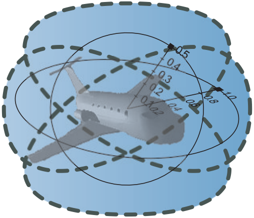
Dynamic Logic for Dynamical Systems

André Platzer

Carnegie Mellon University



Outline

Cyber-Physical Systems Analysis: Aircraft Example

Which control decisions are safe for aircraft collision avoidance?

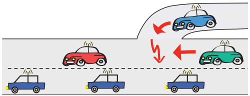
Cyber-Physical Systems

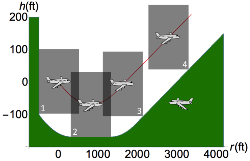
CPSs combine cyber capabilities with physical capabilities
to solve problems that neither part could solve alone.

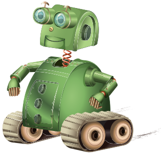
Cyber-Physical Systems Promise Transformative Impact!

Prospects: Safety & Efficiency

(Autonomous) cars | (Auto)Pilot support | Robots near humans







Cyber-Physical Systems

CPSs combine cyber capabilities with physical capabilities
to solve problems that neither part could solve alone.

Can you trust a computer to control physics?

Can you trust a computer to control physics?

1. Depends on how it has been programmed
2. And on what will happen if it malfunctions

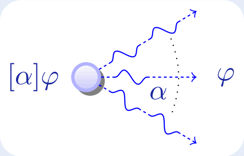
Rationale

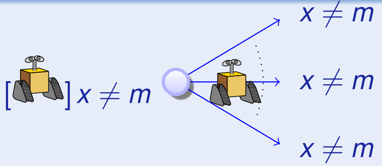
1. Safety guarantees require analytic foundations.
2. A common foundational core helps all application domains.
3. Foundations revolutionized digital computer science & our society.
4. Need even stronger foundations when software reaches out into our physical world.

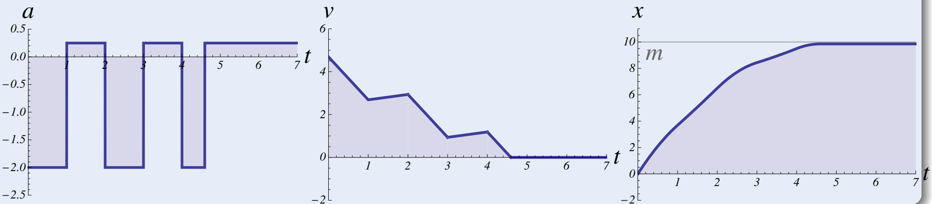
CPSs deserve proofs as safety evidence!

Outline (Modeling CPS)

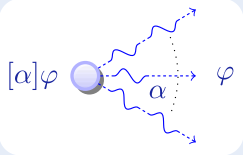
Concept (Differential Dynamic Logic) (JAR'08,LICS'12)

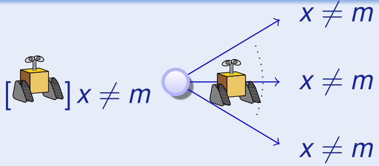






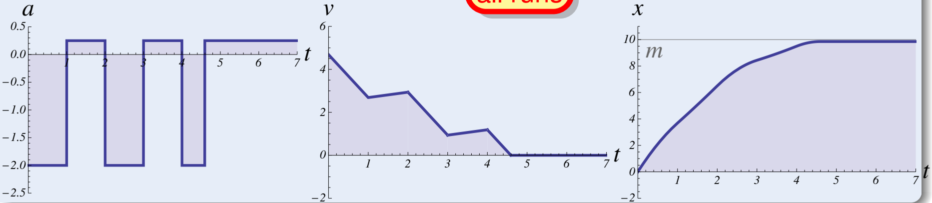
Concept (Differential Dynamic Logic) (JAR'08,LICS'12)





$$\underbrace{x \neq m \land b > 0}_{\text{init}} \rightarrow \big[\big((\text{if}(SB(x,m)) \quad a := -b) \; ; \; x' = v, v' = a\big)^*\big]\underbrace{x \neq m}_{\text{post}}$$

all runs



Definition (Hybrid program α)

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

Definition (dL Formula P)

$$e \geq \tilde{e} \mid \neg P \mid P \wedge Q \mid \forall x\, P \mid \exists x\, P \mid [\alpha]P \mid \langle \alpha \rangle P$$

JAR'08,LICS'12,JAR'17

Differential Dynamic Logic dL: Syntax

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

Definition (Hybrid program α)

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

Definition (dL Formula P)

$$e \geq \tilde{e} \mid \neg P \mid P \wedge Q \mid \forall x\, P \mid \exists x\, P \mid [\alpha]P \mid \langle\alpha\rangle P$$

All Reals | Some Reals | All Runs | Some Runs

JAR'08,LICS'12,JAR'17

Differential Dynamic Logic dL: Semantics

Definition (Hybrid program semantics) $(\llbracket\cdot\rrbracket : \mathrm{HP} \to \wp(\mathcal{S} \times \mathcal{S}))$

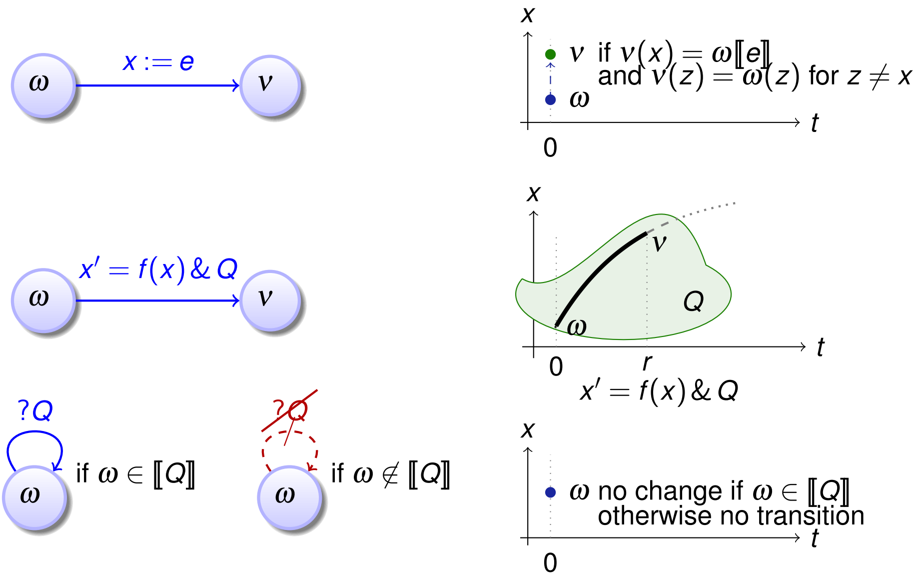
$$
\begin{aligned}
\llbracket x := e \rrbracket &= \{(\omega, \nu) : \nu = \omega \text{ except } \nu\llbracket x \rrbracket = \omega\llbracket e \rrbracket\} \\
\llbracket ?Q \rrbracket &= \{(\omega, \omega) : \omega \in \llbracket Q \rrbracket\} \\
\llbracket x' = f(x) \rrbracket &= \{(\varphi(0),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

