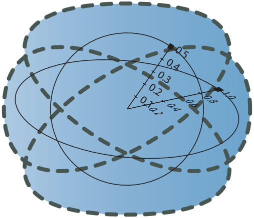
Logical Foundations of Cyber-Physical Systems

André Platzer

aplatzer@cs.cmu.edu
Logical Systems Lab
Computer Science Department
Carnegie Mellon University, Pittsburgh, PA

http://symbolaris.com/



DARPA

Outline

1. CPS are Multi-Dynamical Systems
   - Hybrid Systems
   - Hybrid Games
2. Dynamic Logic for Multi-Dynamical Systems
   - Syntax
   - Semantics
3. Proofs for CPS
4. Theory of CPS
   - Soundness and Completeness
   - Differential Invariants
   - Differential Radical Invariants
5. Applications
   - Ground Robots
6. Summary

Outline

Can you trust a computer to control physics?

Can you trust a computer to control physics?

Rationale

1. Safety guarantees require analytic foundations
2. Foundations revolutionized digital computer science & society
3. Need even stronger foundations when software reaches out into our physical world

Can you trust a computer to control physics?

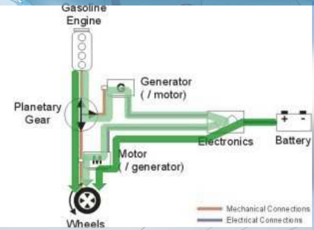
Rationale

1. Safety guarantees require analytic foundations
2. Foundations revolutionized digital computer science & society
3. Need even stronger foundations when software reaches out into our physical world

CPS Core Question

How can we provide people with cyber-physical systems they can bet their lives on?

CPS Applications



Correctness Questions in CPS Design

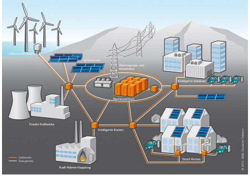
Safety The system must be safe under all circumstances

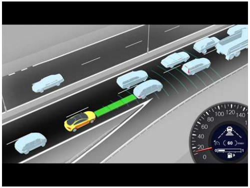
Liveness The system must reach a given goal

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 V & V

Proving

Safety	Formalize system properties: What is "Safe"? "Reach goal"?
Models	Formalize system models
Assumptions	Make assumptions explicit
Constraints	Reveal invariants, switching conditions, starting conditions
Design	Invariants guide safe controller design
Constructive	Construct models along with their proof

Byproducts

Analyze	Determine design trade-offs & feasibility early
Synthesize	Turn high-level models into code & correctness monitors
Certify	Proofs as artifacts for certification

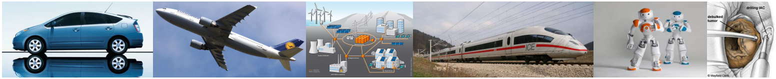
Tools

KeYmaera	Theorem prover for CPS

Successful CPS Proofs

Diverse Application Domains

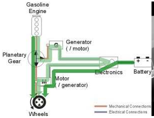
- Automotive
- Aircraft
- Railway
- Energy
- Robotics
- Surgery



Various Levels

All the way from

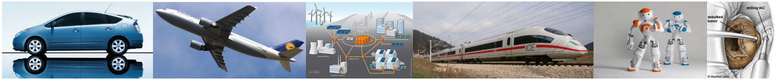
- Engine idle control
- Adaptive cruise control
- Highway traffic control



Successful CPS Proofs

Diverse Application Domains

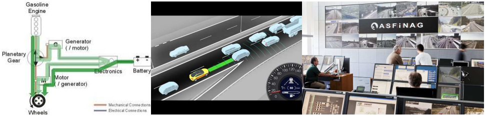
- Automotive
- Aircraft
- Railway
- Energy
- Robotics
- Surgery



Various Levels

All the way from

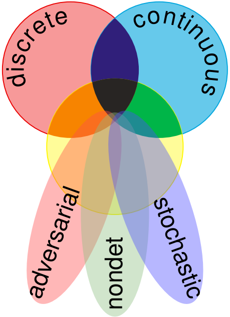
- Engine idle control
- Adaptive cruise control
- Highway traffic control



CPS are Multi-Dynamical Systems

CPS Dynamics

CPS are characterized by multiple facets of dynamical systems.



CPS Compositions

CPS combine multiple simple dynamical effects.

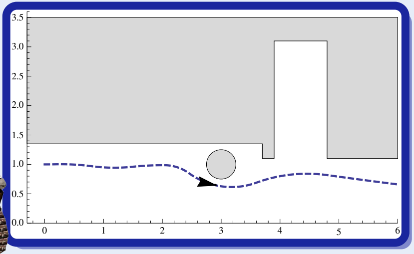
Tame Parts

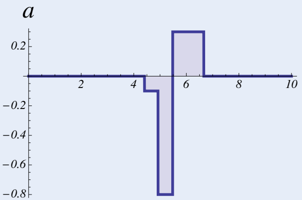
Exploiting compositionality tames complexity.

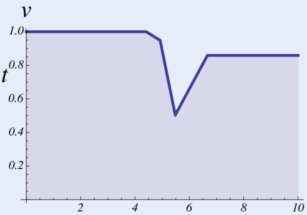
Challenge (Hybrid Systems)

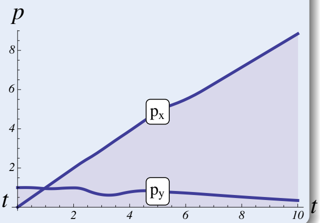
Fixed rule describing state evolution with both

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





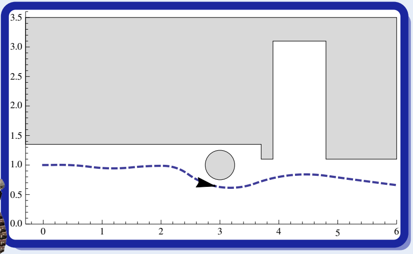


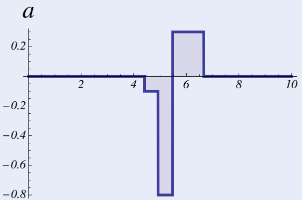


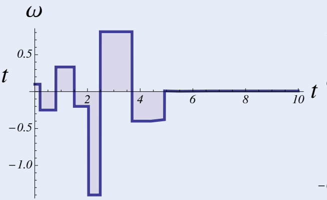
Challenge (Hybrid Systems)

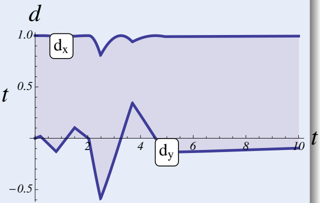
Fixed rule describing state evolution with both

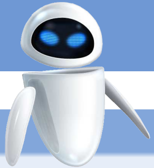
- Discrete dynamics (control decisions)
- Continuous dynamics (differential equations)







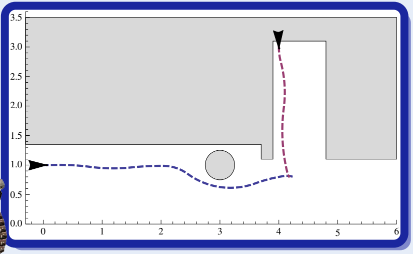


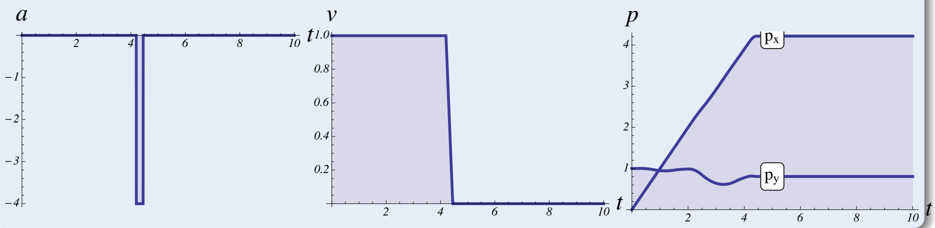


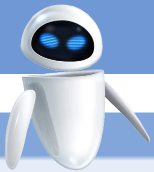
Challenge (Hybrid Systems)

Fixed rule describing state evolution with both

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



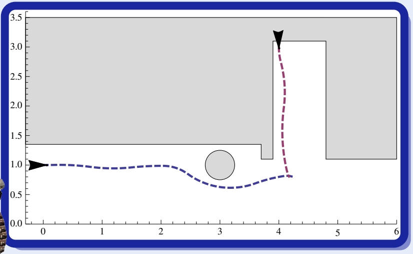


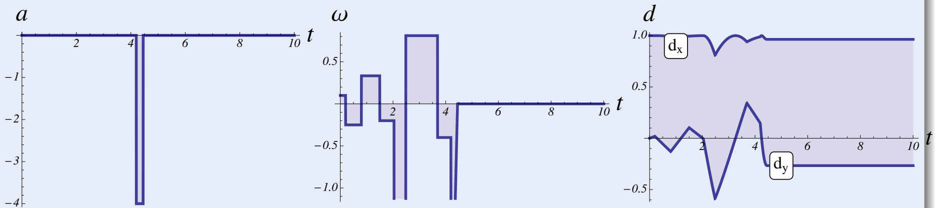


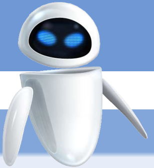
Challenge (Hybrid Systems)

Fixed rule describing state evolution with both

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



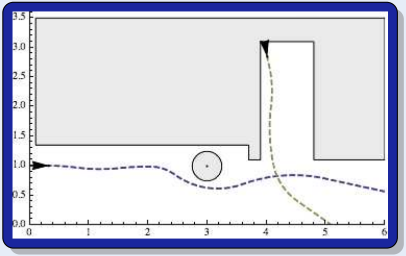


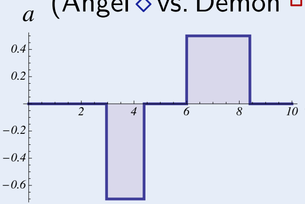


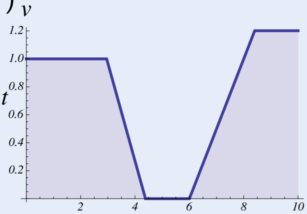
Challenge (Hybrid Games)

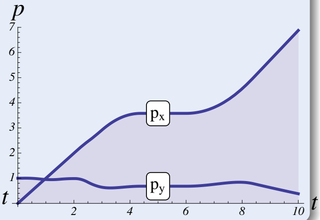
Game rules describing play evolution with

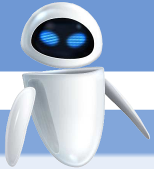
- Discrete dynamics (control decisions)
- Continuous dynamics (differential equations)
- Adversarial dynamics (Angel ◇ vs. Demon □)







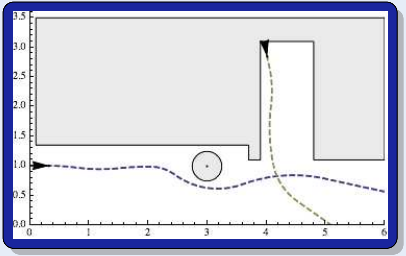


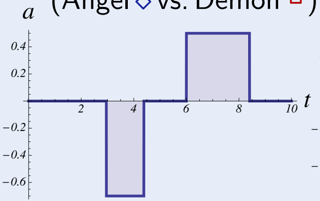


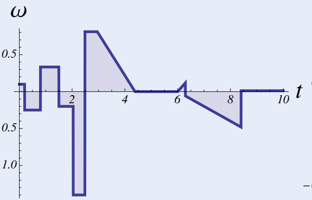
Challenge (Hybrid Games)

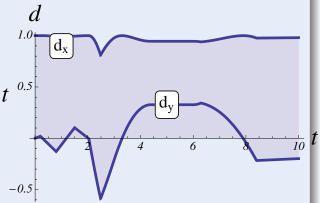
Game rules describing play evolution with

- Discrete dynamics (control decisions)
- Continuous dynamics (differential equations)
- Adversarial dynamics (Angel $\Diamond$ vs. Demon $\Box$)







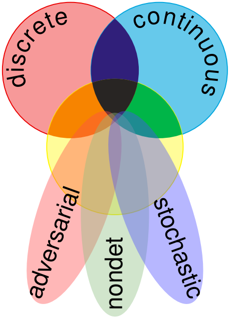


Outline

CPS are Multi-Dynamical Systems

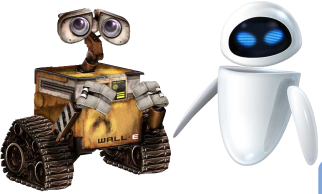
hybrid systems

HS = discrete + ODE



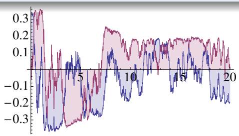
hybrid games

HG = HS + adversary



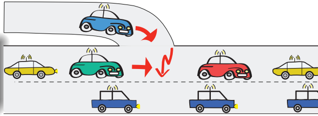
stochastic hybrid sys.

