CoasterX: A Case Study in Component-Driven Hybrid Systems Proof Automation
Based on ADHS ’18 [BLCP18]

Brandon Bohrer
(joint work with Adriel Luo, Xuean Chuang, André Platzer)

Logical Systems Lab
Computer Science Department
Carnegie Mellon University

Speaking Skills, Mar 26 2018
Outline

1. Motivation

2. Approach

3. Modeling and Verification
   - Background: dL
   - Identifying Assumptions
   - Formal Specification
   - Formal Verification

4. Evaluation

5. Future Work and Conclusion
Roller Coasters are Safety-Critical Systems

Top Thrill

Steel Phantom

Mindbender

Joker’s Jinx

Phantom’s Revenge

Fujin Raijin II

Rollback

Head Injury

Derailment

[BLCP18]
Formal Proofs in $d\mathcal{L}$ Ensure Safe Designs

Top Thrill  Steel Phantom  Mindbender
Rollback  Head Injury  Derailment

Identify:
- Notion of safety $Post \ (acc < acc_{hi})$
Formal Proofs in $\mathcal{L}$ Ensure Safe Designs

Identify:

- Notion of safety $Post (acc < acc_{hi})$
- Safe conditions $Pre (v = v_0)$
Formal Proofs in $\mathcal{dL}$ Ensure Safe Designs

Top Thrill  Steel Phantom  Mindbender

Rollback  Head Injury  Derailment

$\Downarrow$

$\text{Pre} \rightarrow [\text{phys}]\text{Post}$

Identify:

- Notion of safety $\text{Post} (\text{acc} < \text{acc}_{\text{hi}})$
- Safe conditions $\text{Pre} (v = v_0)$

Verify physical environment design $\text{phys} (\{x' = \ldots, y' = \ldots\})$
Simulations typically used today [XXLY12, Wei15]

<table>
<thead>
<tr>
<th>Approach</th>
<th>Pro</th>
<th>Con</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulate</td>
<td>Rich dynamics, easy</td>
<td>Low rigor+precision</td>
</tr>
<tr>
<td>Verify</td>
<td>High rigor+precision</td>
<td>Simple dynamics, <strong>hard</strong></td>
</tr>
</tbody>
</table>
Verifying Physical Designs is a Challenge

- How do we verify models at scale?
- How do we make verification accessible to non-experts?
Verifying Environment Designs is Important
Outline

1 Motivation

2 Approach

3 Modeling and Verification
   Background: $d\mathcal{L}$
   Identifying Assumptions
   Formal Specification
   Formal Verification

4 Evaluation

5 Future Work and Conclusion
Component-Driven Proof Automation
Enables Design Verification

Goal
Accessible

Solution
High-level graphical modeling
### Component-Driven Proof Automation Enables Design Verification

<table>
<thead>
<tr>
<th>Goal</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessible</td>
<td>High-level graphical modeling</td>
</tr>
<tr>
<td>Rigorous</td>
<td>Formal proof checked by small prover core</td>
</tr>
</tbody>
</table>

- **GUI Builder**
- **Component Backend**
- **CoasterX Backend**
- **KeYmaera X Prover Core (1700 Lines)**

\[ \text{dL fml.} \]
\[ \text{dL pf.} \]
Component-Driven Proof Automation Enables Design Verification

<table>
<thead>
<tr>
<th>Goal</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessible</td>
<td>High-level graphical modeling</td>
</tr>
<tr>
<td>Rigorous</td>
<td>Formal proof checked by small prover core</td>
</tr>
<tr>
<td>Scalable</td>
<td>Proof scales by exploiting component structure</td>
</tr>
</tbody>
</table>

CoasterX Backend

GUI Builder

KeYmaera X Prover Core (1700 Lines)
Track Sections are Components for Coasters

Generic Component
Track Sections are Components for Coasters

Generic Component

Automatic Composition
Track Sections are Components for Coasters

Generic Component

Automatic Composition
Track Sections are Components for Coasters

Generic Component

Automatic Composition
Track Sections are Components for Coasters

Generic Component

Automatic Composition
Track Sections are Components for Coasters

Generic Component

Automatic Composition
1 Motivation

2 Approach

3 Modeling and Verification
   Background: dL
   Identifying Assumptions
   Formal Specification
   Formal Verification

4 Evaluation

5 Future Work and Conclusion
Outline

1 Motivation

2 Approach

3 Modeling and Verification
   Background: dL
   Identifying Assumptions
   Formal Specification
   Formal Verification

4 Evaluation

5 Future Work and Conclusion
**Background: d\(L\) Formulas**

\[ P, Q ::= P \land Q \mid \neg P \mid \forall x P \mid \exists x P \mid \theta_1 \geq \theta_2 \mid [\alpha]P \mid \langle \alpha \rangle P \]

