = f(x) \& Q]r(x) \]

\[ \text{DE} \ [x' = f(x) \& Q]P \iff [x' = f(x) \& Q][x' := f(x)]P \]

\[ \text{DI} \ [x' = f(x) \& Q]P \iff (Q \to P \land [x' = f(x)](P')) \]

\[ \text{DG} \ [x' = f(x) \& Q]P \iff \exists y \ [x' = f(x), y' = a(x)y + b(x) \& Q]P \]

\[ \text{DS} \ [x' = c() \& Q]P \iff \forall t \geq 0 ((\forall 0 \leq s \leq t q(x + c()s)) \to [x := x + c()t]P) \]

\[ [':=] \ [x' := e]p(x') \iff p(e) \]
\[ +' (e + k)' = (e)' + (k)' \]
\[ .' (e \cdot k)' = (e)' \cdot k + e \cdot (k)' \]
\[ o' [y := g(x)][y' := 1]((f(g(x)))') = (f(y))' \cdot (g(x))' \]
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, CADE'15
Differential Invariants for Differential Equations

Differential Invariant

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

Differential Cut

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

Differential Ghost

\[ x' = f(x) \]

Logic

Provability theory

Math

Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15

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

Differential Invariant

\[ x' = f(x) \]

Differential Cut

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

Differential Ghost

\[ x \rightarrow f(x) \]

Logic

Provability theory

Math

Characteristic PDE

JLogCompt'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15
Differential Invariants for Differential Equations

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

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

\[ \text{Logic Provability theory} \]

\[ \text{Math Characteristic PDE} \]

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15
Differential Invariants for Differential Equations

Differential Invariant

$\frac{dx}{dt} = f(x)$

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

Logic

Provability theory

Math

Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15
Differential Invariants for Differential Equations

Differential Invariant

\[ x' = f(x) \]

Differential Cut

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

Differential Ghost

Logic

Provability theory

Math

Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15

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

\[ x' = f(x) \]

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

\[ \text{inv} \]

\[ \text{DI} \]

\[ \text{DI}_> \]

\[ \text{DI}_≤ \]

\[ \text{DI}_= \]

\[ \text{Logic} \]

\[ \text{Provability theory} \]

\[ \text{Math} \]

\[ \text{Characteristic PDE} \]

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15

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

Differential Invariant

\[
\frac{\mathrm{d}x}{\mathrm{d}t} = f(x)
\]

Differential Cut

\[
\frac{\mathrm{d}y}{\mathrm{d}t} = g(x, y)
\]

Differential Ghost

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

Logic

Provability theory

Math

Characteristic PDE

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, CADE’15

Nathan Fulton, Stefan Mitsch, André Platzer

KeYmaera X Tutorial: From Idea to Provably Safe Implementation
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

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

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

Logic

Provability theory

Math

Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15
Differential equations cannot leave their evolution domains. Implies:

\[ \left[ \dot{x} = f(x) & Q \right] P \leftrightarrow \left[ \dot{x} = f(x) & Q \right] (Q \rightarrow P) \]
### Axiom (Differential Cut) (CADE’15)

\[
\text{DC} \quad ([x' = f(x) \& Q]P \leftrightarrow [x' = f(x) \& Q \land r(x)]P) \\
\quad \leftarrow [x' = f(x) \& Q]r(x)
\]

---

**DC is a cut for differential equations.**

**DC is a differential modal modus ponens K.**

Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
### Axiom (Differential Cut) (CADE'15)

\[
\text{DC: } ([x' = f(x) \& Q]P \leftrightarrow [x' = f(x) \& Q \land r(x)]P) \\
\leftrightarrow [x' = f(x) \& Q]r(x)
\]

---

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \(r(x)\), then might as well restrict state space to \(r(x)\).
Axiom (Differential Cut) (CADE’15)

$$\text{DC} \quad ([x' = f(x) \& Q]P \leftrightarrow [x' = f(x) \& Q \land r(x)]P) \leftrightarrow [x' = f(x) \& Q]r(x)$$

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave $r(x)$, then might as well restrict state space to $r(x)$. 

Nathan Fulton, Stefan Mitsch, André Platzer | KeYmaera X Tutorial: From Idea to Provably Safe Implementation
Axiom (Differential Cut) \[(\text{DC})\]
\[
\begin{align*}
[x' = f(x) \& Q]P & \iff [x' = f(x) \& Q \land r(x)]P \\
& \iff [x' = f(x) \& Q]r(x)
\end{align*}
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \(r(x)\), then might as well restrict state space to \(r(x)\).
Axiom (Differential Cut) (CADE’15)

\[
DC \quad ([x' = f(x) \& Q] P \leftrightarrow [x' = f(x) \& Q \land r(x)] P) \\
\quad \leftrightarrow [x' = f(x) \& Q] r(x)
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Differential Equation Axioms

Axiom (Differential Cut) \((\text{CADE'15})\)

\[
\text{DC} \quad ([x' = f(x) & Q]P \leftrightarrow [x' = f(x) & Q \land r(x)]P) \leftarrow [x' = f(x) & Q]r(x)
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens \(K\).
Can’t leave \(r(x)\), then might as well restrict state space to \(r(x)\).
### Axiom (Differential Cut) (CADE’15)

\[
\text{DC} \quad ([x' = f(x) \& Q] P \leftrightarrow [x' = f(x) \& Q \land r(x)] P) \\
\quad \rightarrow [x' = f(x) \& Q] r(x)
\]

**Diagram**

- A line with a shaded area representing the differential equation.
- Arrows indicating the direction of change in the variable.
- A point labeled 0.
- Another point labeled r.