SHS = HS + stochastics

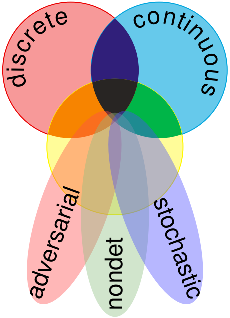


distributed hybrid sys.

DHS = HS + distributed



Family of Differential Dynamic Logics

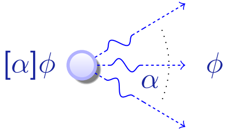


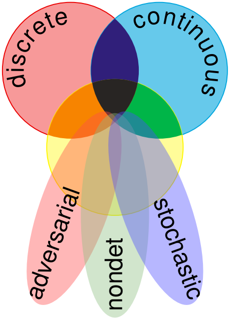
JAR'08,CADE'11,LMCS'12,LICS'12,LICS'12

Family of Differential Dynamic Logics

differential dynamic logic

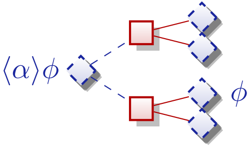
$d\mathcal{L} = \text{DL} + \text{HP}$





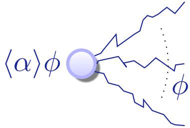
differential game logic

$dG\mathcal{L} = \text{GL} + \text{HG}$



stochastic differential DL

$Sd\mathcal{L} = \text{DL} + \text{SHP}$

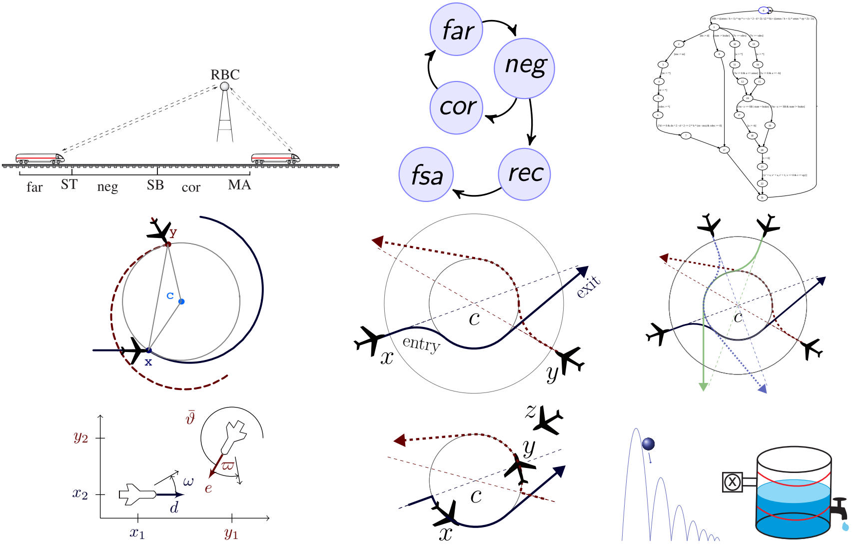


quantified differential DL

$Qd\mathcal{L} = \text{FOL} + \text{DL} + \text{QHP}$

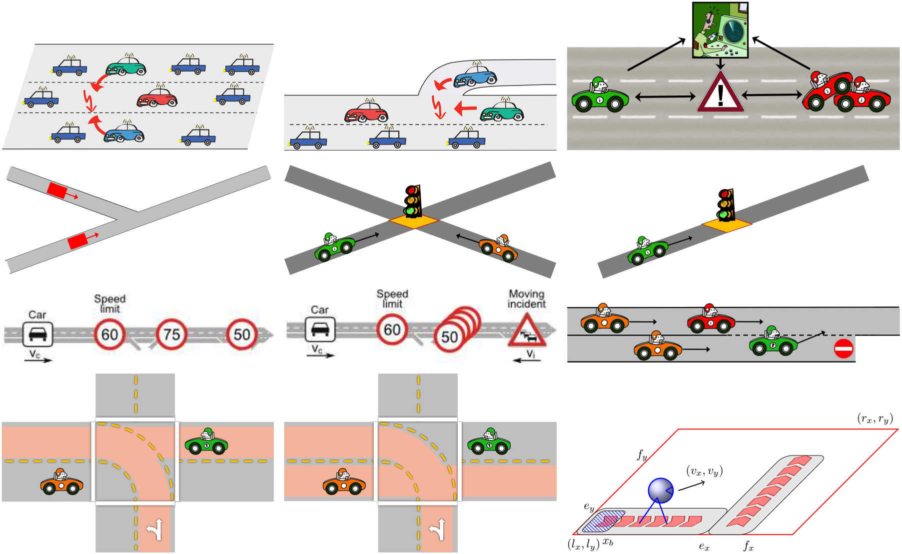
JAR'08,CADE'11,LMCS'12,LICS'12,LICS'12

Successful CPS Proofs



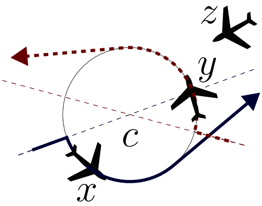
ICFEM'09,CAV'08,FM'09,HSCC'11

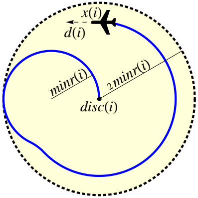
Successful CPS Proofs

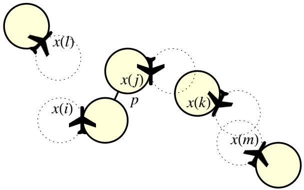


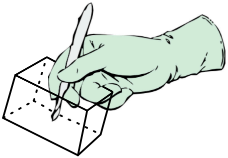
FM'11,LMCS'12,ICCPS'12,ITSC'11,ITSC'13,IJCAR'12

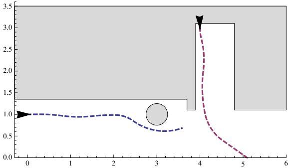
Successful CPS Proofs

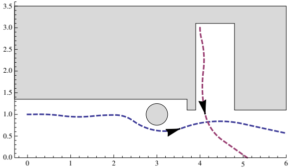


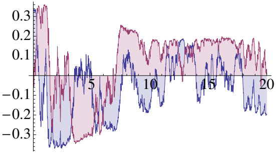


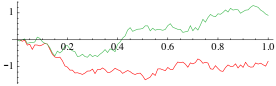


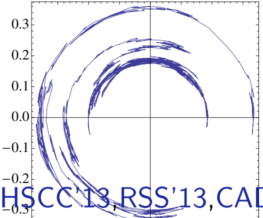






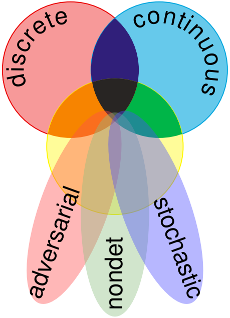






HSCC'11,HSCC'13,HSCC'13,RSS'13,CADE'12

Family of Differential Dynamic Logics

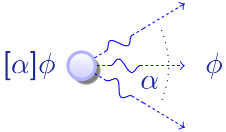


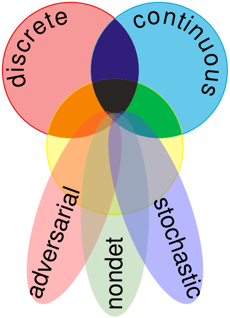
JAR'08,CADE'11,LMCS'12,LICS'12,LICS'12

Family of Differential Dynamic Logics

differential dynamic logic

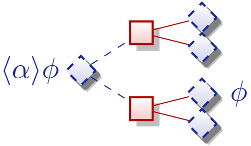
$d\mathcal{L} = \mathsf{DL} + \mathsf{HP}$





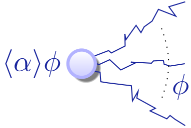
differential game logic

$dG\mathcal{L} = \mathsf{GL} + \mathsf{HG}$



stochastic differential DL

$Sd\mathcal{L} = \mathsf{DL} + \mathsf{SHP}$



quantified differential DL

$Qd\mathcal{L} = \mathsf{FOL} + \mathsf{DL} + \mathsf{QHP}$

JAR'08,CADE'11,LMCS'12,LICS'12,LICS'12

Differential Dynamic Logic d$\mathcal{L}$: Syntax

Definition (Hybrid program α)

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

Definition (d$\mathcal{L}$ Formula ϕ)

$$\theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \forall x\, \phi \mid \exists x\, \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi$$

Differential Dynamic Logic dℒ: Syntax

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

Definition (Hybrid program α)

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

Definition (dℒ Formula ϕ)

$$\theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \forall x\, \phi \mid \exists x\, \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi$$

All Reals | Some Reals | All Runs | Some Runs

Differential Dynamic Logic d$\mathcal{L}$: Semantics

Definition (Hybrid program α)

$$
\begin{aligned}
\rho(x := \theta) &= \{(v, w) \ : \ w = v \text{ except } [\![x]\!]_w = [\![\theta]\!]_v\} \\
\rho(?H) &= \{(v, v) \ : \ v \models H\} \\
\rho(x' = f(x)) &= \{(\varphi(0), \varphi(r)) \ : \ \varphi \models x' = f(x) \text{ for some duration } r\} \\
\rho(\alpha \cup \beta) &= \rho(\alpha) \cup \rho(\beta) \\
\rho(\alpha; \beta) &= \rho(\beta) \circ \rho(\alpha) \\
\rho(\alpha^*) &= \bigcup_{n \in \mathbb{N}} \rho(\alpha^n)
\end{aligned}
$$

Definition (d$\mathcal{L}$ Formula ϕ)

$$
\begin{array}{lll}
v \models \theta_1 \geq \theta_2 & \text{iff} & [\![\theta_1]\!]_v \geq [\![\theta_2]\!]_v \\
v \models [\alpha]\phi & \text{iff} & w \models \phi \text{ for all } w \text{ with } v\rho(\alpha)w \\
v \models \langle\alpha\rangle\phi & \text{iff} & w \models \phi \text{ for some } w \text{ with } v\rho(\alpha)w \\
v \models \forall x\, \phi & \text{iff} & w \models \phi \text{ for all } w \text{ that agree with } v \text{ except for } x \\
v \models \exists x\, \phi & \text{iff} & w \models \phi \text{ for some } w \text{ that agrees with } v \text{ except for } x \\
v \models \phi \wedge \psi & \text{iff} & v \models \phi \text{ and } v \models \psi
\end{array}
$$

Outline

Differential Dynamic Logic: Axiomatization

equations of truth

$$[:=] \quad [x := \theta]\phi(x) \leftrightarrow \phi(\theta)$$

$$[?] \quad [?H]\phi \leftrightarrow (H \to \phi)$$

$$['] \quad [x' = f(x)]\phi \leftrightarrow \forall t{\geq}0\,[x := y(t)]\phi \qquad (y'(t) = f(y))$$

$$[\cup] \quad [\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \land [\beta]\phi$$

$$[;] \quad [\alpha;\beta]\phi \leftrightarrow [\alpha][\beta]\phi$$

$$[^*] \quad [\alpha^*]\phi \leftrightarrow \phi \land [\alpha][\alpha^*]\phi$$

$$\mathrm{K} \quad [\alpha](\phi \to \psi) \to ([\alpha]\phi \to [\alpha]\psi)$$

$$\mathrm{I} \quad [\alpha^*](\phi \to [\alpha]\phi) \to (\phi \to [\alpha^*]\phi)$$

$$\mathrm{C} \quad [\alpha^*]\forall v{>}0\,(\varphi(v) \to \langle\alpha\rangle\varphi(v-1)) \to \forall v\,(\varphi(v) \to \langle\alpha^*\rangle\exists v{\leq}0\,\varphi(v))$$

LICS'12

Differential Dynamic Logic: Axiomatization

equations of truth

G $\dfrac{\phi}{[\alpha]\phi}$

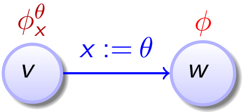
MP $\dfrac{\phi \rightarrow \psi \quad \phi}{\psi}$

$\forall$ $\dfrac{\phi}{\forall x\, \phi}$

LICS'12

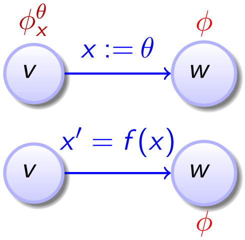
Proofs for Hybrid Systems

$$\frac{\phi_x^\theta}{[x := \theta]\phi}$$



$$\frac{\phi_x^\theta}{[x := \theta]\phi}$$

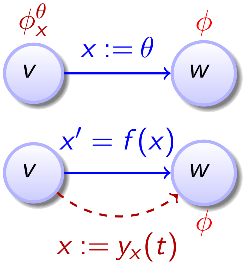
$$\frac{\forall t{\geq}0\, [x := y_x(t)]\phi}{[x' = f(x)]\phi}$$



Proofs for Hybrid Systems

$$\frac{\phi_x^\theta}{[x := \theta]\phi}$$

$$\frac{\forall t{\geq}0\, [x := y_x(t)]\phi}{[x' = f(x)]\phi}$$

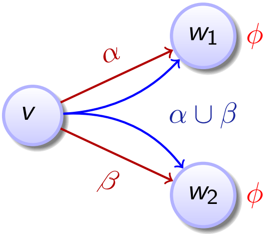


Proofs for Hybrid Systems

compositional semantics $\Rightarrow$ compositional rules!