**Example:** Pre \(\rightarrow\) [phys]Post

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P \land Q, \neg P)</td>
<td>Classical propositional connectives</td>
</tr>
</tbody>
</table>
### Background: \( \mathcal{dL} \) Formulas

\[
P, Q ::= P \land Q \mid \neg P \mid \forall x P \mid \exists x P \mid \theta_1 \geq \theta_2 \mid [\alpha]P \mid \langle \alpha \rangle P
\]

**Example:** \( \text{Pre} \rightarrow [\text{phys}]\text{Post} \)

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \land Q, \neg P )</td>
<td>Classical propositional connectives</td>
</tr>
<tr>
<td>( \forall x P, \exists x P )</td>
<td>First-order real quantifiers</td>
</tr>
<tr>
<td>( \theta_1 \geq \theta_2 )</td>
<td>Real arithmetic comparisons</td>
</tr>
</tbody>
</table>
Background: \( \mathcal{dL} \) Formulas

\[
P, Q ::= P \land Q \mid \neg P \mid \forall x P \mid \exists x P \mid \theta_1 \geq \theta_2 \mid [\alpha]P \mid \langle \alpha \rangle P
\]

**Example:** \( \text{Pre} \rightarrow [\text{phys}]\text{Post} \)

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \land Q, \neg P )</td>
<td>Classical propositional connectives</td>
</tr>
<tr>
<td>( \forall x P, \exists x P )</td>
<td>First-order real quantifiers</td>
</tr>
<tr>
<td>( \theta_1 \geq \theta_2 )</td>
<td>Real arithmetic comparisons</td>
</tr>
<tr>
<td>( [\alpha]P )</td>
<td>After ( \alpha ) runs, ( P ) always holds</td>
</tr>
<tr>
<td>( \langle \alpha \rangle P )</td>
<td>After ( \alpha ) runs, ( P ) sometimes holds</td>
</tr>
</tbody>
</table>
Background: Hybrid Programs

$$\alpha, \beta ::= \ ?P \ | \ x := \theta \ | \ \{x' = \theta \ & \ P\} \ | \ \alpha \cup \beta \ | \ \alpha;\beta \ | \ \alpha^*$$

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>?P</td>
<td>Assert formula $P$, else fail</td>
</tr>
</tbody>
</table>
Background: Hybrid Programs

\[ \alpha, \beta \ ::= \ ?P \mid x := \theta \mid \{ x' = \theta \land P \} \mid \alpha \cup \beta \mid \alpha ; \beta \mid \alpha^* \]

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>?P</td>
<td>Assert formula ( P ), else fail</td>
</tr>
<tr>
<td>( x := \theta )</td>
<td>Assign value of term ( \theta ) to ( x )</td>
</tr>
</tbody>
</table>
Background: Hybrid Programs

$$\alpha, \beta ::= \ ?P \ | \ x := \theta \ | \ \{x' = \theta \ & \ P\} \ | \ \alpha \cup \beta \ | \ \alpha;\beta \ | \ \alpha^*$$

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>?P</td>
<td>Assert formula P, else fail</td>
</tr>
<tr>
<td>x := \theta</td>
<td>Assign value of term \theta to x</td>
</tr>
<tr>
<td>{x' = \theta \ &amp; \ P}</td>
<td>Evolve x at continuous rate \theta</td>
</tr>
</tbody>
</table>
Background: Hybrid Programs

\[ \alpha, \beta ::= \alpha \cup \beta | \alpha; \beta | \alpha^* \]

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>?P</td>
<td>Assert formula ( P ), else fail</td>
</tr>
<tr>
<td>( x := \theta )</td>
<td>Assign value of term ( \theta ) to ( x )</td>
</tr>
<tr>
<td>( {x' = \theta \land P} )</td>
<td>Evolve ( x ) at continuous rate ( \theta )</td>
</tr>
<tr>
<td></td>
<td><em>Evolution domain constraint ( P ) asserted continuously</em></td>
</tr>
</tbody>
</table>
Background: Hybrid Programs

\[
\alpha, \beta ::= \ ?P \mid x := \theta \mid \{x' = \theta \& P\} \mid \alpha \cup \beta \mid \alpha;\beta \mid \alpha^*
\]