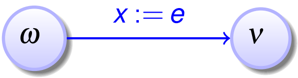
compositional semantics

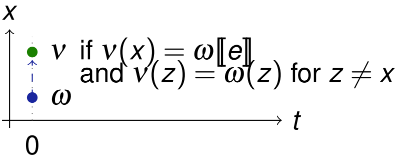
Definition (dL semantics) $(\llbracket\cdot\rrbracket : \mathrm{Fml} \to \wp(\mathcal{S}))$

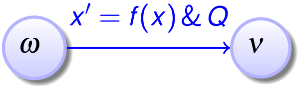
$$
\begin{aligned}
\llbracket e \geq \tilde{e} \rrbracket &= \{\omega : \omega\llbracket e \rrbracket \geq \omega\llbracket \tilde{e} \rrbracket\} \\
\llbracket \neg P \rrbracket &= \llbracket P \rrbracket^{\complement} \\
\llbracket P \wedge Q \rrbracket &= \llbracket P \rrbracket \cap \llbracket Q \rrbracket \\
\llbracket \langle \alpha \rangle P \rrbracket &= \llbracket \alpha \rrbracket \circ \llbracket P \rrbracket = \{\omega : \nu \in \llbracket P \rrbracket \text{ for some } \nu : (\omega, \nu) \in \llbracket \alpha \rrbracket\} \\
\llbracket [\alpha] P \rrbracket &= \llbracket \neg \langle \alpha \rangle \neg P \rrbracket = \{\omega : \nu \in \llbracket P \rrbracket \text{ for all } \nu : (\omega, \nu) \in \llbracket \alpha \rrbracket\} \\
\llbracket \exists x\, P \rrbracket &= \{\omega : \omega_x^r \in \llbracket P \rrbracket \text{ for some } r \in \mathbb{R}\}
\end{aligned}
$$

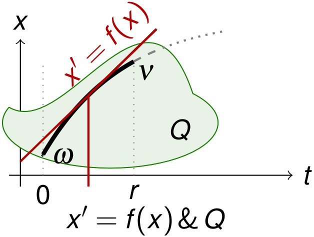
Differential Dynamic Logic dL: Semantics

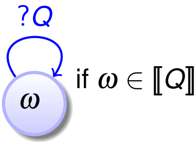


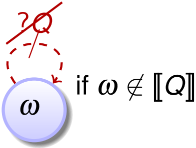

ω
$x := e$
ν

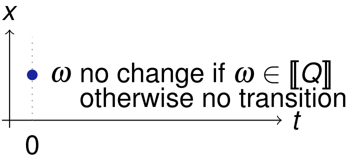

x
ν if $\nu(x) = \omega[\![e]\!]$
and $\nu(z) = \omega(z)$ for $z \neq x$
ω
t
0


ω
$x' = f(x) \,\&\, Q$
ν

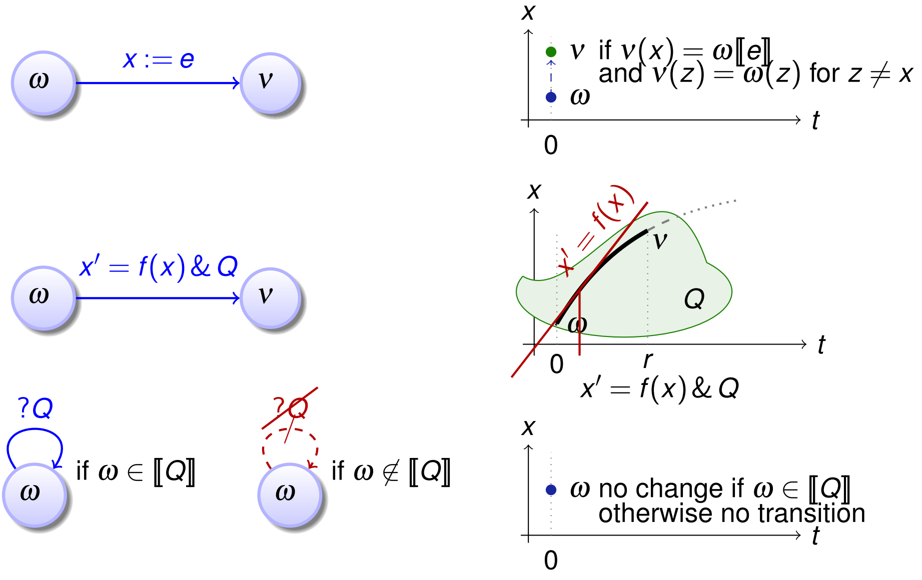

x
$x' = f(x)$
ν
Q
ω
t
0
r
$x' = f(x) \,\&\, Q$


$?Q$
ω
if $\omega \in [\![Q]\!]$

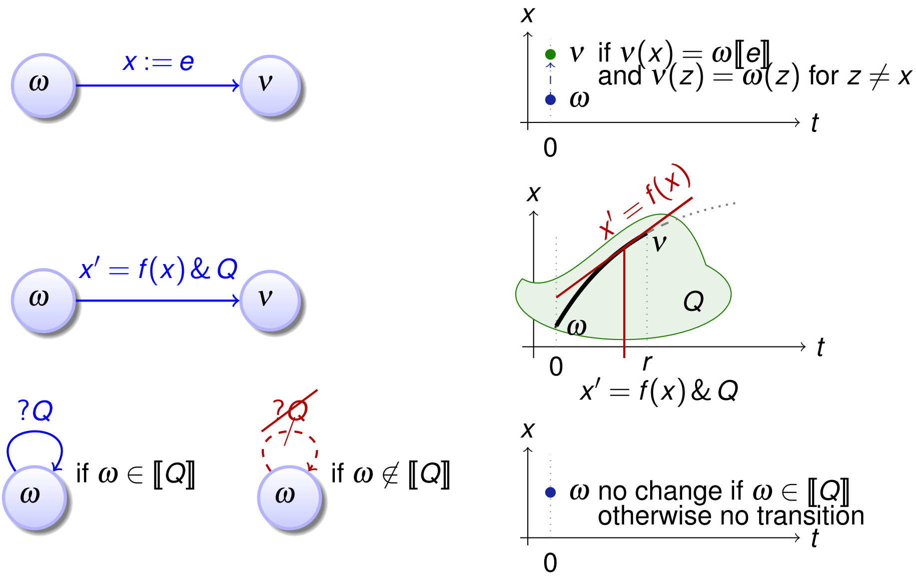

$?Q$
ω
if $\omega \notin [\![Q]\!]$


x
ω no change if $\omega \in [\![Q]\!]$
otherwise no transition
t
0

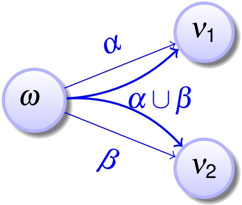
Differential Dynamic Logic dL: Semantics

