Logical Foundations of Autonomous Cyber-Physical Systems
Outline

1. Autonomous Cyber-Physical Systems
2. Foundation: Differential Dynamic Logic
3. ModelPlex: Model Safety Transfer
4. VeriPhy: Executable Proof Transfer
5. Safe Learning in CPSs
6. Applications
   - Airborne Collision Avoidance System
   - Ground Robot Navigation
7. Summary
1. Autonomous Cyber-Physical Systems

2. Foundation: Differential Dynamic Logic

3. ModelPlex: Model Safety Transfer

4. VeriPhy: Executable Proof Transfer

5. Safe Learning in CPSs

6. Applications
   - Airborne Collision Avoidance System
   - Ground Robot Navigation

7. Summary
Cyber-Physical Systems

CPSs combine cyber capabilities with physical capabilities to solve problems that neither part could solve alone.
Cyber-Physical Systems

CPSs combine cyber capabilities with physical capabilities to solve problems that neither part could solve alone.
CPSs combine cyber capabilities with physical capabilities to solve problems that neither part could solve alone.
Cyber-Physical Systems

CPSs combine cyber capabilities with physical capabilities to solve problems that neither part could solve alone.
Autonomous Cyber-Physical Systems Analysis

CPS Analysis
- Simple control
- ODE model
- Strong predictions
- Nondet decisions

AI Learning
- Flexible responses
- “No” model*
- Hard to predict
- Optimal decision ($t \rightarrow \infty$)

Cyber-Physical Systems
CPSs combine cyber capabilities with physical capabilities to solve problems that neither part could solve alone.
Autonomous CPSs Are on the Rise Everywhere

Prospects: Safety & Efficiency & Autonomy

<table>
<thead>
<tr>
<th>Autonomous cars</th>
<th>Autonomous pilots</th>
<th>Robots near humans</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image of autonomous car]</td>
<td>![Image of drones]</td>
<td>![Image of robot near human]</td>
</tr>
</tbody>
</table>

Objective

Best of both worlds: safety from CPS + flexibility from AI
Approach: Safety Proofs for Autonomous CPS

KeYmaera X

ModelPlex proof synthesizes

Model Safety

Compliance Monitor

Autonomous CPS

Model

actions: \{acc, brake\}

motion: \(x'' = a\)

Monitor transfers safety

generates proofs

Proof and invariant search

André Platzer (CMU)

Logical Foundations of Autonomous Cyber-Physical Systems

MOD'19 4 / 21
1 Autonomous Cyber-Physical Systems

2 Foundation: Differential Dynamic Logic

3 ModelPlex: Model Safety Transfer

4 VeriPhy: Executable Proof Transfer

5 Safe Learning in CPSs

6 Applications
   - Airborne Collision Avoidance System
   - Ground Robot Navigation

7 Summary
CPS Analysis

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

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

\[ [\alpha] \varphi \]

\[ x \neq m \]

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

\[ x \neq m \]

\[ x \neq m \]

Diagram:

- Graph representation of differential dynamic logic with an arrow indicating a transition from \( \alpha \varphi \) to \( \varphi \).

Graphical representation:

- A system with 7 time steps (t = 0, 1, ..., 6, 7) and three variables: \( a, v, x \).
- \( a \) varies between -2.5 and 0 with intervals at t = 2, 3, 4.
- \( v \) decreases from 6 to 0 over time, with peaks at t = 2 and troughs at t = 5.
- \( x \) increases from 0 to 10 over time, with a smooth curve.

Legend:
- \( \alpha \) represents a differential event or change.
- \( \varphi \) represents a propositional formula.
- \( x \neq m \) represents an inequality constraint.

André Platzer (CMU)
CPS Analysis

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

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

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

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

\[ x \neq m \]

all runs

\[ a \]

\[ v \]

\[ m \]
CPS Analysis

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

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

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

All runs

\[ \text{init} \]

\[ \text{post} \]
Model Requirements

Proposition (Continuous image computation undecidable)

\[ \phi(D) \cap B = \emptyset \]

is undecidable by evaluating \( \phi(x) \) for arbitrarily effective \( \phi \in C_k(D \subseteq \mathbb{R}^n, \mathbb{R}^m) \) with effective \( D, B \) even if tolerating error \( \epsilon > 0 \) in decisions.