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>?P</td>
<td>Assert formula ( P ), else fail</td>
</tr>
<tr>
<td>( x := \theta )</td>
<td>Assign value of term ( \theta ) to ( x )</td>
</tr>
<tr>
<td>( {x' = \theta &amp; P} )</td>
<td>Evolve ( x ) at continuous rate ( \theta ) \nEvolution domain constraint ( P ) asserted continuously \n( P ) must also hold initially (like ( ?P ))</td>
</tr>
</tbody>
</table>
Background: Hybrid Programs

\[ \alpha, \beta ::= \ ?P \mid x := \theta \mid \{x' = \theta & P\} \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* \]

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>?P</td>
<td>Assert formula ( P ), else fail</td>
</tr>
<tr>
<td>( x := \theta )</td>
<td>Assign value of term ( \theta ) to ( x )</td>
</tr>
<tr>
<td>( {x' = \theta &amp; P} )</td>
<td>Evolve ( x ) at continuous rate ( \theta )</td>
</tr>
<tr>
<td>( \text{Evolution domain constraint } P ) asserted continuously ( P ) must also hold initially (like ( ?P ))</td>
<td></td>
</tr>
<tr>
<td>( \alpha \cup \beta )</td>
<td>Choose either ( \alpha ) or ( \beta ) nondeterministically</td>
</tr>
</tbody>
</table>
**Background: Hybrid Programs**

\[ \alpha, \beta ::= \ ?P \mid x := \theta \mid \{ x' = \theta \land P \} \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* \]

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>?P</td>
<td>Assert formula ( P ), else fail</td>
</tr>
<tr>
<td>( x := \theta )</td>
<td>Assign value of term ( \theta ) to ( x )</td>
</tr>
<tr>
<td>( { x' = \theta \land P } )</td>
<td>Evolve ( x ) at continuous rate ( \theta )</td>
</tr>
<tr>
<td>( \text{Evolution domain constraint } P ) asserted continuously ( P ) must also hold initially (like ( ?P ))</td>
<td></td>
</tr>
<tr>
<td>( \alpha \cup \beta )</td>
<td>Choose either ( \alpha ) or ( \beta ) nondeterministically</td>
</tr>
<tr>
<td>( \alpha; \beta )</td>
<td>First ( \alpha ) then ( \beta ) in any resulting state</td>
</tr>
</tbody>
</table>
Background: Hybrid Programs

\[ \alpha, \beta ::= \begin{array}{l}
?P \ | \ x := \theta \ | \ \{x' = \theta \ \& \ P\} \ | \ \alpha \cup \beta \ | \ \alpha; \beta \ | \ \alpha^* 
\end{array} \]

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>?P</td>
<td>Assert formula ( P ), else fail</td>
</tr>
<tr>
<td>( x := \theta )</td>
<td>Assign value of term ( \theta ) to ( x )</td>
</tr>
<tr>
<td>( {x' = \theta \ &amp; \ P} )</td>
<td>Evolve ( x ) at continuous rate ( \theta )</td>
</tr>
<tr>
<td>( \alpha \cup \beta )</td>
<td>Choose either ( \alpha ) or ( \beta ) nondeterministically</td>
</tr>
<tr>
<td>( \alpha; \beta )</td>
<td>First ( \alpha ) then ( \beta ) in any resulting state</td>
</tr>
<tr>
<td>( \alpha^* )</td>
<td>Loop ( \alpha ) nondeterministically ( n \geq 0 ) times</td>
</tr>
</tbody>
</table>
Outline

1. Motivation
2. Approach
3. Modeling and Verification
   Background: dL
   Identifying Assumptions
   Formal Specification
   Formal Verification
4. Evaluation
5. Future Work and Conclusion
Velocity and Acceleration Bounds are Fundamental

Rollback
\[ 0 < v_{lo} \leq v \]

Head Injury
\[ |a| \leq a_{hi} \]

Derailment
\[ |a| \leq a_{hi} \]

[AST17]
Tracks are 2D

• 2D modeling greatly simplifies GUI

• Vertical and horizontal bounds only (no lateral bound)

• Ignores *banking, wind, roll resistance* (1-2%)
Acceleration Bound is Conservative

Top Thrill
Steel Phantom
Mindbender

Joker’s Jinx
Phantom’s Revenge
Fujin Raijin II

Rollback
Head Injury
Derailment

⚠️(>)
✔️(<)
✔️(<)
Conservative Bound Suffices for Phantom

Top Thrill
Steel Phantom
Mindbender
Joker’s Jinx
Phantom’s Revenge
Fujin Raijin II