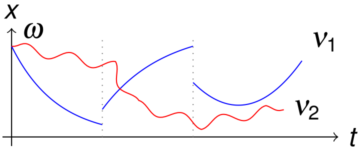


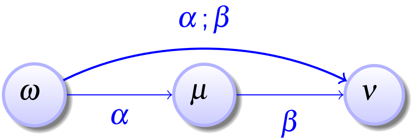
Differential Dynamic Logic dL: Semantics

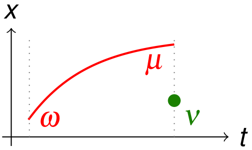


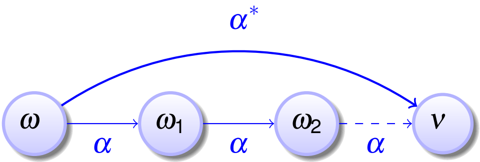
Differential Dynamic Logic dL: Semantics

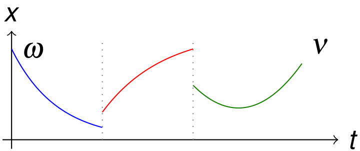




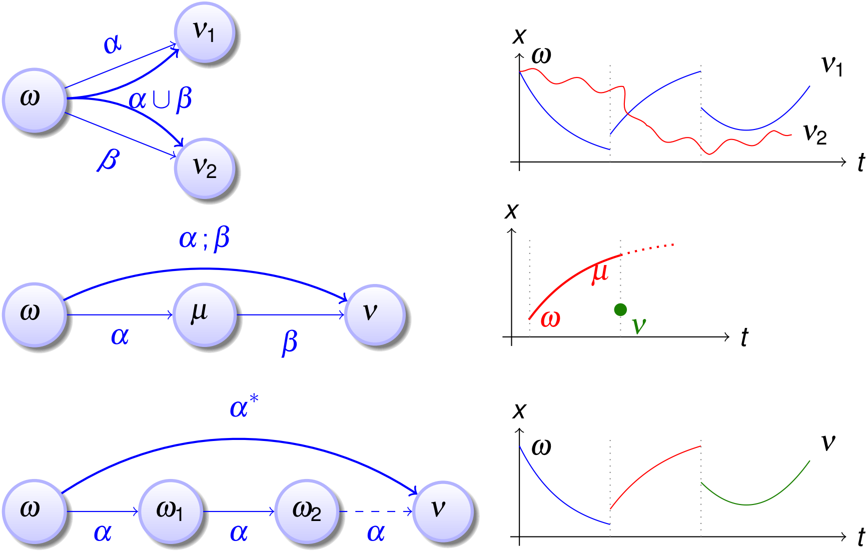




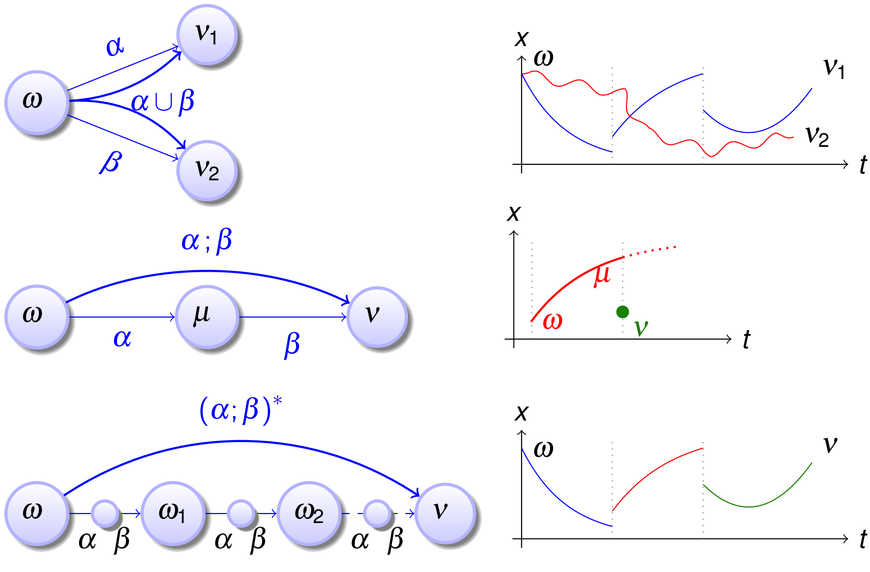




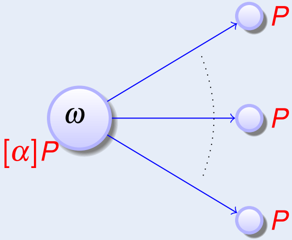
Differential Dynamic Logic dL: Semantics



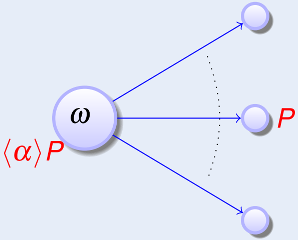
Differential Dynamic Logic dL: Semantics



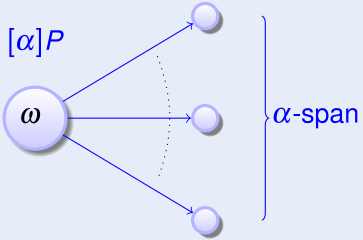
Definition (dL Formulas)



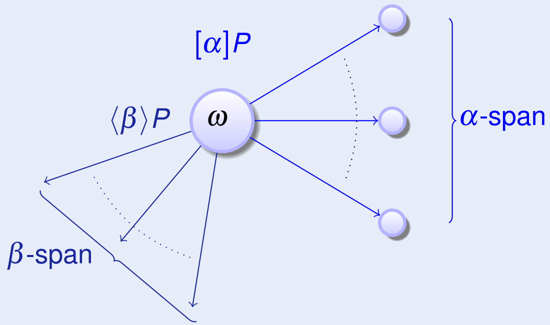
Definition (dL Formulas)