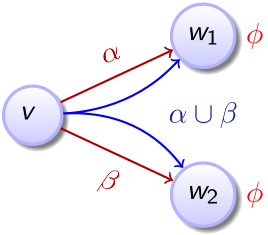
Proofs for Hybrid Systems

$$\frac{[\alpha]\phi \land [\beta]\phi}{[\alpha \cup \beta]\phi}$$

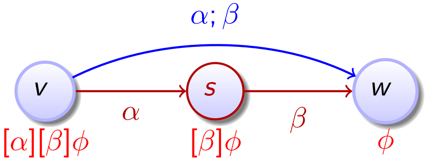


Proofs for Hybrid Systems

$$\frac{[\alpha]\phi \land [\beta]\phi}{[\alpha \cup \beta]\phi}$$

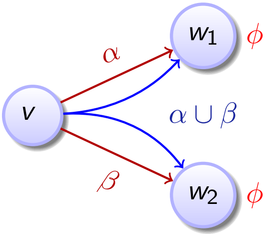


$$\frac{[\alpha][\beta]\phi}{[\alpha; \beta]\phi}$$

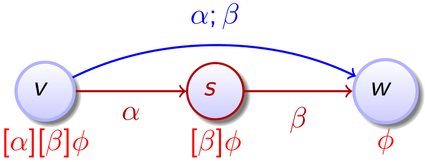


Proofs for Hybrid Systems

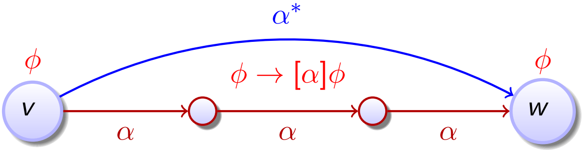
$$\frac{[\alpha]\phi \land [\beta]\phi}{[\alpha \cup \beta]\phi}$$



$$\frac{[\alpha][\beta]\phi}{[\alpha;\beta]\phi}$$



$$\frac{\phi \quad (\phi \to [\alpha]\phi)}{[\alpha^*]\phi}$$



Outline

1. CPS are Multi-Dynamical Systems
   - Hybrid Systems
   - Hybrid Games
2. Dynamic Logic for Multi-Dynamical Systems
   - Syntax
   - Semantics
3. Proofs for CPS
4. Theory of CPS
   - Soundness and Completeness
   - Differential Invariants
   - Differential Radical Invariants
5. Applications
   - Ground Robots
6. Summary

Complete Proof Theory of Hybrid Systems

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.* ▸ Proof 25pp

Corollary (Complete Proof-theoretical Alignment & Bridging)

proving continuous = proving hybrid = proving discrete

JAutomReas'08,LICS'12

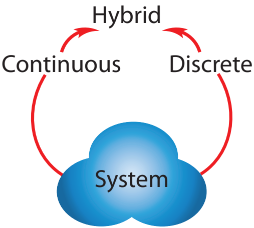
Complete Proof Theory of Hybrid Systems

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.* ▸ Proof 25pp

Corollary (Complete Proof-theoretical Alignment & Bridging)

proving continuous = proving hybrid = proving discrete



JAutomReas'08,LICS'12

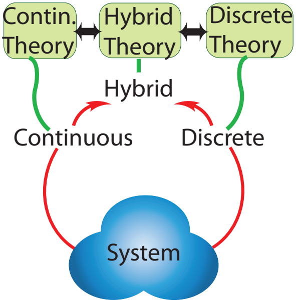
Complete Proof Theory of Hybrid Systems

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.* ▸ Proof 25pp

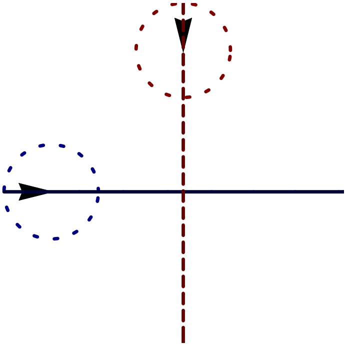
Corollary (Complete Proof-theoretical Alignment & Bridging)

proving continuous = proving hybrid = proving discrete

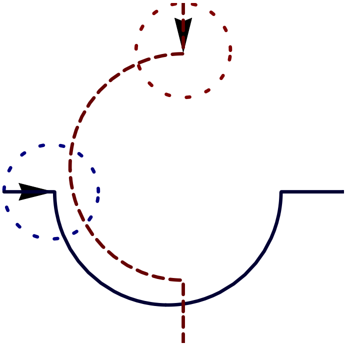


JAutomReas'08,LICS'12

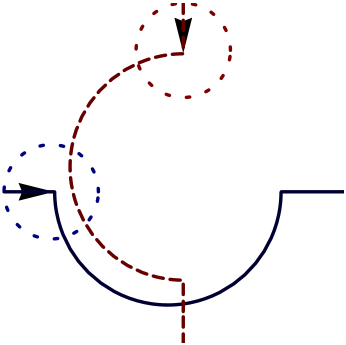
Air Traffic Control



Air Traffic Control



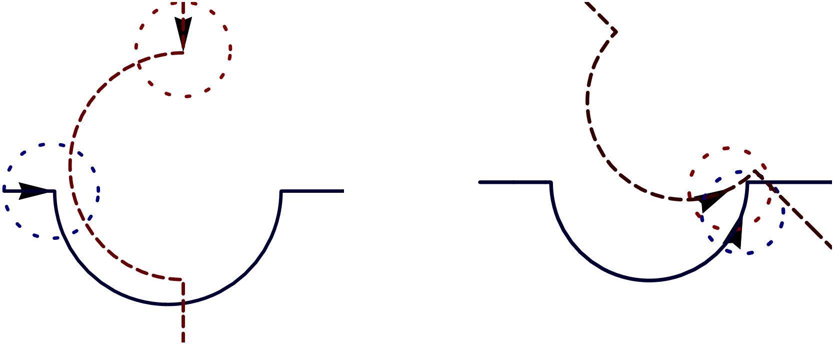
Air Traffic Control



Verification?

looks correct

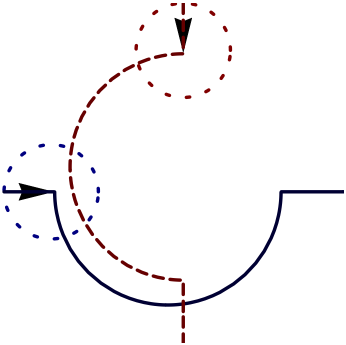
Air Traffic Control

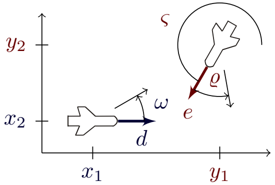


Verification?

looks correct NO!

Air Traffic Control



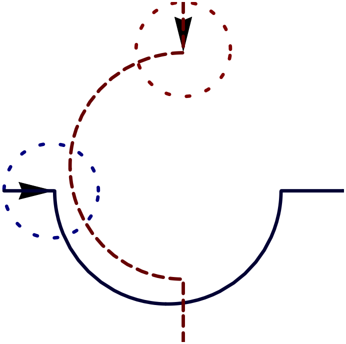


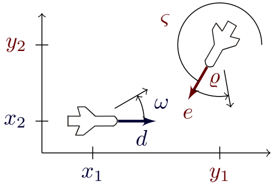
$$\begin{bmatrix} x_1' = -v_1 + v_2 \cos\vartheta + \omega x_2 \\ x_2' = v_2 \sin\vartheta - \omega x_1 \\ \vartheta' = \varpi - \omega \end{bmatrix}$$

Verification?

looks correct NO!

Air Traffic Control

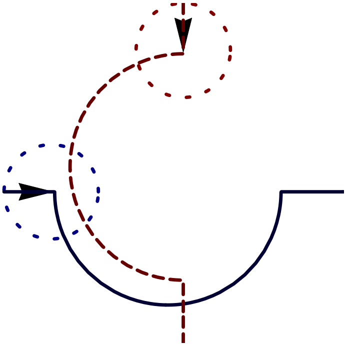


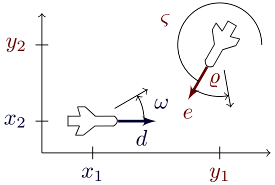


$$\begin{bmatrix} x_1' = -v_1 + v_2 \cos\vartheta + \omega x_2 \\ x_2' = v_2 \sin\vartheta - \omega x_1 \\ \vartheta' = \varpi - \omega \end{bmatrix}$$

Example (“Solving” differential equations)

$$x_1(t) = \frac{1}{\omega\varpi}\big(x_1\omega\varpi\cos t\omega - v_2\omega\cos t\omega\sin\vartheta + v_2\omega\cos t\omega\cos t\varpi\sin\vartheta - v_1\varpi\sin t\omega$$
$$+ x_2\omega\varpi\sin t\omega - v_2\omega\cos\vartheta\cos t\varpi\sin t\omega - v_2\omega\sqrt{1 - \sin\vartheta^2}\sin t\omega$$
$$+ v_2\omega\cos\vartheta\cos t\omega\sin t\varpi + v_2\omega\sin\vartheta\sin t\omega\sin t\varpi\big)\ldots$$





$$\begin{bmatrix} x_1' = -v_1 + v_2 \cos\vartheta + \omega x_2 \\ x_2' = v_2 \sin\vartheta - \omega x_1 \\ \vartheta' = \varpi - \omega \end{bmatrix}$$

Example ("Solving" differential equations)