The promise of "no model" is a myth.

HSCC'07

André Platzer (CMU)
Model Requirements

Numerical $\mathbb{R}$-Turing Machine

$\varphi(x)$

$\varphi'(x)$

$x \in \mathbb{R}$

$\mathbb{R}$

$\neq \emptyset$

$= \emptyset$

Proposition (Continuous image computation undecidable)

$\varphi(D) \cap B ? \neq \emptyset$ is undecidable by evaluating $\varphi(x)$ for arbitrarily effective flow $\varphi \in C_k(D \subseteq \mathbb{R}^n, \mathbb{R}^m)$ with effective $D, B$ even if tolerating error $\varepsilon > 0$ in decisions even $\varphi$ smooth polynomial function with $Q$-coefficients even in Blum-Shub-Smale "real Turing machines"

The promise of "no model" is a myth

HSCC'07

André Platzer (CMU)  Logical Foundations of Autonomous Cyber-Physical Systems  MOD'19  6 / 21
Proposition (Continuous image computation undecidable)

\( \varphi(D) \cap B \neq \emptyset \) is undecidable by evaluating \( \varphi(x) \) for

- arbitrarily effective flow \( \varphi \in C^k (D \subseteq \mathbb{R}^n, \mathbb{R}^m) \) with effective \( D, B \)
- even if tolerating error \( \varepsilon > 0 \) in decisions
Proposition (Continuous image computation undecidable)

\( \varphi(D) \cap B \neq \emptyset \) is undecidable by evaluating \( \varphi(x) \) for

- arbitrarily effective flow \( \varphi \in C^k(D \subseteq \mathbb{R}^n, \mathbb{R}^m) \) with effective \( D, B \)
- even if tolerating error \( \varepsilon > 0 \) in decisions
- even \( \varphi \) smooth polynomial function with \( \mathbb{Q} \)-coefficients
- even in Blum-Shub-Smale “real Turing machines”
Model Requirements: Symbolic/Intensional!

Numerical $\mathbb{R}$-Turing Machine

$\phi(x) \neq 0$ for $x \in \mathbb{R}$

Proposition (Continuous image computation undecidable)

$\phi(D) \cap B \neq \emptyset$ is undecidable by evaluating $\phi(x)$ for

- arbitrarily effective flow $\phi \in C^k(D \subseteq \mathbb{R}^n, \mathbb{R}^m)$ with effective $D, B$
- even if tolerating error $\varepsilon > 0$ in decisions
- even $\phi$ smooth polynomial function with $\mathbb{Q}$-coefficients
- even in Blum-Shub-Smale “real Turing machines”

The promise of “no model” is a myth
Outline

1. Autonomous Cyber-Physical Systems
2. Foundation: Differential Dynamic Logic
3. ModelPlex: Model Safety Transfer
4. VeriPhy: Executable Proof Transfer
5. Safe Learning in CPSs
6. Applications
   - Airborne Collision Avoidance System
   - Ground Robot Navigation
7. Summary
Approach: Safety Proofs for Autonomous CPS

Monitor transfers safety

ModelPlex proof synthesizes

KeYmaera X

generates proofs

Model Safety

Compliance Monitor

Autonomous CPS

actions: \{\text{acc}, \text{brake}\}

motion: \dot{x}'' = a

André Platzer (CMU) Logical Foundations of Autonomous Cyber-Physical Systems MOD'19 7 / 21
Formal Verification in CPS Development

Real CPS

Verification Results

Proof
Reachability Analysis

safe

Challenge

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 $\alpha^*$

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

$\nu := \nu + 1$

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

$\dot{x} = \nu$

Proof

Reachability Analysis

Verification Results

Verification results about models only apply if CPS fits to the model; verifiably correct runtime model validation.