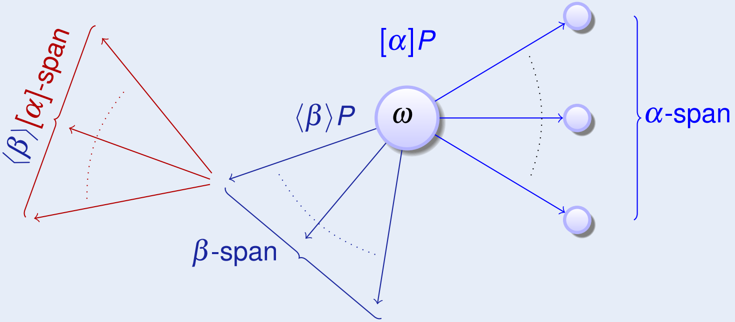
Definition (dL Formulas)



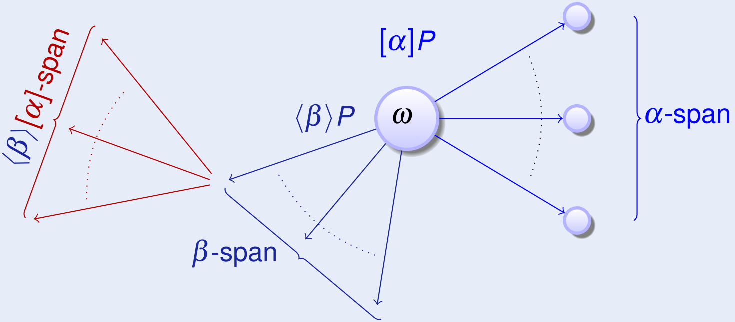
Definition (dL Formulas)



Definition (dL Formulas)



Definition (dL Formulas)



compositional semantics $\Rightarrow$ compositional proofs!

Differential Dynamic Logic dL: Semantics

Definition (Hybrid program semantics) ($[\![\cdot]\!] : \mathrm{HP} \to \wp(\mathcal{S} \times \mathcal{S})$)

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

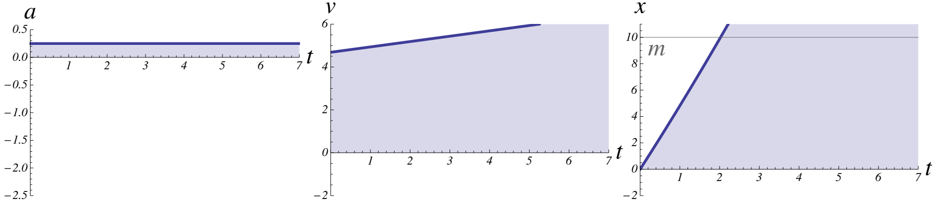
compositional semantics

Definition (dL semantics) ($[\![\cdot]\!] : \mathrm{Fml} \to \wp(\mathcal{S})$)

$$
\begin{aligned}
[\![e \geq \tilde{e}]\!] &= \{\omega : \omega[\![e]\!] \geq \omega[\![\tilde{e}]\!]\} \\
[\![\neg P]\!] &= [\![P]\!]^{\complement} \\
[\![P \wedge Q]\!] &= [\![P]\!] \cap [\![Q]\!] \\
[\![\langle\alpha\rangle P]\!] &= [\![\alpha]\!] \circ [\![P]\!] = \{\omega : \nu \in [\![P]\!] \text{ for some } \nu : (\omega,\nu) \in [\![\alpha]\!]\} \\
[\![[\alpha]P]\!] &= [\![\neg\langle\alpha\rangle\neg P]\!] = \{\omega : \nu \in [\![P]\!] \text{ for all } \nu : (\omega,\nu) \in [\![\alpha]\!]\} \\
[\![\exists x\, P]\!] &= \{\omega : \omega_x^r \in [\![P]\!] \text{ for some } r \in \mathbb{R}\}
\end{aligned}
$$

Example (Single car car_s)

$$\{x' = v, v' = a\}$$

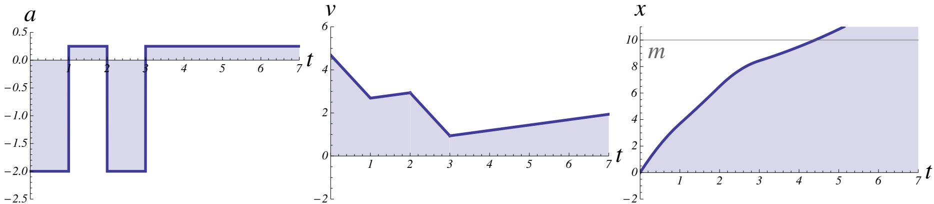


Ex: Car Control Programs

Repeat control decisions

Example (Single car car_s)

$$((a:=A \cup a:=-b);\ \{x'=v, v'=a\})^*$$

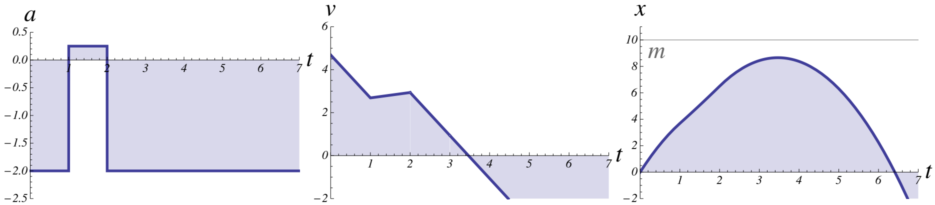


Ex: Car Control Programs

How does this model brake?

Example (Single car car_s)

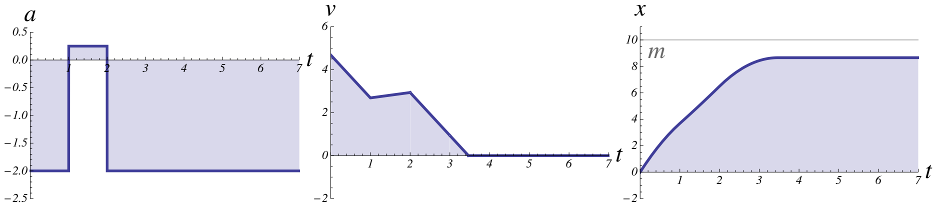
$$\big((\ a := A\ \cup a := -b);\ \{x' = v, v' = a\}\big)^*$$