$$\forall t \geq 0 \quad \frac{1}{\omega\varpi}\Big(x_1\omega\varpi \cos t\omega - v_2\omega \cos t\omega \sin\vartheta + v_2\omega \cos t\omega \cos t\varpi \sin\vartheta - v_1\varpi \sin t\omega$$
$$+ x_2\omega\varpi \sin t\omega - v_2\omega \cos\vartheta \cos t\varpi \sin t\omega - v_2\omega\sqrt{1 - \sin\vartheta^2} \sin t\omega$$
$$+ v_2\omega \cos\vartheta \cos t\omega \sin t\varpi + v_2\omega \sin\vartheta \sin t\omega \sin t\varpi\Big) \ldots$$

```
\forall R ts2.
  (   0 <= ts2 & ts2 <= t2_0
   ->       (  (om_1)^-1
             * (omb_1)^-1
             * (  om_1 * omb_1 * x1 * Cos(om_1 * ts2)
                + om_1 * v2 * Cos(om_1 * ts2) * (1 + -1 * (Cos(u))^2)^(1 / 2)
                + -1 * omb_1 * v1 * Sin(om_1 * ts2)
                + om_1 * omb_1 * x2 * Sin(om_1 * ts2)
                + om_1 * v2 * Cos(u) * Sin(om_1 * ts2)
                + -1 * om_1 * v2 * Cos(omb_1 * ts2) * Cos(u) * Sin(om_1 * ts2)
                + om_1 * v2 * Cos(om_1 * ts2) * Cos(u) * Sin(omb_1 * ts2)
                + om_1 * v2 * Cos(om_1 * ts2) * Cos(omb_1 * ts2) * Sin(u)
                + om_1 * v2 * Sin(om_1 * ts2) * Sin(omb_1 * ts2) * Sin(u)))
        ^2
       +    (  (om_1)^-1
             * (omb_1)^-1
             * (  -1 * omb_1 * v1 * Cos(om_1 * ts2)
                + om_1 * omb_1 * x2 * Cos(om_1 * ts2)
                + omb_1 * v1 * (Cos(om_1 * ts2))^2
                + om_1 * v2 * Cos(om_1 * ts2) * Cos(u)
                + -1 * om_1 * v2 * Cos(om_1 * ts2) * Cos(omb_1 * ts2) * Cos(u)
                + -1 * om_1 * omb_1 * x1 * Sin(om_1 * ts2)
                +   -1
                  * om_1
                  * v2
                  * (1 + -1 * (Cos(u))^2)^(1 / 2)
                  * Sin(om_1 * ts2)
                + omb_1 * v1 * (Sin(om_1 * ts2))^2
                + -1 * om_1 * v2 * Cos(u) * Sin(om_1 * ts2) * Sin(omb_1 * ts2)
                + -1 * om_1 * v2 * Cos(omb_1 * ts2) * Sin(om_1 * ts2) * Sin(u)
                + om_1 * v2 * Cos(om_1 * ts2) * Sin(omb_1 * ts2) * Sin(u)))
        ^2
      >= (p)^2),
 t2_0 >= 0,
 x1^2 + x2^2 >= (p)^2
==>
```

```
\forall R t7.
  (   t7 >= 0
   ->       (   (om_3)^-1
             * (    om_3
                 * (   (om_1)^-1
                    * (omb_1)^-1
                    * (   om_1 * omb_1 * x1 * Cos(om_1 * t2_0)
                       +   om_1
                         * v2
                         * Cos(om_1 * t2_0)
                         * (1 + -1 * (Cos(u))^2)^(1 / 2)
                       + -1 * omb_1 * v1 * Sin(om_1 * t2_0)
                       + om_1 * omb_1 * x2 * Sin(om_1 * t2_0)
                       + om_1 * v2 * Cos(u) * Sin(om_1 * t2_0)
                       +   -1
                         * om_1
                         * v2
                         * Cos(omb_1 * t2_0)
                         * Cos(u)
                         * Sin(om_1 * t2_0)
                       +   om_1
                         * v2
                         * Cos(om_1 * t2_0)
                         * Cos(u)
                         * Sin(omb_1 * t2_0)
                       +   om_1
                         * v2
                         * Cos(om_1 * t2_0)
                         * Cos(omb_1 * t2_0)
                         * Sin(u)
                       +   om_1
                         * v2
                         * Sin(om_1 * t2_0)
                         * Sin(omb_1 * t2_0)
                         * Sin(u)))
```

```
    * Cos(om_3 * t5)
  +   v2
    * Cos(om_3 * t5)
    *   (  1
         +   -1
           * (Cos(-1 * om_1 * t2_0 + omb_1 * t2_0 + u + Pi / 4))^2)
     ^(1 / 2)
  + -1 * v1 * Sin(om_3 * t5)
  +   om_3
    * (  (om_1)^-1
       * (omb_1)^-1
       * (  -1 * omb_1 * v1 * Cos(om_1 * t2_0)
          + om_1 * omb_1 * x2 * Cos(om_1 * t2_0)
          + omb_1 * v1 * (Cos(om_1 * t2_0))^2
          + om_1 * v2 * Cos(om_1 * t2_0) * Cos(u)
          +   -1
            * om_1
            * v2
            * Cos(om_1 * t2_0)
            * Cos(omb_1 * t2_0)
            * Cos(u)
          + -1 * om_1 * omb_1 * x1 * Sin(om_1 * t2_0)
          +   -1
            * om_1
            * v2
            * (1 + -1 * (Cos(u))^2)^(1 / 2)
            * Sin(om_1 * t2_0)
          + omb_1 * v1 * (Sin(om_1 * t2_0))^2
          +   -1
            * om_1
            * v2
            * Cos(u)
            * Sin(om_1 * t2_0)
            * Sin(omb_1 * t2_0)
```

```
                  +   -1
                    * om_1
                    * v2
                    * Cos(omb_1 * t2_0)
                    * Sin(om_1 * t2_0)
                    * Sin(u)
                  +   om_1
                    * v2
                    * Cos(om_1 * t2_0)
                    * Sin(omb_1 * t2_0)
                    * Sin(u)))
            * Sin(om_3 * t5)
          +   v2
            * Cos(-1 * om_1 * t2_0 + omb_1 * t2_0 + u + Pi / 4)
            * Sin(om_3 * t5)
          +   v2
            * (Cos(om_3 * t5))^2
            * Sin(-1 * om_1 * t2_0 + omb_1 * t2_0 + u + Pi / 4)
          +   v2
            * (Sin(om_3 * t5))^2
            * Sin(-1 * om_1 * t2_0 + omb_1 * t2_0 + u + Pi / 4)))
 ^2
+   (  (om_3)^-1
     * (  -1 * v1 * Cos(om_3 * t5)
        +   om_3
          * (  (om_1)^-1
             * (omb_1)^-1
             * (  -1 * omb_1 * v1 * Cos(om_1 * t2_0)
                + om_1 * omb_1 * x2 * Cos(om_1 * t2_0)
                + omb_1 * v1 * (Cos(om_1 * t2_0))^2
                + om_1 * v2 * Cos(om_1 * t2_0) * Cos(u)
                +   -1
                  * om_1
                  * v2
                  * Cos(om_1 * t2_0)
                  * Cos(omb_1 * t2_0)
```

```
        + -1 * om_1 * omb_1 * x1 * Sin(om_1 * t2_0)
        +   -1
          * om_1
          * v2
          * (1 + -1 * (Cos(u))^2)^(1 / 2)
          * Sin(om_1 * t2_0)
        + omb_1 * v1 * (Sin(om_1 * t2_0))^2
        +   -1
          * om_1
          * v2
          * Cos(u)
          * Sin(om_1 * t2_0)
          * Sin(omb_1 * t2_0)
        +   -1
          * om_1
          * v2
          * Cos(omb_1 * t2_0)
          * Sin(om_1 * t2_0)
          * Sin(u)
        +   om_1
          * v2
          * Cos(om_1 * t2_0)
          * Sin(omb_1 * t2_0)
          * Sin(u)))
  * Cos(om_3 * t5)
+ v1 * (Cos(om_3 * t5))^2
+   v2
  * Cos(om_3 * t5)
  * Cos(-1 * om_1 * t2_0 + omb_1 * t2_0 + u + Pi / 4)
+   -1
  * v2
  * (Cos(om_3 * t5))^2
  * Cos(-1 * om_1 * t2_0 + omb_1 * t2_0 + u + Pi / 4)
```

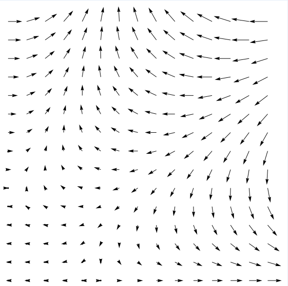
```
+   -1
  * om_3
  * (  (om_1)^-1
     * (omb_1)^-1
     * (  om_1 * omb_1 * x1 * Cos(om_1 * t2_0)
        +   om_1
          * v2
          * Cos(om_1 * t2_0)
          * (1 + -1 * (Cos(u))^2)^(1 / 2)
        + -1 * omb_1 * v1 * Sin(om_1 * t2_0)
        + om_1 * omb_1 * x2 * Sin(om_1 * t2_0)
        + om_1 * v2 * Cos(u) * Sin(om_1 * t2_0)
        +   -1
          * om_1
          * v2
          * Cos(omb_1 * t2_0)
          * Cos(u)
          * Sin(om_1 * t2_0)
        +   om_1
          * v2
          * Cos(om_1 * t2_0)
          * Cos(u)
          * Sin(omb_1 * t2_0)
        +   om_1
          * v2
          * Cos(om_1 * t2_0)
          * Cos(omb_1 * t2_0)
          * Sin(u)
        +   om_1
          * v2
          * Sin(om_1 * t2_0)
          * Sin(omb_1 * t2_0)
          * Sin(u)))
  * Sin(om_3 * t5)
```

```
        +   -1
          * v2
          *   (  1
                +   -1
                  * (Cos(-1 * om_1 * t2_0 + omb_1 * t2_0 + u + Pi / 4))^2)
           ^(1 / 2)
          * Sin(om_3 * t5)
        + v1 * (Sin(om_3 * t5))^2
        +   -1
          * v2
          * Cos(-1 * om_1 * t2_0 + omb_1 * t2_0 + u + Pi / 4)
          * (Sin(om_3 * t5))^2))
  ^2
>= (p)^2)
```

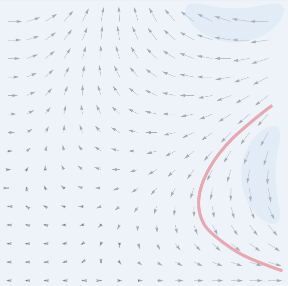
This is just one branch to prove for aircraft ...

Differential Invariants for Differential Equations

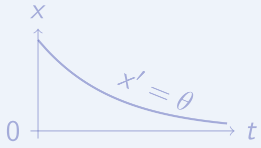
Differential Invariant

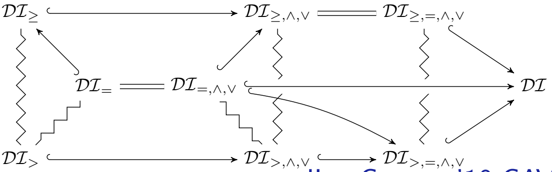


Differential Cut



Differential Ghost





Logic

Provability study

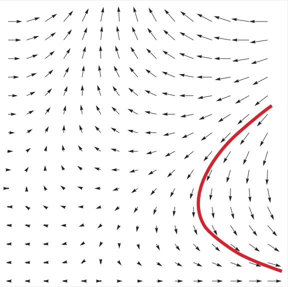
Math

Character- istic PDE

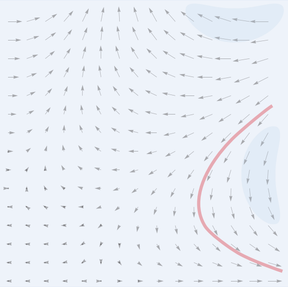
JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12

Differential Invariants for Differential Equations

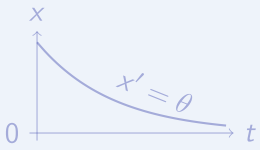
Differential Invariant

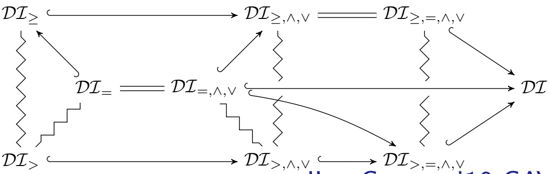


Differential Cut



Differential Ghost





Logic

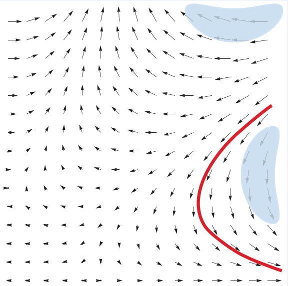
Provability study

Math

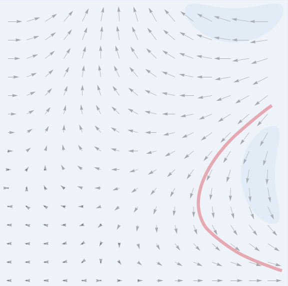
Character-istic PDE

JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12

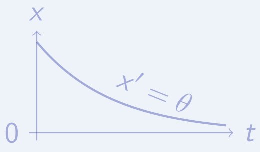
Differential Invariant

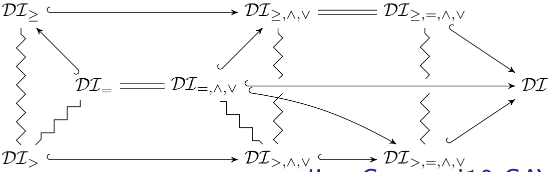


Differential Cut



Differential Ghost





Logic

Provability study

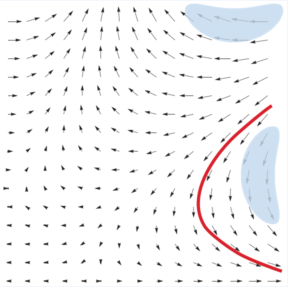
Math

Character-istic PDE

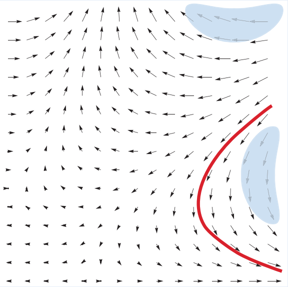
JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12