André Platzer (CMU)
Formal Verification in CPS Development

Real CPS

Model

α

Control

α

ctrl

v := v + 1

plant

x' = v

sense

act

safe

Challenge

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

Verifiably correct runtime model validation

Verification Results

Reachability Analysis

...
ModelPlex ensures that verification results about models apply to CPS implementations.
ModelPlex ensures that verification results about models apply to CPS implementations.

**Insights**

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

model adequate?  control safe?  until next cycle?
When are two states linked through a run of model $\alpha$?

A prior state characterized by $x^-$

Model $\alpha$

A posterior state characterized by $x^+$

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

reachability relation of $\alpha$
Characterizing State Relations in Logic

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

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

Model $\alpha$

Offline

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 $x = x^+$

FMSD’16
Characterizing State Relations in Logic

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

- **Semantical:** $(\omega, \nu) \in [\alpha]$
  \[ \iff \text{Lemma} \]
  \[ \exists \text{a run of } \alpha \text{ to a state where } x = x^+ \]

- **Logical $dL$:** $(\omega, \nu) \models \langle \alpha \rangle (x = x^+)$
  \[ \iff \text{dL proof} \]

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

---

a prior state characterized by $x^-$

Model $\alpha$

a posterior state characterized by $x^+$

Offline

FMSD‘16
Characterizing State Relations in Logic

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

- **Offline**
  - Semantical: $(\omega, \nu) \in [\alpha]$
  - Logical dL: $(\omega, \nu) \models \langle \alpha \rangle (x = x^+)$
  - Arithmetical: $(\omega, \nu) \models F(x^-, x^+)$

- Lemma: exists a run of $\alpha$ to a state where $x = x^+$
- dL proof: check at runtime (efficient)
Logic reduces CPS safety to runtime monitor with offline proof

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

\[ \omega \]

Model \( \alpha \)

\[ v \]

\[ \text{Init } \omega \in [A] \quad \text{Safe } v \in [S] \]

Semantical:

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

\[ \Downarrow \text{Lemma} \]

Logical dL:

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

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

Arithmetical:

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

check at runtime (efficient)
Logical Foundations for Model Safety Transfer

Logic reduces CPS safety to runtime monitor with offline proof

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

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

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

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

Logical dL proof:

check at runtime (efficient)
1 Autonomous Cyber-Physical Systems

2 Foundation: Differential Dynamic Logic

3 ModelPlex: Model Safety Transfer

4 VeriPhy: Executable Proof Transfer

5 Safe Learning in CPSs

6 Applications
   - Airborne Collision Avoidance System
   - Ground Robot Navigation

7 Summary
Approach: Safety Proofs for Autonomous CPS

KeYmaera X generates proofs

actions: \{acc, brake\}
motion: \(x'' = a\)

Autonomous CPS

Monitor transfers safety

ModelPlex proof synthesizes

Compliance Monitor

KeYmaera X

Model Safety

Proof and invariant search

Model Safety

André Platzer (CMU)

Logical Foundations of Autonomous Cyber-Physical Systems

MOD’19 11 / 21
VeriPhy: Automatic, Verified EXEs from Controllers

Hybrid Systems Theorem Proving

Abstract Controllers and Monitors

Sound Discrete Arithmetic

Sound Monitor Compilation

Cyber Physical System

Small Prover Core Proven Sound

Provably Correct Monitoring Conditions

Formalized Soundness Theorem

Verified Compiler

Verified Executable
VeriPhy: Automatic, Verified EXEs from Controllers

Hybrid Systems Theorem Proving

Abstract Controllers and Monitors

Sound Discrete Arithmetic

Sound Monitor Compilation

Cyber Physical System

Small Prover Core Proven Sound

Provably Correct Monitoring Conditions

Formalized Soundness Theorem

Verified Compiler

Verified Executable

KeYmaera X

Isabelle/HOL

HOL4
VeriPhy: Takeaway Metaphor

Your Model

Low-Level Proofs

Safe CPS
1 Autonomous Cyber-Physical Systems
2 Foundation: Differential Dynamic Logic
3 ModelPlex: Model Safety Transfer
4 VeriPhy: Executable Proof Transfer
5 Safe Learning in CPSs
6 Applications
   - Airborne Collision Avoidance System
   - Ground Robot Navigation
7 Summary
Approach: Safety Proofs for Autonomous CPS