Rollback
⚠️ (>)

Head Injury
✅ (<)

Derailment
✅ (<)
Outline

1. Motivation
2. Approach
3. Modeling and Verification
   - Background: dL
   - Identifying Assumptions
   - Formal Specification
   - Formal Verification
4. Evaluation
5. Future Work and Conclusion
Example

\[
\text{phys} \equiv \{\{x' = \sqrt{2}/2 \ v, y' = \sqrt{2}/2 \ v, v' = -\sqrt{2}/2 \ g & 0 \leq x \leq 100\} \\
\cup \{x' = dx \ v, y' = dy, v' = -dy \ g, dx' = -dy \ v/100\sqrt{2}, \\
\quad dy' = dx \ v/100\sqrt{2} & 100 \leq x \leq 200\} \\
\cup \{x' = \sqrt{2}/2 \ v, y' = -\sqrt{2}/2 \ v, v' = \sqrt{2}/2 \ g & 200 \leq x \leq 300\}\}
\]
phys \equiv \{\{\text{Line}(...) \& \ 0 \leq x \leq 100\}\}
\cup \{\{\text{Arc}(...) \& \ 100 \leq x \leq 200\}\}
\cup \{\{\text{Line}(...) \& \ 200 \leq x \leq 300\}\}\star
Example

\[ \text{phys} \equiv \left\{ \{ \text{Line}(...) \ & \ 0 \leq x \leq 100 \} \right\} \]
\[ \cup \left\{ \{ \text{Arc}(...) \ & \ 100 \leq x \leq 200 \} \right\} \]
\[ \cup \left\{ \{ \text{Line}(...) \ & \ 200 \leq x \leq 300 \} \right\}^* \]
Example

phys ≡ \{\{\text{Line(\ldots)} \land 0 \leq x \leq 100\}\}
\cup \{\{\text{Arc(\ldots)} \land 100 \leq x \leq 200\}\}
\cup \{\{\text{Line(\ldots)} \land 200 \leq x \leq 300\}\}\}$
Individual Components are Modeled as ODEs

**Line Segment:**

\[
\text{Line} \overset{\text{def}}{=} \{ x' = v \cdot dx, y' = v \cdot dy, v' = -dy \cdot g \}
\]

& \text{InBounds}\((x_1, x_2, y_1, y_2)\)
Individual Components are Modeled as ODEs

**Line Segment:**

\[
\text{Line} \overset{\text{def}}{=} \{ x' = v \cdot dx, y' = v \cdot dy, v' = -dy \cdot g \quad \& \quad \text{InBounds}(x_1, x_2, y_1, y_2) \}
\]

**Arc Segment:**

\[
\text{Arc} \overset{\text{def}}{=} \{ x' = v \cdot dx, y' = v \cdot dy, v' = -dy \cdot g, \\
\quad dx' = -dy \cdot v/r, dy' = dx \cdot v/r \\
\quad \& \quad \text{InBounds}(x_1, x_2, y_1, y_2) \}
\]
Concrete Parameters are Plugged in From GUI

**Line Segment:**

\[
\text{Line} \quad \equiv \quad \{ x' = v \cdot \Delta x, \; y' = v \cdot \Delta y, \; v' = -\Delta y \cdot g \}
\]

& \quad \text{InBounds}(x_1, x_2, y_1, y_2)\}
Concrete Parameters are Plugged in From GUI

Line Segment:

\[
\text{Line} \triangleq \{ x' = v \cdot dx, y' = v \cdot dy, v' = -dy \cdot g \\
& \text{\ & InBounds}(x_1, x_2, y_1, y_2) \}
\]

\[\downarrow \text{Subst}\]

\[
\text{Line}(1, 0, \ldots) \triangleq \{ x' = v \cdot 1, y' = v \cdot 0, v' = -0 \cdot g \\
& \text{\ & InBounds}(0, 100, 200, 200) \}
\]
Composition is Modeled with Discrete Programs

Let track sections $sec_i$ be component instances:

$$sec_i \overset{\text{def}}{=} \text{Line}(args_i) \text{ or Arc}(args_i)$$

and system model $\alpha$:

$$\text{phys} \overset{\text{def}}{=} (sec_1 \cup \cdots \cup sec_n)^*$$
1 Motivation