Differential Invariants for Differential Equations

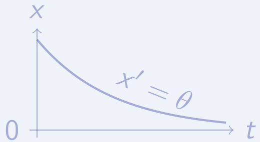
Differential Invariant

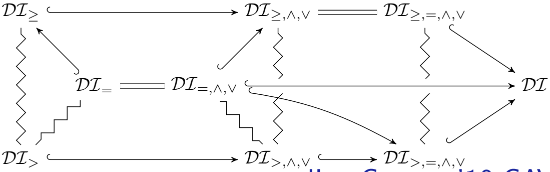


Differential Cut



Differential Ghost





Logic

Provability study

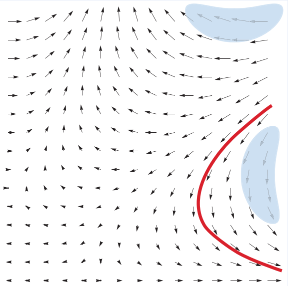
Math

Character-istic PDE

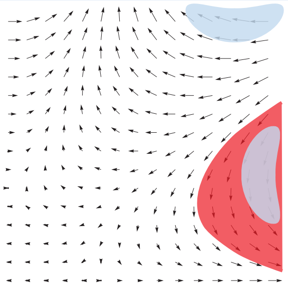
JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12

Differential Invariants for Differential Equations

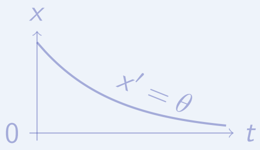
Differential Invariant

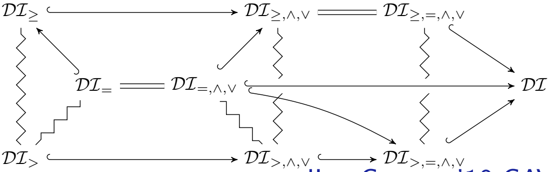


Differential Cut



Differential Ghost





Logic

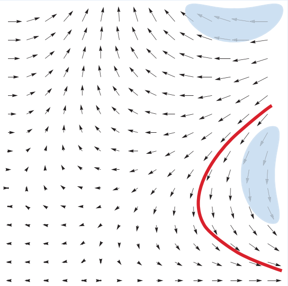
Provability study

Math

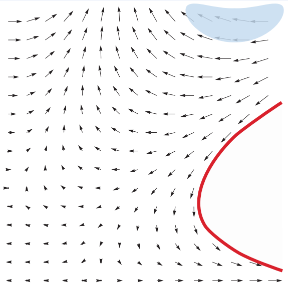
Characteristic PDE

JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12

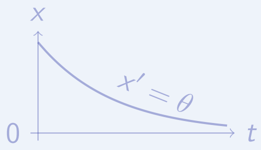
Differential Invariant

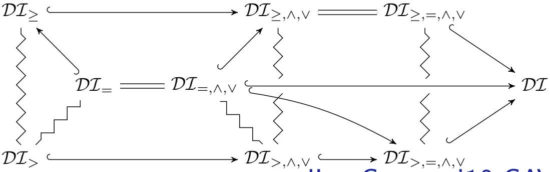


Differential Cut



Differential Ghost





Logic

Provability study

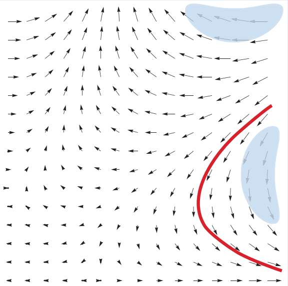
Math

Character-istic PDE

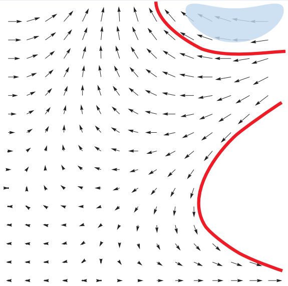
JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12

Differential Invariants for Differential Equations

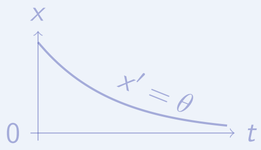
Differential Invariant

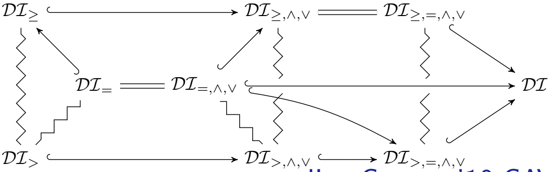


Differential Cut



Differential Ghost





Logic

Provability study

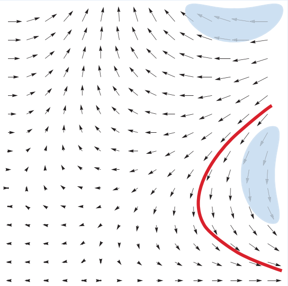
Math

Character-istic PDE

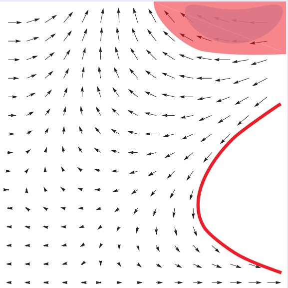
JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12

Differential Invariants for Differential Equations

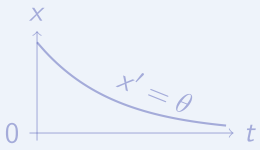
Differential Invariant

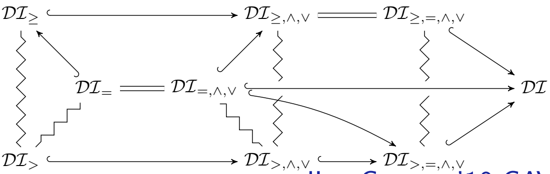


Differential Cut



Differential Ghost





Logic

Provability study

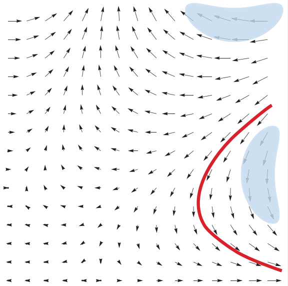
Math

Characteristic PDE

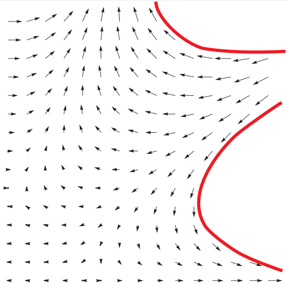
JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12

Differential Invariants for Differential Equations

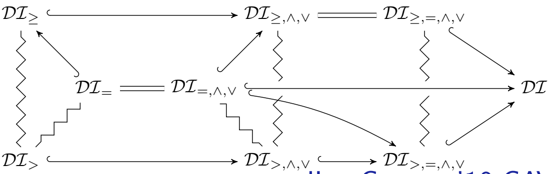
Differential Invariant



Differential Cut



Differential Ghost



Logic

Provability study

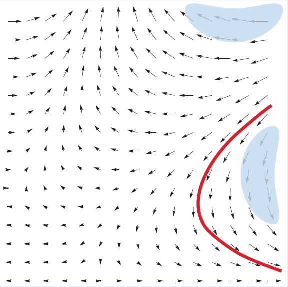
Math

Character-istic PDE

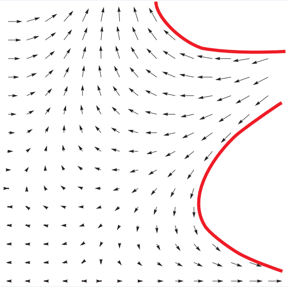
JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12

Differential Invariants for Differential Equations

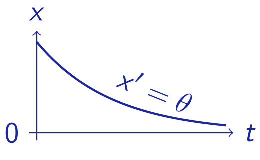
Differential Invariant

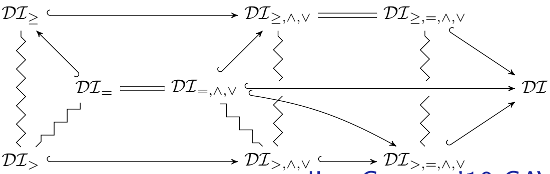


Differential Cut



Differential Ghost





Logic

Provability study

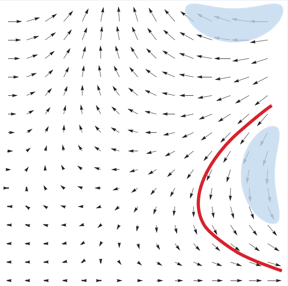
Math

Character-istic PDE

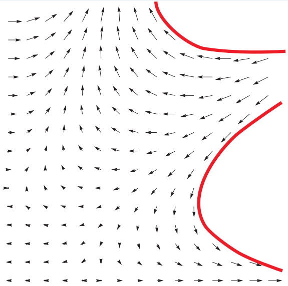
JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12

Differential Invariants for Differential Equations

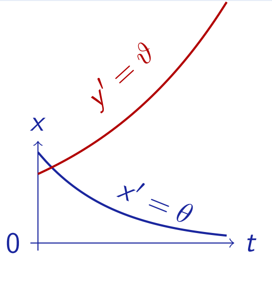
Differential Invariant

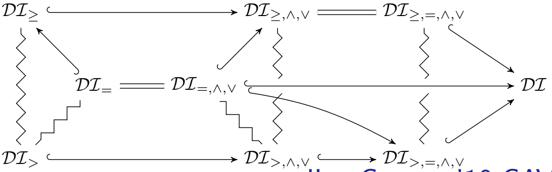


Differential Cut



Differential Ghost





Logic

Provability study

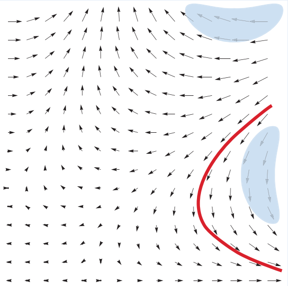
Math

Character-istic PDE

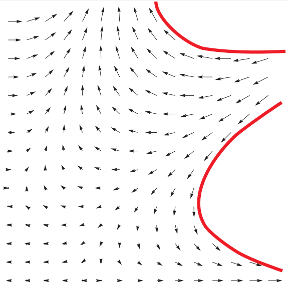
JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12

Differential Invariants for Differential Equations

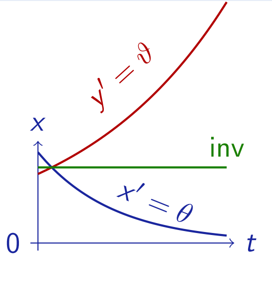
Differential Invariant

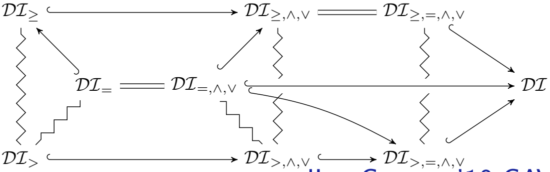


Differential Cut



Differential Ghost





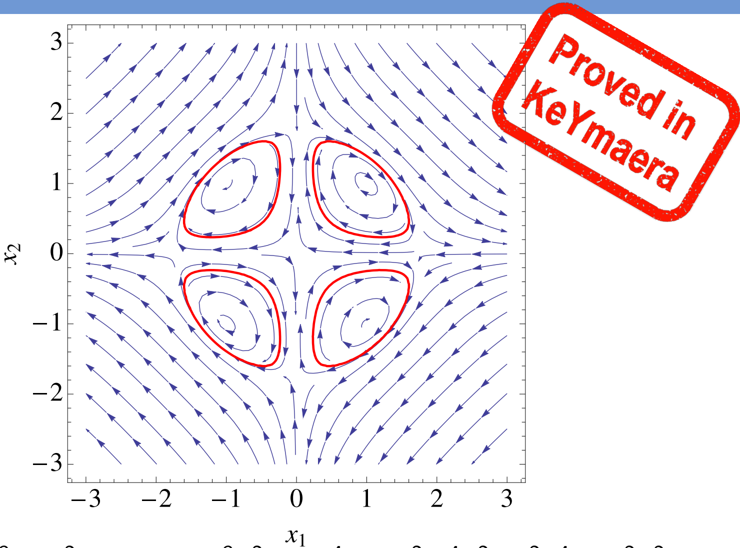
Logic

Provability study

Math

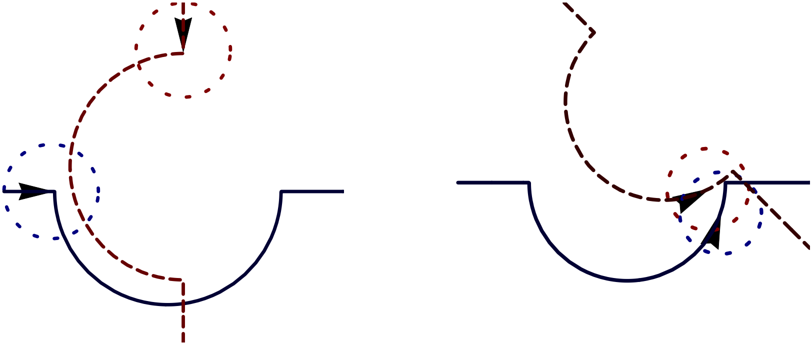
Character-istic PDE

JLogComput'10,CAV'08,FMSD'09,LMCS'12,ITP'12



$$[x' = 2x^4y+4x^2y^3-6x^2y, y' = -4x^3y^2-2xy^4+6xy^2]x^4y^2+x^2y^4-3x^2y^2 \leq c$$