KeYmaera X generates proofs

actions: \{\text{acc, brake}\}
motion: \dot{x}'' = a

Monitor transfers safety

ModelPlex proof synthesizes

Compliance Monitor

Model Safety

Proof and invariant search
Reinforcement Learning learns from experience of trying actions
RL chooses an action, observes outcome, reinforces in policy if successful
ModelPlex monitor inspects each decision, vetoes if unsafe
ModelPlex monitor gives early feedback about possible future problems. No need to wait till disaster strikes and propagate back.
Learning to Act Safely in a CPS

dL benefits from RL optimization.  
RL benefits from dL safety signal.

André Platzer (CMU)  
Logical Foundations of Autonomous Cyber-Physical Systems  
MOD’19  15 / 21
Learning to Act Safely in a CPS

Theorem
Safe policy if ODE accurate

Experiment
Graceful recovery outside ODE \(\leadsto\) quantitative ModelPlex
Detect modeled versus unmodeled state space \(\leadsto\) ModelPlex

AAA'I'18, ITC’18, TACAS’19, QEST’19
What’s safe when off model?
Learning to Act Safely in a CPS with Multiple Models

What’s safe with multiple possible models?

accel ∪ brake

observe
ModelPlex monitors conjunction of all plausible models
Learning to Act Safely in a CPS with Multiple Models

Remove incompatible models after contradictory observation

André Platzer (CMU)
Logical Foundations of Autonomous Cyber-Physical Systems

AAAI’18, ITC’18, TACAS’19, QEST’19
Learning to Act Safely in a CPS with Multiple Models

Plan differentiating experiment ⇐ predictive monitor distinctions
Learning to Act Safely in a CPS with Multiple Models

Plausible models converge to true model a.s., if possible
Modify model to fit observations by verification-preserving model update. Safety proofs reified: modify model + proof tactic to preserve fit + safety.
Outline

1. Autonomous Cyber-Physical Systems
2. Foundation: Differential Dynamic Logic
3. ModelPlex: Model Safety Transfer
4. VeriPhy: Executable Proof Transfer
5. Safe Learning in CPSs
6. Applications
   - Airborne Collision Avoidance System
   - Ground Robot Navigation
7. Summary
Developed by the FAA to replace current TCAS in aircraft
Approximately optimizes Markov Decision Process on a grid
Advisory from lookup tables with numerous 5D interpolation regions

Identified safe region for each advisory symbolically
Proved safety for hybrid systems flight model in KeYmaera X
ACAS X table comparison shows safe advisory in 97.7% of the 648,591,384,375 states compared (15,160,434,734 counterexamples).

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

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

Identified safeable region for each advisory symbolically

1. Proved safety for hybrid systems flight model in KeYmaera X
ACAS X table comparison shows safeable advisory in more of the 648,591,384,375 states compared ($\approx 899 \times 10^6$ counterexamples).

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

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

ACAS X issues Maintain advisory instead of CL1500
Fundamental safety question for ground robot navigation

When will which control decision avoid obstacles?

Depends on safety objective, physical capabilities of robot + obstacle

1. Identified safe region for each safety notion symbolically
2. Proved safety for hybrid systems ground robot model in KeYmaera X
Ground Robot Obstacle Avoidance: Verify

- Fundamental safety question for ground robot navigation
- When will which control decision avoid obstacles?
- Depends on safety objective, physical capabilities of robot + obstacle

1. Identified safe region for each safety notion symbolically
2. Proved safety for hybrid systems ground robot model in KeYmaera X
Ground Robot Obstacle Avoidance: Verify

- Fundamental safety question for ground robot navigation
- When will which control decision avoid obstacles?
- Depends on safety objective, physical capabilities of robot + obstacle