Velocity bound $v \geq 0$ in evolution domain

Example (Single car car_s)

$$\big((\ a := A\ \cup a := -b);\ \{x' = v, v' = a \,\&\, v \geq 0\}\big)^*$$

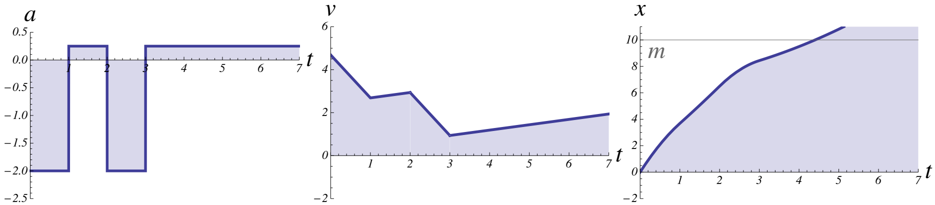


Ex: Car Control Programs

Acceleration not always safe

Example (Single car car_s)

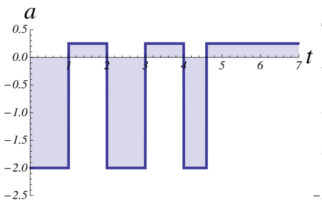
$$\big((\ a := A\ \cup a := -b);\ \{x' = v, v' = a \,\&\, v \geq 0\}\big)^*$$

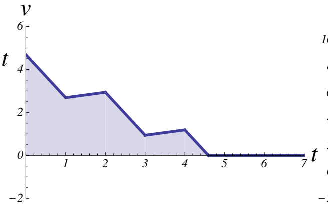


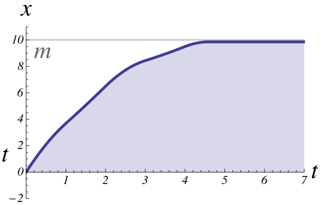
Acceleration condition $?Q$

Example (Single car car_s)

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







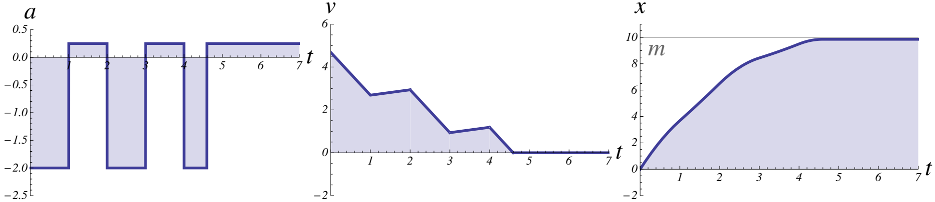
$Q \equiv$

Example (Single car car_ε time-triggered)

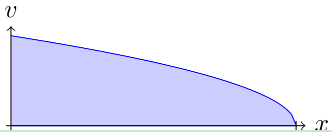
$$\Big(((?Q;a:=A)\cup a:=-b);t:=0;\{x'=v,v'=a,t'=1\,\&\,v\geq 0\wedge t\leq\varepsilon\}\Big)^*$$

Example (Safely stays before traffic light m)

$$A\geq 0\wedge b>0\rightarrow[car_\varepsilon]\,x\leq m$$



$Q \equiv$

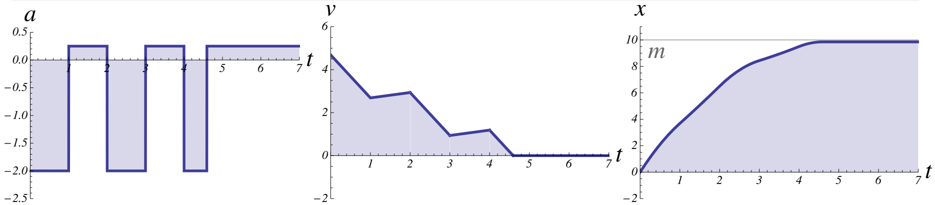


Example (Single car car_ε time-triggered)

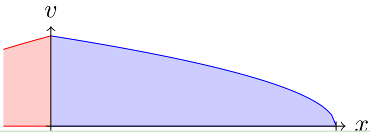
$$\left(((?Q; a := A) \cup a := -b); t := 0; \{x' = v, v' = a, t' = 1 \,\&\, v \geq 0 \wedge t \leq \varepsilon\}\right)^*$$

Example (Safely stays before traffic light m)

$$v^2 \leq 2b(m - x) \wedge A \geq 0 \wedge b > 0 \rightarrow [car_\varepsilon]\, x \leq m$$



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

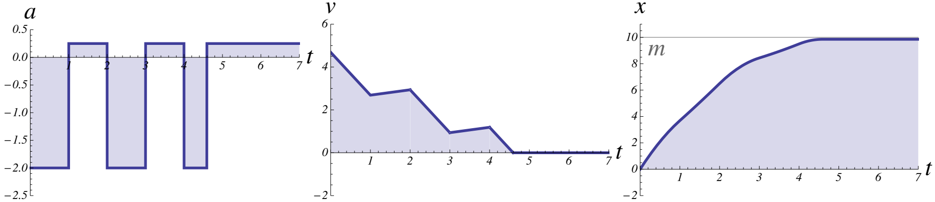


Example (Single car car_ε time-triggered)

$$\Big(\big((?Q; a := A) \cup a := -b\big); t := 0; \{x' = v, v' = a, t' = 1 \,\&\, v \geq 0 \land t \leq \varepsilon\}\Big)^*$$

Example (Safely stays before traffic light m)

$$v^2 \leq 2b(m-x) \land A \geq 0 \land b > 0 \rightarrow [car_\varepsilon]\, x \leq m$$



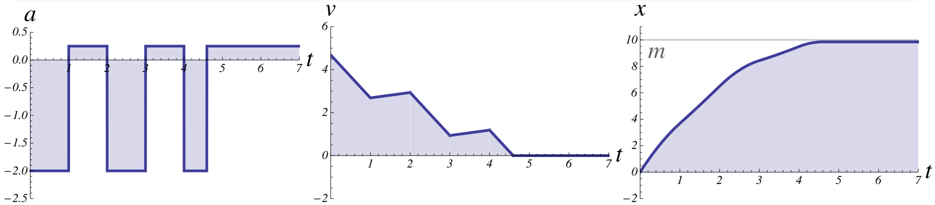
$Q \equiv 2b(m-x) \geq v^2 + (A+b)(A\varepsilon^2 + 2\varepsilon v)$

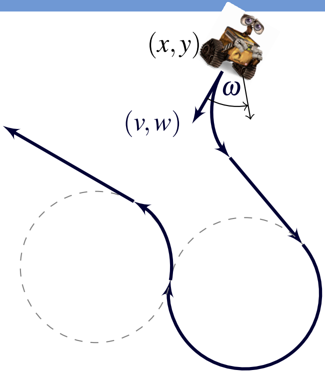
Example (Single car car_ε time-triggered)

$$\Big(\big((?Q;a:=A)\cup a:=-b\big);t:=0;\{x'=v,v'=a,t'=1\,\&\,v\geq 0\wedge t\leq\varepsilon\}\Big)^*$$