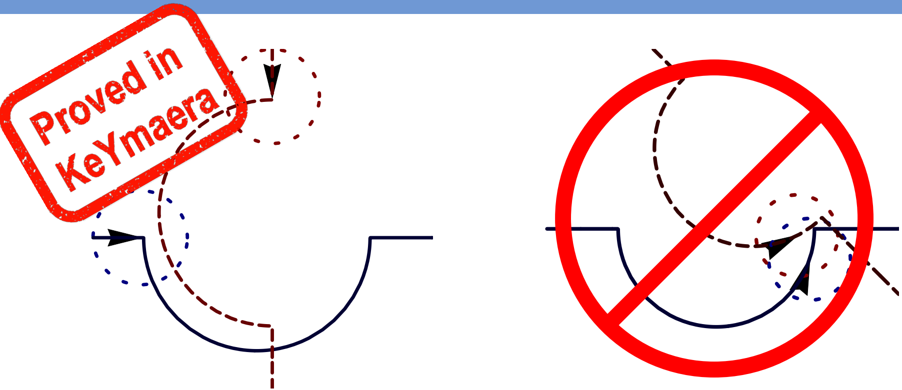
Air Traffic Control



Verification

Simple proof distinguishes safe from unsafe flight maneuver

Air Traffic Control



Verification

Simple proof distinguishes safe from unsafe flight maneuver

Differential Radical Invariants

Theorem (Differential radical invariant characterization)

$$\frac{h = 0 \rightarrow \bigwedge_{i=0}^{N-1} \mathcal{L}_p^{(i)}(h) = 0}{h = 0 \rightarrow [x' = p]h = 0}$$

characterizes all algebraic invariants, where $N = ord\ \sqrt[\mathcal{L}]{(h)}$*, i.e.*

$$\mathcal{L}_p^{(N)}(h) = \sum_{i=0}^{N-1} g_i \mathcal{L}_p^{(i)}(h)$$

Corollary (Algebraic Invariants Decidable)

Invariance decidable for real algebraic $h = 0$*.*

TACAS'14

Case Study: Longitudinal Dynamics of an Airplane

6th Order Longitudinal Equations

$$u' = \frac{X}{m} - g\sin(\theta) - qw \qquad u : \text{axial velocity}$$

$$w' = \frac{Z}{m} + g\cos(\theta) + qu \qquad w : \text{vertical velocity}$$

$$x' = \cos(\theta)u + \sin(\theta)w \qquad x : \text{range}$$

$$z' = -\sin(\theta)u + \cos(\theta)w \qquad z : \text{altitude}$$

$$\theta' = q \qquad \theta : \text{pitch angle}$$

$$q' = \frac{M}{I_{yy}} \qquad q : \text{pitch rate}$$

X : thrust along u, Z : thrust along w, M : thrust moment for w
g : gravity, m : mass, I_{yy} : inertia second diagonal

Case Study: Automatically Generated Invariants

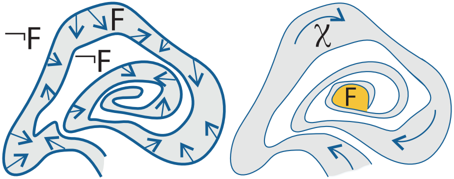
Automatically Generated Invariant Functions

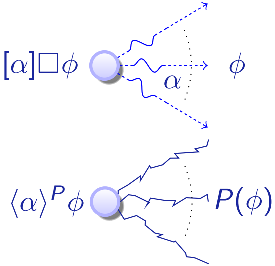
$$\frac{Mz}{I_{yy}} + g\theta + \left(\frac{X}{m} - qw\right)\cos(\theta) + \left(\frac{Z}{m} + qu\right)\sin(\theta)$$

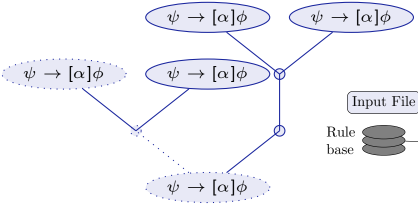
$$\frac{Mx}{I_{yy}} - \left(\frac{Z}{m} + qu\right)\cos(\theta) + \left(\frac{X}{m} - qw\right)\sin(\theta)$$

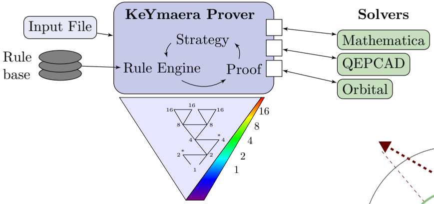
$$-q^2 + \frac{2M\theta}{I_{yy}}$$

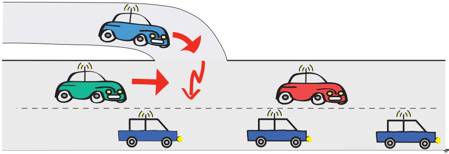
Outline

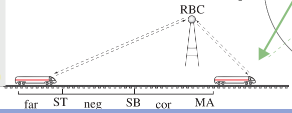

¬F
F
¬F
χ
F

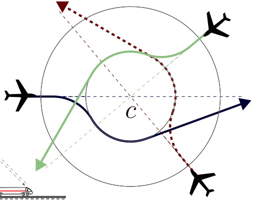

$[\alpha]\Box\phi$
α
ϕ
$\langle\alpha\rangle^P\phi$
$P(\phi)$


$\psi \to [\alpha]\phi$
$\psi \to [\alpha]\phi$
$\psi \to [\alpha]\phi$
$\psi \to [\alpha]\phi$
$\psi \to [\alpha]\phi$

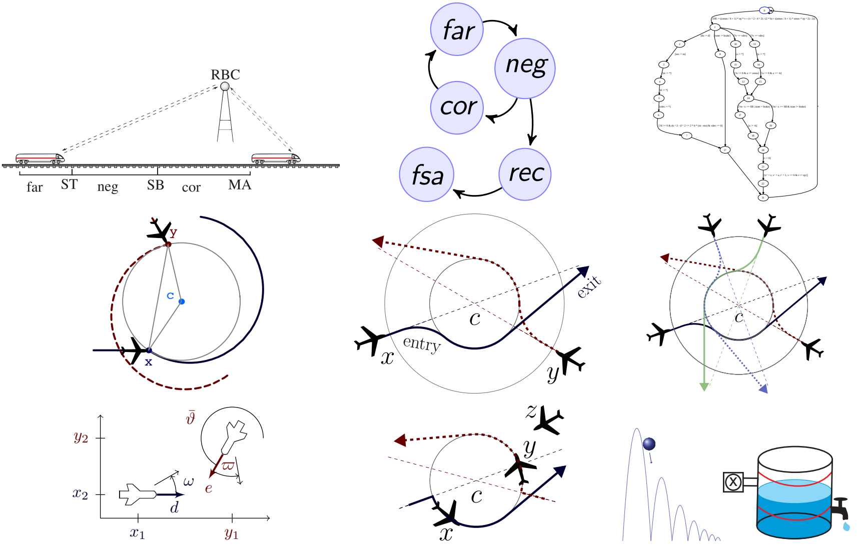

Input File
Rule base
KeYmaera Prover
Strategy
Rule Engine
Proof
Solvers
Mathematica
QEPCAD
Orbital
16
8
4
2
1




RBC
far
ST
neg
SB
cor
MA

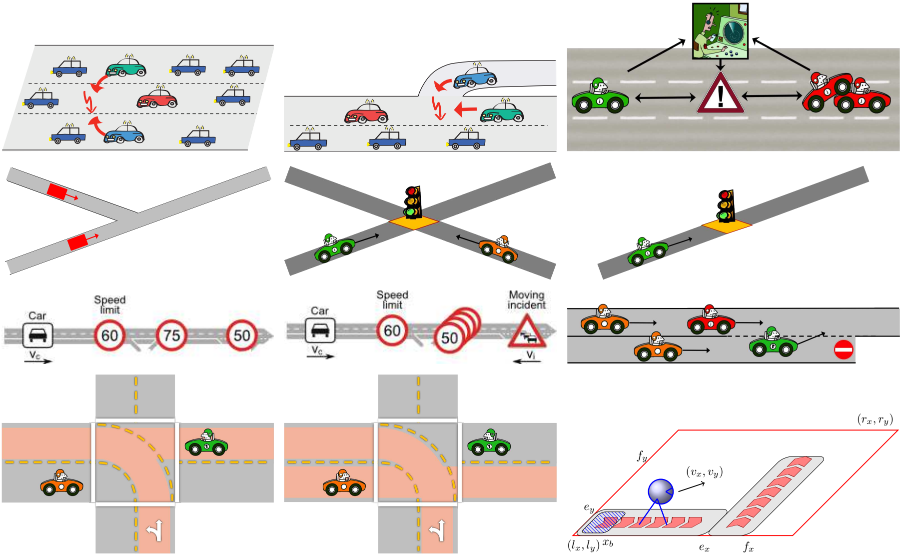

c

Successful CPS Proofs



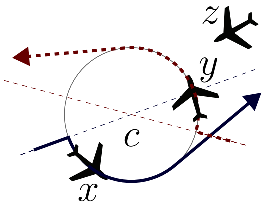
ICFEM'09,CAV'08,FM'09,HSCC'11

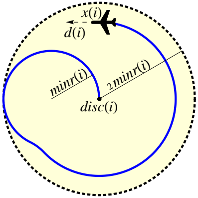
Successful CPS Proofs

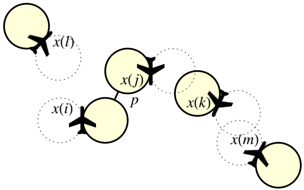


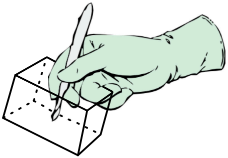
FM'11,LMCS'12,ICCPS'12,ITSC'11,ITSC'13,IJCAR'12

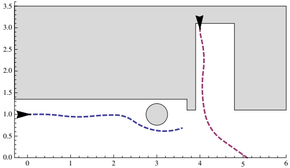
Successful CPS Proofs

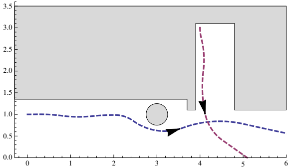


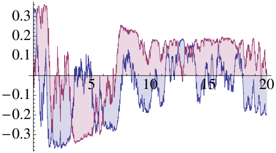


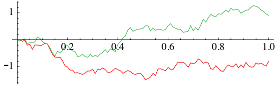


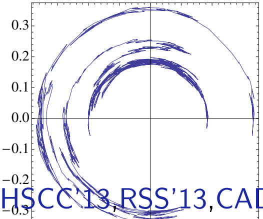




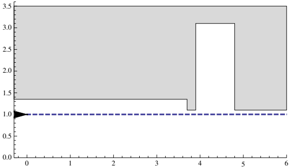
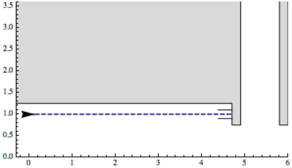
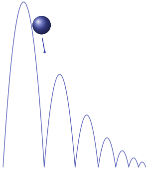
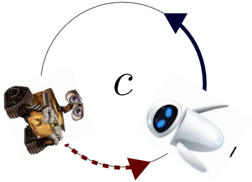


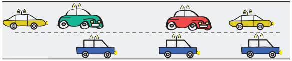
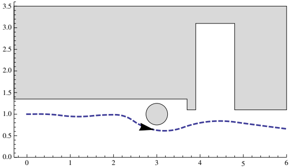
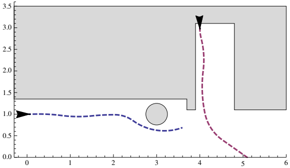
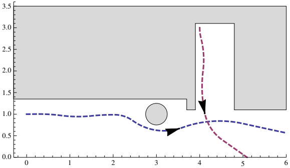






HSCC'11,HSCC'13,HSCC'13,RSS'13,CADE'12

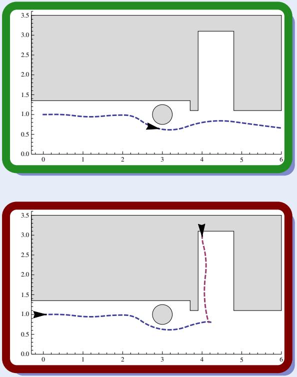





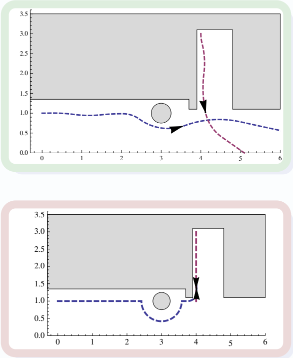
students in 15-424/624 *Foundations of Cyber-Physical Systems* course

What is Safe?