1. Identified safe region for each safety notion symbolically
2. Proved safety for hybrid systems ground robot model in KeYmaera X

André Platzer (CMU) Logical Foundations of Autonomous Cyber-Physical Systems MOD’19 18 / 21
Ground Robot Obstacle Avoidance: Verify

- Fundamental safety question for ground robot navigation
- When will which control decision avoid obstacles?
- Depends on safety objective, physical capabilities of robot + obstacle

<table>
<thead>
<tr>
<th>Acceleration</th>
<th>Speed</th>
<th>Time</th>
<th>Static</th>
<th>Passive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orientation</td>
<td>Safety</td>
<td>Orientation</td>
<td>Parking</td>
<td>Avoid/Follow</td>
</tr>
<tr>
<td>Passive</td>
<td>Static</td>
<td>Orientation</td>
<td>Parking</td>
<td>Avoid/Follow</td>
</tr>
</tbody>
</table>

1. Identified safe region for each safety notion symbolically
2. Proved safety for hybrid systems ground robot model in KeYmaera X
Ground Robot Obstacle Avoidance: Verify

- Fundamental safety question for ground robot navigation
- When will which control decision avoid obstacles?
- Depends on safety objective, physical capabilities of robot + obstacle

Identified safe region for each safety notion symbolically

Proved safety for hybrid systems ground robot model in KeYmaera X
<table>
<thead>
<tr>
<th>Safety</th>
<th>Invariant + Safe Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>[ |p - o|_\infty &gt; \frac{s^2}{2b} + \left( \frac{A}{b} + 1 \right) \left( \frac{A}{2} \epsilon^2 + \epsilon s \right) ]</td>
</tr>
<tr>
<td>passive</td>
<td>[ s \neq 0 \rightarrow |p - o|_\infty &gt; \frac{s^2}{2b} + V \frac{s}{b} + \left( \frac{A}{b} + 1 \right) \left( \frac{A}{2} \epsilon^2 + \epsilon (s + V) \right) ]</td>
</tr>
<tr>
<td>sensor</td>
<td>[ |\hat{p} - o|_\infty &gt; \frac{s^2}{2b} + V \frac{s}{b} + \left( \frac{A}{b} + 1 \right) \left( \frac{A}{2} \epsilon^2 + \epsilon (s + V) \right) + \Delta_p ]</td>
</tr>
<tr>
<td>disturb.</td>
<td>[ |p - o|_\infty &gt; \frac{s^2}{2b \Delta_a} + V \frac{s}{b \Delta_a} + \left( \frac{A}{b \Delta_a} + 1 \right) \left( \frac{A}{2} \epsilon^2 + \epsilon (s + V) \right) ]</td>
</tr>
<tr>
<td>failure</td>
<td>[ |\hat{p} - o|_\infty &gt; \frac{s^2}{2b} + V \frac{s}{b} + \left( \frac{A}{b} + 1 \right) \left( \frac{A}{2} \epsilon^2 + \epsilon (v + V) \right) + \Delta_p + g \Delta ]</td>
</tr>
<tr>
<td>friendly</td>
<td>[ |p - o|_\infty &gt; \frac{s^2}{2b} + \frac{V^2}{2b_o} + V \left( \frac{s}{b} + \tau \right) + \left( \frac{A}{b} + 1 \right) \left( \frac{A}{2} \epsilon^2 + \epsilon (s + V) \right) ]</td>
</tr>
<tr>
<td>Safety</td>
<td>Invariant</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>static</td>
<td>$|p - o|_\infty &gt; \frac{s^2}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2} \varepsilon^2 + \varepsilon s\right)$</td>
</tr>
<tr>
<td>passive</td>
<td>$s \neq 0 \rightarrow |p - o|_\infty &gt; \frac{s^2}{2b} + V \frac{s}{b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2} \varepsilon^2 + \varepsilon(s + V)\right)$</td>
</tr>
<tr>
<td>+ sensor</td>
<td>$|p - o|_\infty &gt; \frac{s^2}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2} \varepsilon^2 + \varepsilon(s + V)\right) + \Delta_p$</td>
</tr>
<tr>
<td>+ disturb.</td>
<td>$|p - o|_\infty &gt; \frac{s^2}{2b} + V \frac{s}{b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2} \varepsilon^2 + \varepsilon(s + V)\right)$</td>
</tr>
<tr>
<td>+ failure</td>
<td>$|\hat{p} - o|_\infty &gt; \frac{s^2}{2b} + V \frac{s}{b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2} \varepsilon^2 + \varepsilon(v + V)\right) + \Delta_p + g\Delta$</td>
</tr>
<tr>
<td>friendly</td>
<td>$|p - o|_\infty &gt; \frac{s^2}{2b} + \frac{V^2}{2b_0} + V\left(\frac{s}{b} + \tau\right) + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2} \varepsilon^2 + \varepsilon(s + V)\right)$</td>
</tr>
</tbody>
</table>