Example (Live, can move everywhere)

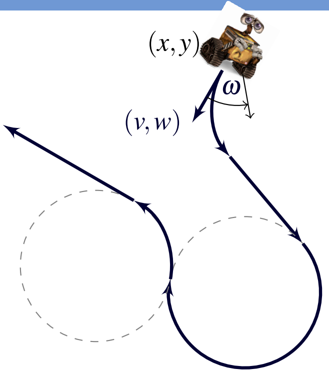
$$\varepsilon>0\wedge A>0\wedge b>0\rightarrow\forall p\exists m\,\langle car_\varepsilon\rangle\,x\geq p$$





(x, y)
ω
(v, w)

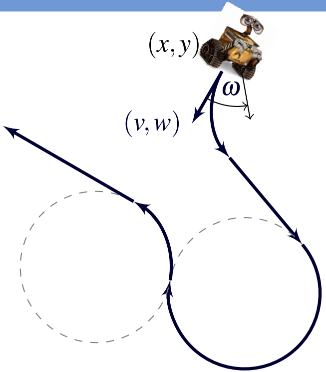
Ex: Runaround Robot



Example (Runaround Robot)

$$\big((\omega := -1 \cup \omega := 1 \cup \omega := 0);$$
$$\{x' = v, y' = w, v' = \omega w, w' = -\omega v\}\big)^*$$

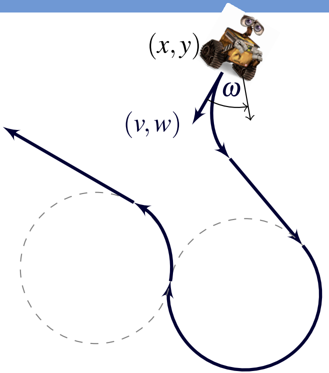
Ex: Runaround Robot



Example (Runaround Robot)

$$(x,y) \neq o \rightarrow \Big[\big((\omega := -1 \cup \omega := 1 \cup \omega := 0);$$
$$\{x' = v, y' = w, v' = \omega w, w' = -\omega v\}\big)^*\Big]\,(x,y) \neq o$$

Ex: Runaround Robot



Example (Runaround Robot)

$$(x,y) \neq o \to \Big[\big((?Q_{-1}; \omega := -1 \cup ?Q_1; \omega := 1 \cup ?Q_0; \omega := 0);$$
$$\{x' = v, y' = w, v' = \omega w, w' = -\omega v\}\big)^*\Big]\,(x,y) \neq o$$

Differential Dynamic Logic dL: Axiomatization

equations of truth

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

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

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

$[\cup]$ $[\alpha \cup \beta]P \leftrightarrow [\alpha]P \wedge [\beta]P$

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

$[^*]$ $[\alpha^*]P \leftrightarrow P \wedge [\alpha][\alpha^*]P$

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

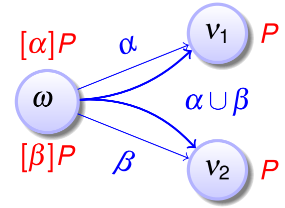
I $[\alpha^*]P \leftrightarrow P \wedge [\alpha^*](P \rightarrow [\alpha]P)$

laws of logic of
laws of physics

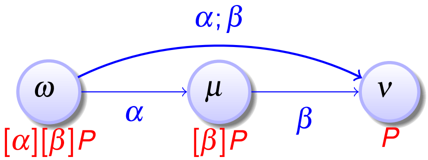
C $[\alpha^*]\forall v{>}0\,(P(v) \rightarrow \langle\alpha\rangle P(v-1)) \rightarrow \forall v\,(P(v) \rightarrow \langle\alpha^*\rangle \exists v{\leq}0\, P(v))$

LICS'12,JAR'17

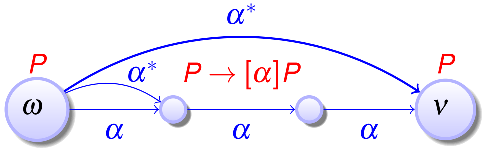
$$[\alpha \cup \beta]P \leftrightarrow [\alpha]P \land [\beta]P$$



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



$$[\alpha^*]P \leftrightarrow P \land [\alpha^*](P \rightarrow [\alpha]P)$$



The lion's share of understanding comes from understanding what does change (variants/progress measures) and what doesn't change (invariants).

Invariants are a fundamental force of CS

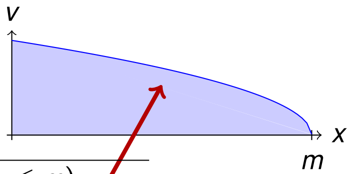
Variants are another fundamental force of CS

$J(x,v) \equiv x \leq m$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{J(x,v) \vdash v^2 \leq 2b(m-x)}{J(x,v) \vdash \forall t{\geq}0\,(-\frac{b}{2}t^2 + vt + x \leq m)}\text{QE}}{J(x,v) \vdash \forall t{\geq}0\,[x := -\frac{b}{2}t^2 + vt + x]J(x,v)}[:=]}{J(x,v) \vdash [x' = v, v' = -b]J(x,v)}[']}{J(x,v) \vdash [a := -b][x' = v, v' = a]J(x,v)}[:=]}{J(x,v) \vdash [a := -b;(x' = v, v' = a)]J(x,v)}[;]
$$

1. $\Gamma \vdash \Delta$ shape of conjecture to prove — sequent
2. Γ is list of all available assumptions — antecedent
3. Δ disjunction needs to be proved from assumptions Γ — succedent
4. Proof reduces desired conclusion (at the bottom) to premises with remaining subgoals (top) until no more subgoals ($*$)

$J(x,v) \equiv v^2 \le 2b(m-x)$



$$J(x,v) \vdash v^2 \le 2b(m-x)$$

$$\text{QE}\quad J(x,v) \vdash \forall t{\ge}0\,(-\tfrac{b}{2}t^2 + vt + x \le m)$$

$$[:=]\quad J(x,v) \vdash \forall t{\ge}0\,[x := -\tfrac{b}{2}t^2 + vt + x]J(x,v)$$

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

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

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

Complete Proof Theory of Hybrid Systems

Theorem (Sound & Complete) (JAR'08, LICS'12, JAR'17)

dL *calculus is a sound & complete axiomatization of hybrid systems relative to either differential equations* ***or*** *to discrete dynamics.*