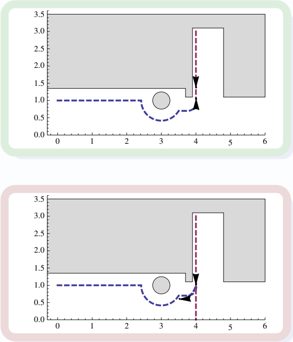
Static safety



Passive safety



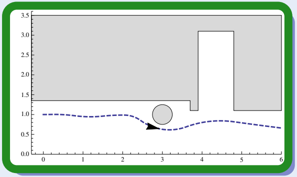
Passive friendly

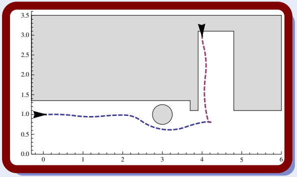


✓ Verified with KeYmaera

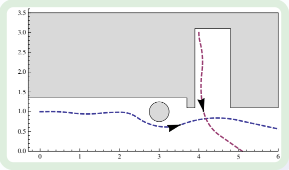
Sensor failure   Actuator disturbance   Robot and obstacle shape

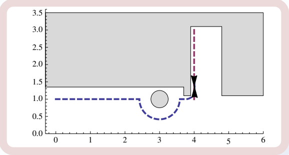
Static safety



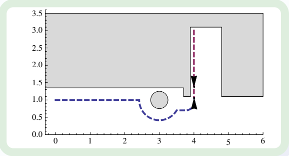


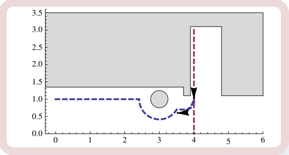
Passive safety





Passive friendly

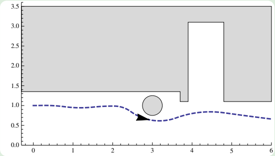


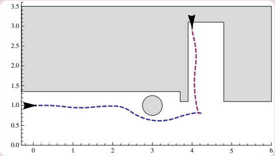


✓ Verified with KeYmaera

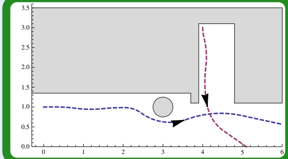
Sensor failure   Actuator disturbance   Robot and obstacle shape

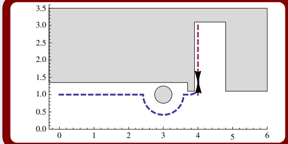
Static safety



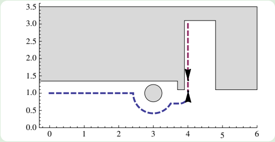


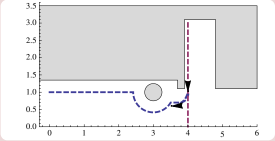
Passive safety





Passive friendly



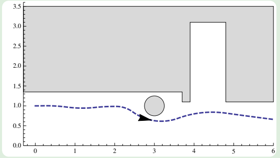


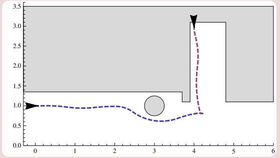
✓ Verified with KeYmaera

▸ Sensor failure ▸ Actuator disturbance ▸ Robot and obstacle shape

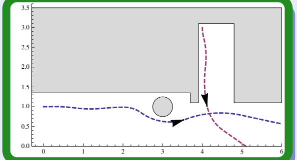
What is Safe?

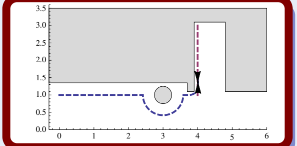
Static safety



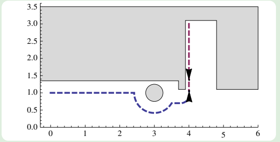


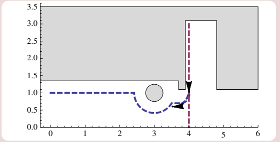
Passive safety





Passive friendly



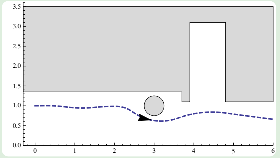


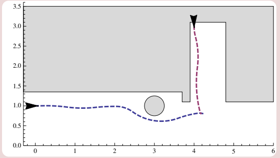
✓ Verified with KeYmaera

▸ Sensor failure ▸ Actuator disturbance ▸ Robot and obstacle shape

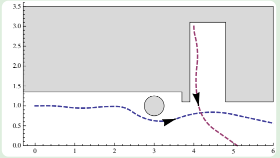
What is Safe?

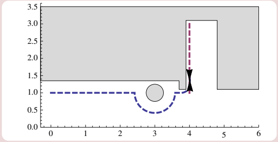
Static safety



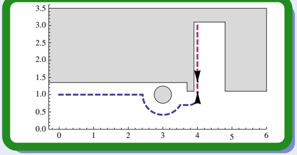


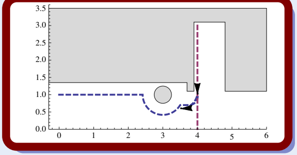
Passive safety





Passive friendly



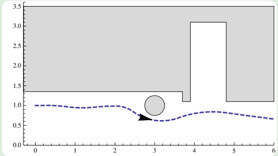


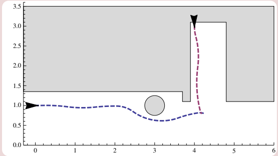
✓ Verified with KeYmaera

▸ Sensor failure ▸ Actuator disturbance ▸ Robot and obstacle shape

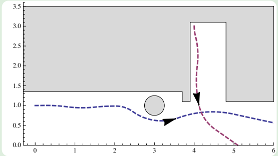
What is Safe?

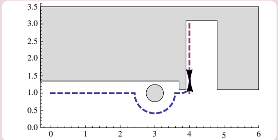
Static safety



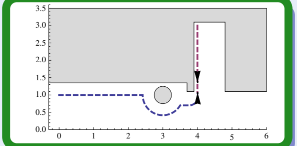


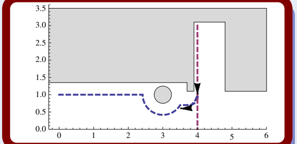
Passive safety





Passive friendly



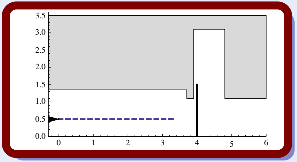


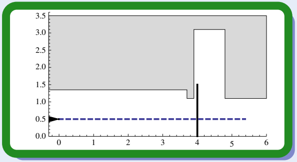
✓ Verified with KeYmaera

Sensor failure · Actuator disturbance · Robot and obstacle shape

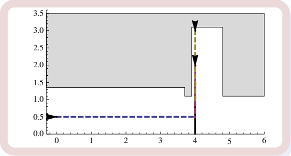
What is the Goal of the Robot?

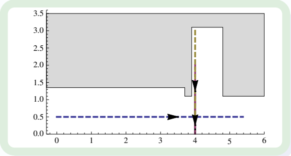
Pass waypoint



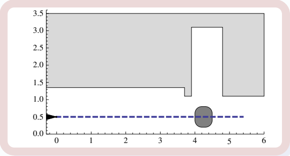


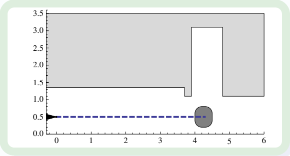
Cross intersection





Reach area

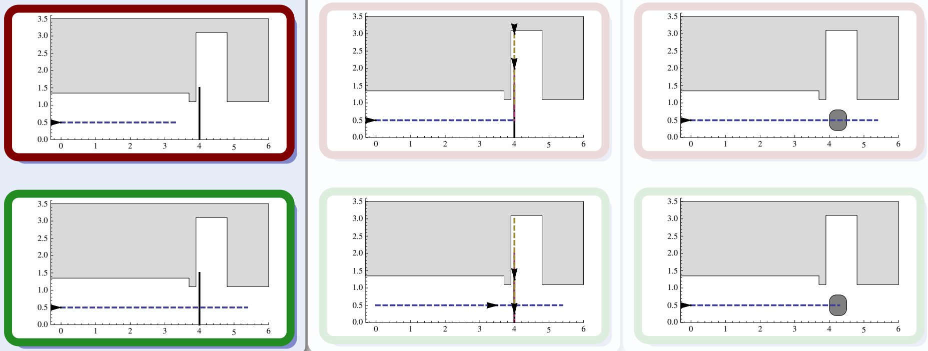




✓ Verified with KeYmaera

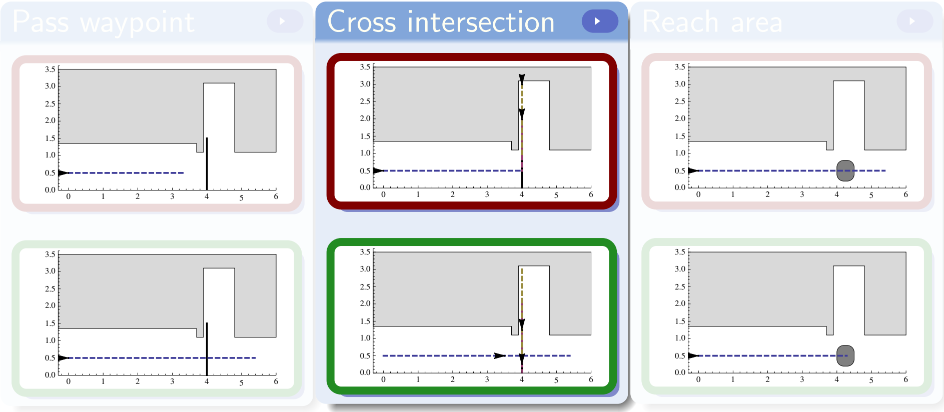
What is the Goal of the Robot?

Pass waypoint | Cross intersection | Reach area



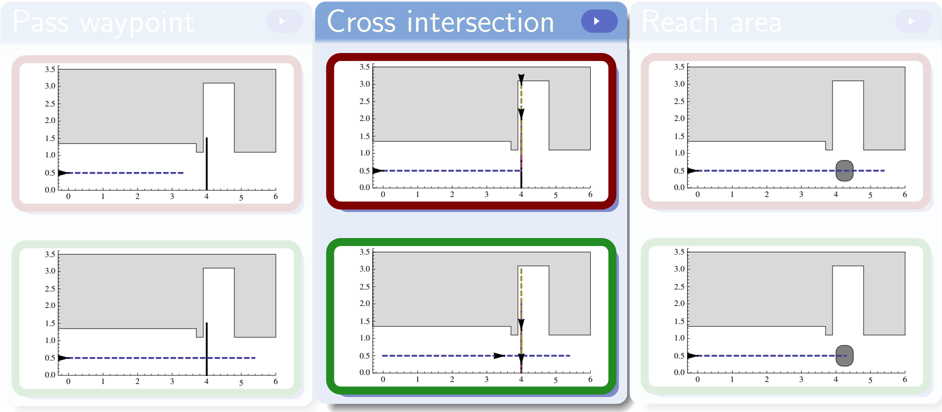
✓ Verified with KeYmaera

What is the Goal of the Robot?



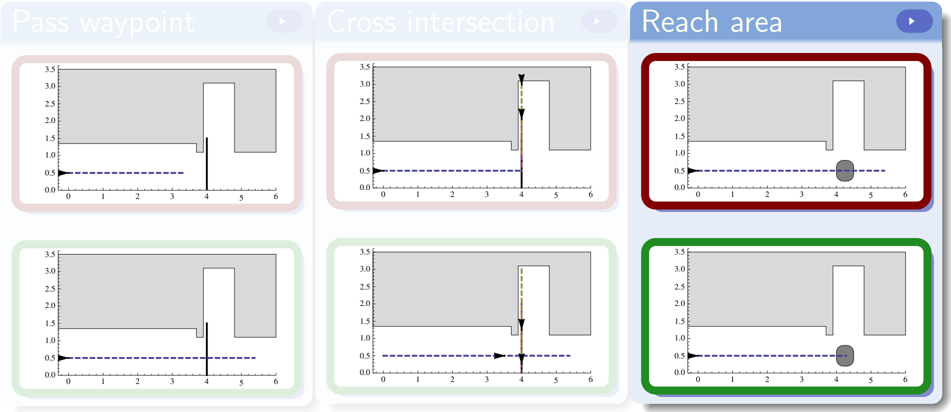
✓ Verified with KeYmaera

What is the Goal of the Robot?