**Question**

How to find and justify constraints? Proof!
Autonomous Cyber-Physical Systems

Foundation: Differential Dynamic Logic

ModelPlex: Model Safety Transfer

VeriPhy: Executable Proof Transfer

Safe Learning in CPSs

Applications
- Airborne Collision Avoidance System
- Ground Robot Navigation

Summary
Acknowledgments

Logical Systems Lab at Carnegie Mellon University, Computer Science
Yong Kiam Tan, Brandon Bohrer, Nathan Fulton, Sarah Loos, Katherine Cordwell
Stefan Mitsch, Khalil Ghorbal, Jean-Baptiste Jeannin, Andrew Sogokon
differential dynamic logic

\[ dL = DL + HP \]

Logical Triumvirate of Technologies for Transitioning Trustworthiness

1. KeYmaera X: safe action in CPS model
2. ModelPlex: safe model \( \leadsto \) safe impl
3. VeriPhy: sandbox \( \leadsto \) safe executable

1. RL optimizes action choice
2. ModelPlex: safe reward for RL
3. VeriPhy: CPS sandbox for RL
I Part: Elementary Cyber-Physical Systems
2. Differential Equations & Domains
3. Choice & Control
4. Safety & Contracts
5. Dynamical Systems & Dynamic Axioms
6. Truth & Proof
7. Control Loops & Invariants
8. Events & Responses
9. Reactions & Delays

II Part: Differential Equations Analysis
10. Differential Equations & Differential Invariants
11. Differential Equations & Proofs
12. Ghosts & Differential Ghosts
13. Differential Invariants & Proof Theory

III Part: Adversarial Cyber-Physical Systems
14-17. Hybrid Systems & Hybrid Games

IV Part: Comprehensive CPS Correctness
### Definition (Hybrid program $\alpha$)

$$
\begin{align*}
x &:= f(x) \mid ?Q \mid x' = f(x) \& Q \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^*
\end{align*}
$$

### Definition (dL Formula $P$)

$$
\begin{align*}
e &\geq \tilde{e} \mid \neg P \mid P \land Q \mid \forall x \ P \mid \exists x \ P \mid [\alpha]P \mid \langle \alpha \rangle P
\end{align*}
$$
Differential Dynamic Logic $dL$: Syntax

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

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

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

$$e \geq \tilde{e} | \neg P | P \land Q | \forall x P | \exists x P | [\alpha] P | \langle \alpha \rangle P$$
### Differential Dynamic Logic $dL$: Semantics