**Text**

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Axiom (Differential Cut) (CADE’15)

\[
DC \quad ([x' = f(x) \land Q]P \leftrightarrow [x' = f(x) \land Q \land r(x)]P) \\
\leftarrow [x' = f(x) \land Q]r(x)
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).

Nathan Fulton, Stefan Mitsch, André Platzer  KeYmaera X Tutorial: From Idea to Provably Safe Implementation
**Axiom (Differential Cut)**

\[ \text{DC} \quad ([x' = f(x) \& Q]P \iff [x' = f(x) \& Q \land r(x)]P) \]

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

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Axiom (Differential Invariant) (CADE’15)

\[ \text{DI} \ \ [x' = f(x) \& Q]P \leftarrow (Q \rightarrow P \land [x' = f(x) \& Q](P)') \]

Differential invariant: \( p(x) \) true now and its differential \((p(x))'\) true always

What’s the differential of a formula???

What’s the meaning of a differential term ... in a state???
Axiom (Differential Effect)  

\[
\text{DE } [x' = f(x) & Q]P \leftrightarrow [x' = f(x) & Q][x' := f(x)]P
\]

Effect of differential equation on differential symbol \(x'\) 

\([x' := f(x)]\) instantly mimics continuous effect \([x' = f(x)]\) on \(x'\) 

\([x' := f(x)]\) selects vector field \(x' = f(x)\) for subsequent differentials
Differential Equation Axioms

**Axiom (Differential Ghost) (CADE’15)**

\[
DG \quad [x' = f(x) \& Q]P \iff \exists y \quad [x' = f(x), y' = a(x)y + b(x) \& Q]P
\]

**Differential ghost/auxiliaries:** extra differential equations that exist

Can cause new invariants

“Dark matter” counterweight to balance conserved quantities
Differential Equation Axioms

Axiom (Differential Solution) (CADE’15)

\[
DS \ [x' = c() \& Q]P \leftrightarrow \forall t \geq 0 \ ((\forall 0 \leq s \leq t \ q(x+c(s))) \rightarrow [x := x+c()t]P)
\]

Differential solutions: solve differential equations with DG, DC and inverse companions
Example: Differential Invariants Don’t Solve. Prove!

1. DI proves a property of an ODE inductively by its differentials
2. DE exports vector field, possibly after DW exports evolution domain
3. CE+CQ reason efficiently in Equivalence or eQuational context
4. G isolates postcondition
5. [’:=] differential substitution uses vector field
6. ’ differential computations are axiomatic (US)

\[ \text{IQ } \vdash x^3 \cdot x + x \cdot x^3 \geq 0 \]

\[ \text{DE } \vdash [x' := x^3]x' \cdot x + x \cdot x' \geq 0 \]

\[ \text{CE } \vdash [x' = x^3][x' := x^3]x' \cdot x + x \cdot x' \geq 0 \]

\[ \text{DI } \vdash x \cdot x \geq 1 \]

\[ \text{US } (x \cdot x)' = (x)' \cdot x + x \cdot (x)' \]

\[ (x \cdot x)' = x' \cdot x + x \cdot x' \]

\[ \text{CQ } (x \cdot x)' \geq 0 \iff x' \cdot x + x \cdot x' \geq 0 \]

\[ (x \cdot x' \geq 1)' \iff x' \cdot x + x \cdot x' \geq 0 \]
Differential Substitution Lemmas

Lemma (Differential lemma)

If $\varphi \models x' = f(x) \land Q$ for duration $r > 0$, then for all $0 \leq \zeta \leq r$:

\[
\begin{align*}
\llbracket (e)' \rrbracket \varphi(\zeta) &= \frac{d \llbracket e \rrbracket \varphi(t)}{dt}(\zeta) \quad \text{Analytic}
\end{align*}
\]

Lemma (Differential assignment)

If $\varphi \models x' = f(x) \land Q$ then $\varphi \models P \leftrightarrow [x' := f(x)]P$

Lemma (Derivations)

\[
\begin{align*}
(e + k)' &= (e)' + (k)' \\
(e \cdot k)' &= (e)' \cdot k + e \cdot (k)'
\end{align*}
\]

\[
[y := e][y' := 1]((f(e))' = (f(y))' \cdot (e)') \quad \text{for } y, y' \not\in e
\]

\[
(c())' = 0 \quad \text{for arity 0 functions/numbers } c()
\]
André Platzer.
Logics of dynamical systems. 
In LICS [13], pages 13–24. 

André Platzer.
Differential dynamic logic for hybrid systems. 

André Platzer.
A uniform substitution calculus for differential dynamic logic. 
doi:10.1007/978-3-319-21401-6_32.

André Platzer.
Differential game logic. 
André Platzer.
The complete proof theory of hybrid systems.
In LICS [13], 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.
doi:10.1007/978-3-540-70545-1_17.

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.

Khalil Ghorbal, Andrew Sogokon, and André Platzer.
Invariance of conjunctions of polynomial equalities for algebraic differential equations.
doi:10.1007/978-3-319-10936-7_10.

Stefan Mitsch and André Platzer.
ModelPlex: Verified runtime validation of verified cyber-physical system models.