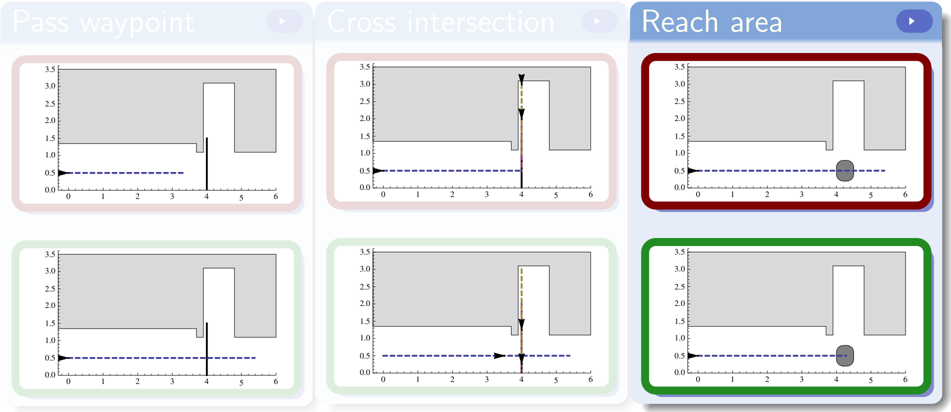
✓ Verified with KeYmaera

What is the Goal of the Robot?



✓ Verified with KeYmaera

What is the Goal of the Robot?



✓ Verified with KeYmaera

Robot Invariants and Constraints

Safety	Invariant + Safe Control (RSS'13)
static	$\lVert p_r - p_o \rVert_\infty > \frac{v_r^2}{2b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v_r\right)$
passive	$v_r = 0 \lor \lVert p_r - p_o \rVert_\infty > \frac{v_r^2}{2b} + V\frac{v_r}{b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right)$
+ sensor	$\lVert \hat{p}_r - p_o \rVert_\infty > \frac{v_r^2}{2b} + V\frac{v_r}{b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right) + U_p$
+ disturb	$\lVert p_r - p_o \rVert_\infty > \frac{v_r^2}{2bU_m} + V\frac{v_r}{bU_m} + \left(\frac{A}{bU_m}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right)$
+ failure	$\lVert \hat{p}_r - p_o \rVert_\infty > \frac{v_r^2}{2b} + V\frac{v_r}{b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v + V)\right) + U_p + g\Delta$
friendly	$\lVert p_r - p_o \rVert_\infty > \frac{v_r^2}{2b} + \frac{V^2}{2b_o} + V\left(\frac{v_r}{b} + \tau\right) + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right)$

RSS'13

Robot Invariants and Constraints

Safety	Invariant + Safe Control	(RSS'13)
static	$\|p_r - p_o\|_\infty > \frac{v_r^2}{2b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v_r\right)$	
passive	$v_r = 0 \lor \|p_r - p_o\|_\infty > \frac{v_r^2}{2b} + V\frac{v_r}{b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right)$	
+ sensor	$\ldots + \varepsilon(v_r + V)\big) + U_p$	
+ disturb	$\|p_r - p_o\|_\infty > \frac{v_r^2}{2bU_m} + V\frac{v_r}{bU_m} + \left(\frac{A}{bU_m}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right)$	
+ failure	$\|\hat{p}_r - p_o\|_\infty > \frac{v_r^2}{2b} + V\frac{v_r}{b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v + V)\right) + U_p + g\Delta$	
friendly	$\|p_r - p_o\|_\infty > \frac{v_r^2}{2b} + \frac{V^2}{2b_o} + V\left(\frac{v_r}{b} + \tau\right) + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right)$	

Question

How to find and justify constraints? Proof!

RSS'13

Outline

Benefits of Logical Foundations for V & V

Proving

Safety	Formalize system properties: What is "Safe"? "Reach goal"?
Models	Formalize system models
Assumptions	Make assumptions explicit
Constraints	Reveal invariants, switching conditions, starting conditions
Design	Invariants guide safe controller design
Constructive	Construct models along with their proof

Byproducts

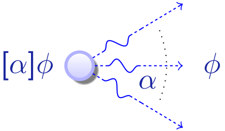
Analyze	Determine design trade-offs & feasibility early
Synthesize	Turn high-level models into code & correctness monitors
Certify	Proofs as artifacts for certification

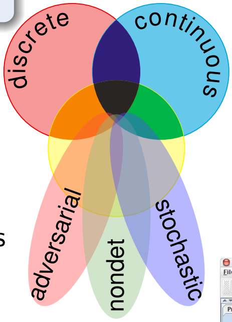
Tools

KeYmaera	Theorem prover for CPS

differential dynamic logic

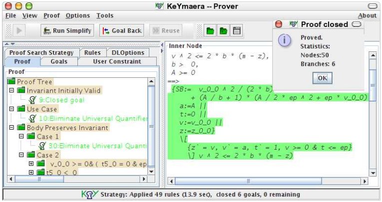
$$d\mathcal{L} = \mathrm{DL} + \mathrm{HP}$$

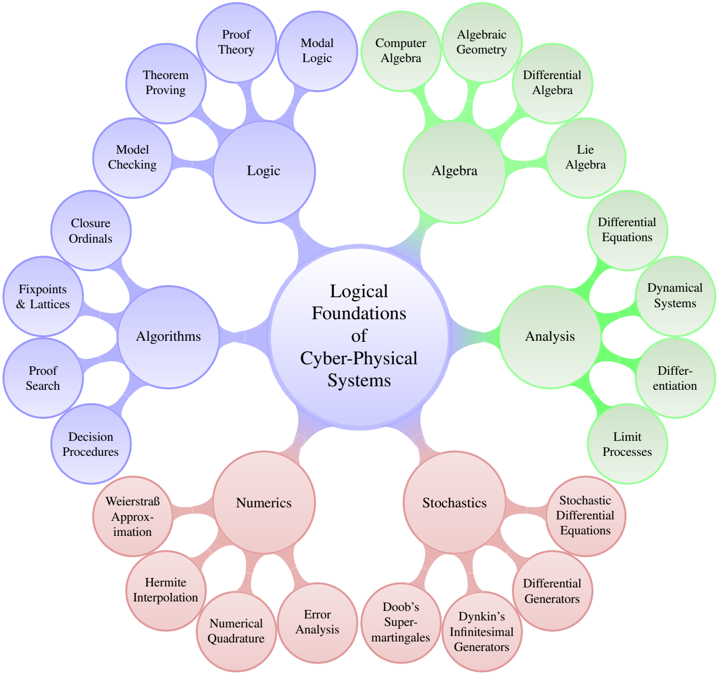


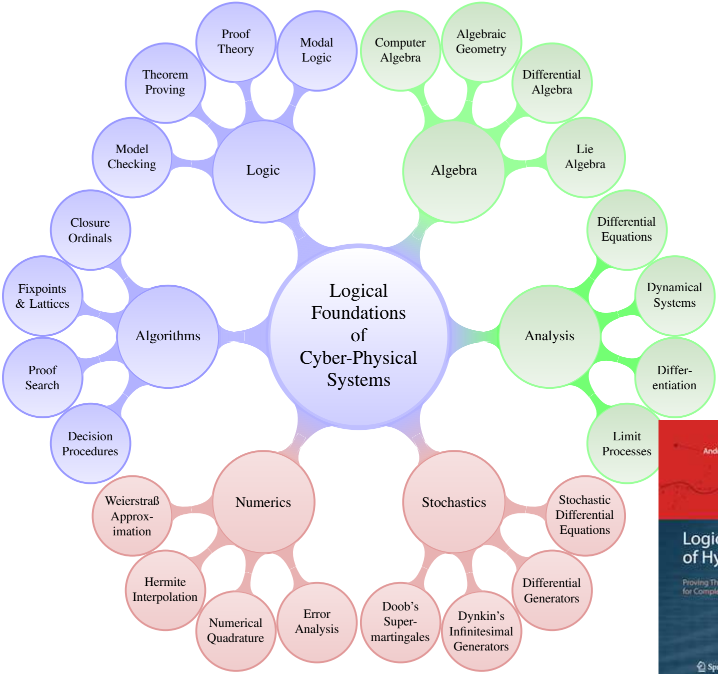


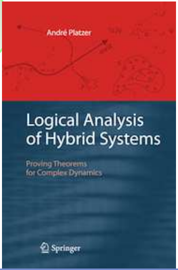
- Multi-dynamical systems
- Combine simple dynamics
- Tame complexity
- Logic & proofs for CPS
- Theory of CPS
- Applications
- Undergrad course 15-424

KeYmaera




Logical Foundations of Cyber-Physical Systems
Logic
Proof Theory
Modal Logic
Theorem Proving
Model Checking
Algebra
Computer Algebra
Algebraic Geometry
Differential Algebra
Lie Algebra
Analysis
Differential Equations
Dynamical Systems
Differ- entiation
Limit Processes
Algorithms
Closure Ordinals
Fixpoints & Lattices
Proof Search
Decision Procedures
Numerics
Weierstraß Approx- imation
Hermite Interpolation
Numerical Quadrature
Error Analysis
Stochastics
Stochastic Differential Equations
Differential Generators
Dynkin's Infinitesimal Generators
Doob's Super- martingales


Logical Foundations of Cyber-Physical Systems
Logic
Proof Theory
Modal Logic
Theorem Proving
Model Checking
Algebra
Computer Algebra
Algebraic Geometry
Differential Algebra
Lie Algebra
Algorithms
Closure Ordinals
Fixpoints & Lattices
Proof Search
Decision Procedures
Analysis
Differential Equations
Dynamical Systems
Differ-entiation
Limit Processes
Numerics
Weierstraß Approx-imation
Hermite Interpolation
Numerical Quadrature
Error Analysis
Stochastics
Stochastic Differential Equations
Differential Generators
Dynkin's Infinitesimal Generators
Doob's Super-martingales


André Platzer
Logical Analysis of Hybrid Systems
Proving Theorems for Complex Dynamics
Springer

André Platzer.
Logics of dynamical systems.
In LICS [13], pages 13–24.
doi:10.1109/LICS.2012.13.

André Platzer.
A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems.
Logical Methods in Computer Science, 8(4):1–44, 2012.
Special issue for selected papers from CSL'10.
doi:10.2168/LMCS-8(4:17)2012.

André Platzer.
Stochastic differential dynamic logic for stochastic hybrid programs.
In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *CADE*, volume 6803 of *LNCS*, pages 431–445. Springer, 2011.
doi:10.1007/978-3-642-22438-6_34.

André Platzer.
A complete axiomatization of differential game logic for hybrid games.

Technical Report CMU-CS-13-100R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January, Revised and extended in July 2013.

André Platzer.
Differential dynamic logic for hybrid systems.
J. Autom. Reas., 41(2):143–189, 2008.
doi:10.1007/s10817-008-9103-8.

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.
J. Log. Comput., 20(1):309–352, 2010.
doi:10.1093/logcom/exn070.

André Platzer and Edmund M. Clarke.
Computing differential invariants of hybrid systems as fixedpoints.

Form. Methods Syst. Des., 35(1):98–120, 2009.
Special issue for selected papers from CAV'08.
doi:10.1007/s10703-009-0079-8.

André Platzer.
The structure of differential invariants and differential cut elimination.
Logical Methods in Computer Science, 8(4):1–38, 2012.
doi:10.2168/LMCS-8(4:16)2012.

André Platzer.
A differential operator approach to equational differential invariants.
In Lennart Beringer and Amy Felty, editors, *ITP*, volume 7406 of *LNCS*, pages 28–48. Springer, 2012.
doi:10.1007/978-3-642-32347-8_3.

Khalil Ghorbal and André Platzer.
Characterizing algebraic invariants by differential radical invariants.
In Erika Ábrahám and Klaus Havelund, editors, *TACAS*, volume 8413 of *LNCS*, pages 279–294. Springer, 2014.
doi:10.1007/978-3-642-54862-8_19.

André Platzer.
Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics.
Springer, Heidelberg, 2010.
doi:10.1007/978-3-642-14509-4.

Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012.
IEEE, 2012.