**Definition (Hybrid program semantics) ($[[\cdot]] : \text{HP} \rightarrow \mathcal{P}(S \times S)$)**

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

**Definition (dL semantics) ($[[\cdot]] : \text{Fml} \rightarrow \mathcal{P}(S)$)**

- $[[e \geq \bar{e}]] = \{ \omega : \omega[e] \geq \omega[\bar{e}] \}$
- $[[\neg P]] = [[P]]^c$
- $[[P \land Q]] = [[P]] \cap [[Q]]$
- $[[\langle \alpha \rangle P]] = [[\alpha]] \circ [[P]] = \{ \omega : \nu \in [[P]] \text{ for some } \nu : (\omega, \nu) \in [[\alpha]] \}$
- $[[[\alpha]P]] = [[\neg \langle \alpha \rangle \neg P]] = \{ \omega : \nu \in [[P]] \text{ for all } \nu : (\omega, \nu) \in [[\alpha]] \}$
- $[[\exists x P]] = \{ \omega : \omega'_x \in [[P]] \text{ for some } r \in \mathbb{R} \}$
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

\[ x' = f(x) \&E Q \]
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

\[ x' = f(x) \]

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

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

\[ x' = f(x) \& Q \]
### Differential Invariants for Differential Equations

<table>
<thead>
<tr>
<th>Differential Invariant</th>
<th>Differential Cut</th>
<th>Differential Ghost</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Invarianter Image" /></td>
<td><img src="image2.png" alt="Cut Image" /></td>
<td><img src="image3.png" alt="Ghost Image" /></td>
</tr>
</tbody>
</table>

The differential invariant, cut, and ghost are visualized in the diagrams above, where $x'$ and $y'$ are related by differential equations:

1. $x' = f(x)$ & $Q$
2. $x' = f(x)$ & $Q$
3. $x' = f(x)$ & $Q$

The images illustrate the concept of differential invariants and their corresponding cuts and ghosts in the context of differential equations.
Differential Invariants for Differential Equations

Differential Invariant

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

Differential Cut

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

Differential Ghost

\[ x' = f(x) \land Q \]
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

\[ \begin{align*}
  x' &= f(x) & Q \\
  x' &= f(x) & Q \\
  x' &= f(x) & Q
\end{align*} \]
Differential Invariants for Differential Equations

Differential Invariant

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

Differential Cut

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

Differential Ghost

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

Differential Invariant

Differential Cut

Differential Ghost

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

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

\[ x' = f(x) \& Q \]
### Differential Invariants for Differential Equations

<table>
<thead>
<tr>
<th>Differential Invariant</th>
<th>Differential Cut</th>
<th>Differential Ghost</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Differential Invariant Diagram" /></td>
<td><img src="image2.png" alt="Differential Cut Diagram" /></td>
<td><img src="image3.png" alt="Differential Ghost Diagram" /></td>
</tr>
</tbody>
</table>

The diagrams illustrate the concepts of differential invariants, cuts, and ghosts in the context of differential equations.

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

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

\[
x = 0, t
\]

\[
x' = f(x)
\]

\[
x = 0, s
\]

\[
x' = f(x)
\]

\[
x = 0, t
\]
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

\[ x' = f(x) \]

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

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

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

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

**Differential Invariant**

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

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

**Differential Cut**

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

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

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

**Differential Ghost**

\[
P \leftrightarrow \exists y G
\]

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

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

**deductive power added**

\[
\text{DI} \prec \text{DI+DC} \prec \text{DI+DC+DG}
\]

\[
\omega[(e)'] = \sum_x \omega(x') \frac{\partial [e]}{\partial x}(\omega)
\]
### Differential Invariant

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

### Differential Cut

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

### Differential Ghost

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

if \( g(x, y) = a(x)y + b(x) \), so has long solution!
André Platzer. 
*Logical Foundations of Cyber-Physical Systems.* 
Springer, Cham, 2018. 
doi:10.1007/978-3-319-63588-0.

André Platzer. 
The logical path to autonomous cyber-physical systems. 

André Platzer. 
Differential dynamic logic for hybrid systems. 

André Platzer. 
Logics of dynamical systems. 
In *LICS,* pages 13–24, Los Alamitos, 2012. IEEE. 
André Platzer.
A complete uniform substitution calculus for differential dynamic logic.

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

Nathan Fulton and André Platzer.
Safe reinforcement learning via formal methods: Toward safe control through proof and learning.

Nathan Fulton and André Platzer.
Verifiably safe off-model reinforcement learning.


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

André Platzer.
The structure of differential invariants and differential cut elimination.