2 Approach

3 Modeling and Verification
   Background: dL
   Identifying Assumptions
   Formal Specification
   Formal Verification

4 Evaluation

5 Future Work and Conclusion
Components Verified with Invariants and Solving

- Straight line is solvable, thus decidable.
- Arc needs invariant (energy conservation), proved manually:

\[ E = E_0 \land \text{OnTrack} \rightarrow [\text{Arc}] (E = E_0 \land \text{OnTrack}) \]

- Even for straight line, manual proof more performant
Instantiation is Verified by Substitution

- Conceptually simple step
- Greatly improves performance (20x in some cases)

\[
\text{Line} \overset{\text{def}}{=} \{ x' = v \cdot dx, y' = v \cdot dy, v' = -dy \cdot g \\
& \text{\& InBounds}(x_1, x_2, y_1, y_2) \}
\]

\[
\Downarrow \text{Subst}
\]

\[
\text{Line}(1, 0, \ldots) \overset{\text{def}}{=} \{ x' = v \cdot 1, y' = v \cdot 0, v' = -0 \cdot g \\
& \text{\& InBounds}(0, 100, 200, 200) \}
\]
Composition is Verified by Contract-Checking

- At boundary, invariants for both sections hold
- Checked with arithmetic solving + custom automation

Example:

\[ J_1 \equiv (x = y) \]
\[ J_2 \equiv (y^2 + (x - 200)^2 = 100^2) \]
1 Motivation

2 Approach

3 Modeling and Verification
   Background: dL
   Identifying Assumptions
   Formal Specification
   Formal Verification

4 Evaluation

5 Future Work and Conclusion
We Modeled 6 Real Coasters

- Top Thrill
- Steel Phantom
- Backyard
- El Toro
- Phantom’s Revenge
- Lil’ Phantom
Analysis Distinguished Safe and Unsafe Acceleration

Top Thrill

Steel Phantom (6.5g)

Backyard

El Toro

Phantom’s Revenge (3.5g)

Lil’ Phantom
This is the Largest $d\mathcal{L}$ Model Ever

<table>
<thead>
<tr>
<th>Stats</th>
<th>CoasterX Max</th>
<th>Previous Max (Est.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vars</td>
<td>256</td>
<td>&gt; 27</td>
</tr>
<tr>
<td>Components</td>
<td>56</td>
<td>&gt; 3</td>
</tr>
<tr>
<td>Fml size</td>
<td>52KB</td>
<td>&gt; 6.5KB</td>
</tr>
<tr>
<td>Proof Steps</td>
<td>20M (29K w/ reuse)</td>
<td>&gt; 100K</td>
</tr>
</tbody>
</table>
Scalability is Quadratic

Runtime vs. Problem Size

(on a recent workstation)
<table>
<thead>
<tr>
<th>Component</th>
<th>Time</th>
<th># Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>140s</td>
<td>900K</td>
</tr>
<tr>
<td>Q1 Arc</td>
<td>3.1s</td>
<td>9K</td>
</tr>
<tr>
<td>Q2 Arc</td>
<td>5.1s</td>
<td>14K</td>
</tr>
<tr>
<td>Q3 Arc</td>
<td>3.6s</td>
<td>10K</td>
</tr>
<tr>
<td>Q4 Arc</td>
<td>6.3s</td>
<td>17K</td>
</tr>
</tbody>
</table>

Automatic proof (Line) vastly slower than manual proof (Arcs)
Outline

1 Motivation

2 Approach

3 Modeling and Verification
   Background: dL
   Identifying Assumptions
   Formal Specification
   Formal Verification

4 Evaluation

5 Future Work and Conclusion
Advanced Dynamical Models Answer
Deeper Questions

Acceleration

\[ |a| \leq a_{hi} \]
Advanced Dynamical Models Answer
Deeper Questions

Accelration
\[ |a| \leq a_{hi} \]

Rollback
\[ 0 < v_{lo} \leq v \]

Stuck
\[ 0 < v_{lo} \leq v \]

Friction

Wind
3D Modeling support enables lateral bounds and banking support
Rich Contracts Enable High-Impact Domains

- Transit networks: Contracts at intersections switches
- Flight plans: Contracts at crossing points
Coasters Support Pedagogical Mission

- 15-424 CPS Foundations: Fun applications motivate students
- Course feeds into undergraduate research
- Initial stages were Adriel + Xuean’s 15-424 course project

GPWS
Chute
Pong
Coaster
Chess
Baseball
Questions?

Top Thrill

Steel Phantom

Backyard

El Toro

Phantom’s Revenge

Lil’ Phantom
References I