Corollary (Complete Proof-theoretical Bridge)

proving continuous = proving hybrid = proving discrete

$$\vDash P \text{ iff } \text{FOD} \vdash_{\text{dL}} P$$

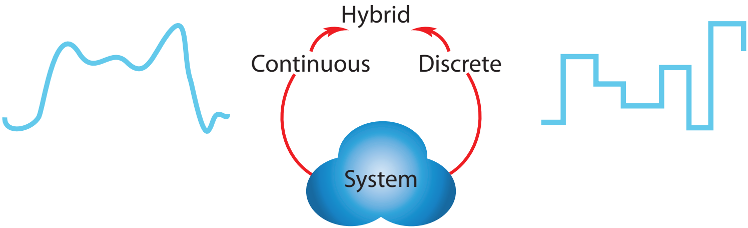
Complete Proof Theory of Hybrid Systems

Theorem (Sound & Complete) (JAR'08, LICS'12, JAR'17)

dL *calculus is a sound & complete axiomatization of hybrid systems relative to either differential equations* ***or*** *to discrete dynamics.*

Corollary (Complete Proof-theoretical Bridge)

proving continuous = proving hybrid = proving discrete

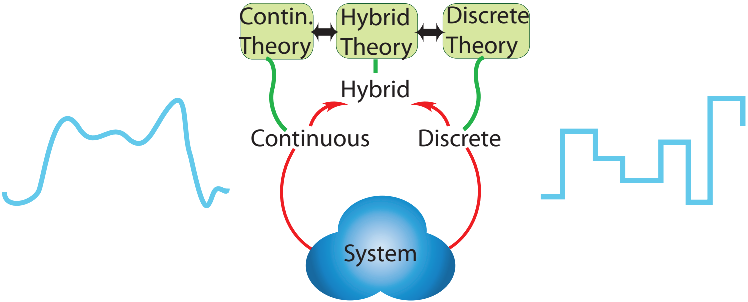


Theorem (Sound & Complete) (JAR'08, LICS'12, JAR'17)

dL *calculus is a sound & complete axiomatization of hybrid systems relative to either differential equations* ***or*** *to discrete dynamics.*

Corollary (Complete Proof-theoretical Bridge)

proving continuous = proving hybrid = proving discrete

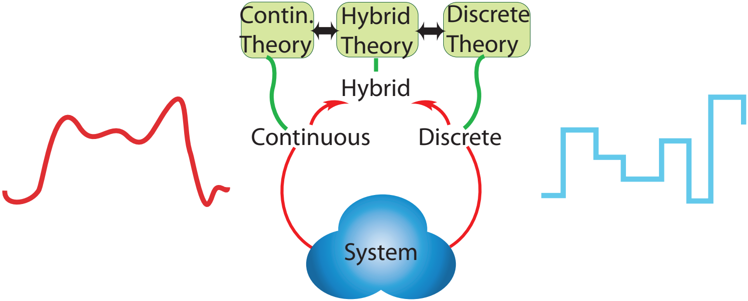


Theorem (Sound & Complete) (JAR'08, LICS'12, JAR'17)

dL *calculus is a sound & complete axiomatization of hybrid systems relative to either differential equations* ***or*** *to discrete dynamics.*

Corollary (Complete Proof-theoretical Bridge)

proving continuous = proving hybrid = proving discrete

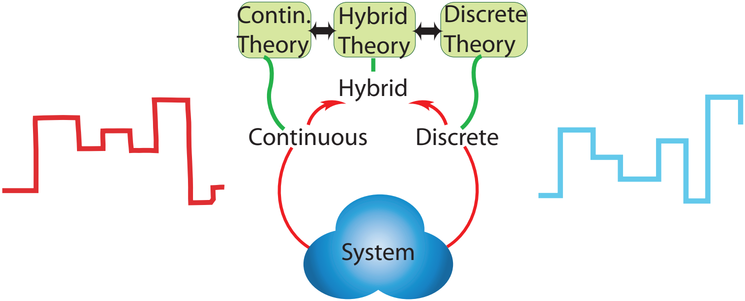


Theorem (Sound & Complete) (JAR'08, LICS'12, JAR'17)

dL *calculus is a sound & complete axiomatization of hybrid systems relative to either differential equations* ***or*** *to discrete dynamics.*

Corollary (Complete Proof-theoretical Bridge)

proving continuous = proving hybrid = proving discrete



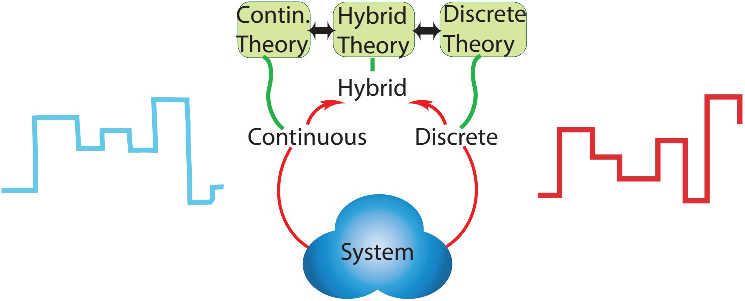
Complete Proof Theory of Hybrid Systems

Theorem (Sound & Complete) (JAR'08, LICS'12, JAR'17)

dL *calculus is a sound & complete axiomatization of hybrid systems relative to either differential equations* ***or*** *to discrete dynamics.*

Corollary (Complete Proof-theoretical Bridge)

proving continuous = proving hybrid = proving discrete



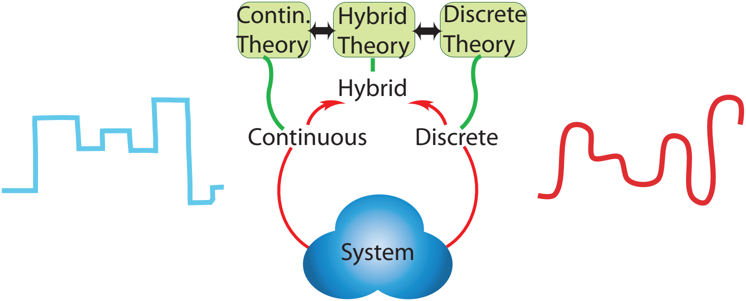
Complete Proof Theory of Hybrid Systems

Theorem (Sound & Complete) (JAR'08, LICS'12, JAR'17)

dL *calculus is a sound & complete axiomatization of hybrid systems relative to either differential equations* ***or*** *to discrete dynamics.*

Corollary (Complete Proof-theoretical Bridge)

proving continuous = proving hybrid = proving discrete

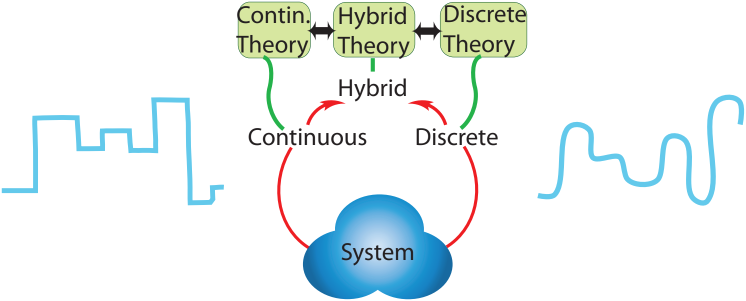


Theorem (Sound & Complete) (JAR'08, LICS'12, JAR'17)

dL *calculus is a sound & complete axiomatization of hybrid systems relative to either differential equations* ***or*** *to discrete dynamics.*

Corollary (Complete Proof-theoretical Bridge)

proving continuous = proving hybrid = proving discrete



Theorem (Sound & Complete) (JAR'08, LICS'12, JAR'17)

dL *calculus is a sound & complete axiomatization of hybrid systems relative to either differential equations* ***or*** *to discrete dynamics.*

Corollary (Complete Proof-theoretical Bridge)

proving continuous = proving hybrid = proving discrete

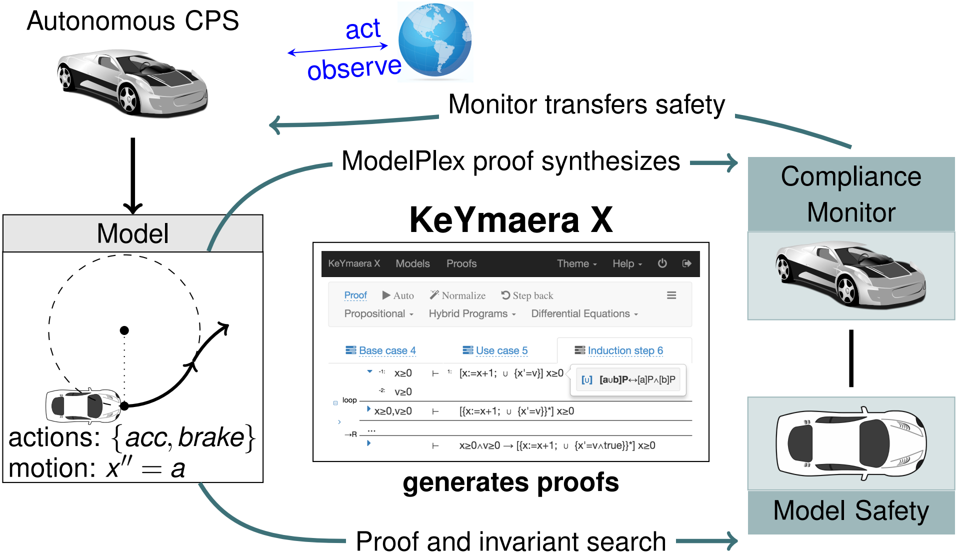
Theorem (Equi-expressibility) (LICS'12)

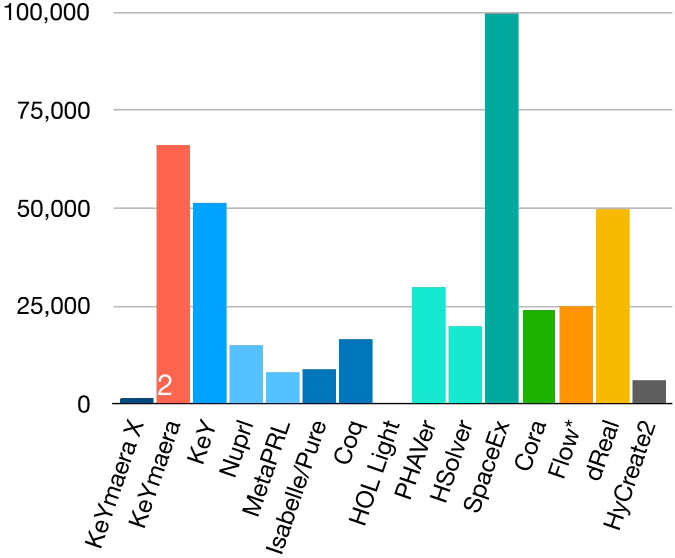
$$\forall P \in \mathrm{dL}\ \exists P^\flat \in FOD\ \vDash P \leftrightarrow P^\flat$$
$$\forall P \in \mathrm{dL}\ \exists P^\# \in DL\ \ \vDash P \leftrightarrow P^\#$$

Theorem (Relative Decidability) (LICS'12)

Validity of dL *sentences is decidable relative to FOD or DL.*

Approach: Safety Proofs for Autonomous CPS





Disclaimer: Self-reported estimates of the soundness-critical lines of code + rules

Theorem (Soundness) replace all occurrences of $p(\cdot)$

$$\text{US}\ \frac{\phi}{\sigma(\phi)}$$

provided $FV(\sigma|_{\Sigma(\theta)}) \cap BV(\otimes(\cdot)) = \emptyset$ *for each operation* $\otimes(\theta)$ *in* ϕ

i.e. bound variables $U = \text{BV}(\otimes(\cdot))$ of **no** operator $\otimes$
are free in the substitution on its argument θ (U-admissible)

$$\text{US}\ \frac{[a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \land [b]p(\bar{x})}{[x := x+1 \cup x' = 1]x \geq 0 \leftrightarrow [x := x+1]x \geq 0 \land [x' = 1]x \geq 0}$$

Theorem (Soundness) replace all occurrences of $p(\cdot)$

$$US \; \frac{\phi}{\sigma(\phi)}$$

provided $FV(\sigma|_{\Sigma(\theta)}) \cap BV(\otimes(\cdot)) = \emptyset$ *for each operation* $\otimes(\theta)$ *in* ϕ

i.e. bound variables $U = \mathrm{BV}(\otimes(\cdot))$ of **no** operator $\otimes$
are free in the substitution on its argument θ (U-admissible)

$$\frac{[v := f]p(v) \leftrightarrow p(f)}{[v := -x][x' = v]\,x \geq 0 \leftrightarrow [x' = -x]\,x \geq 0}$$

Uniform Substitution

Theorem (Soundness) replace all occurrences of $p(\cdot)$

Modular interface: Prover vs. Logic

$$\text{US}\ \frac{\phi}{\sigma(\phi)}$$

provided $FV(\sigma|_{\Sigma(\theta)}) \cap BV(\otimes(\cdot)) = \emptyset$ *for each operation* $\otimes(\theta)$ *in* ϕ

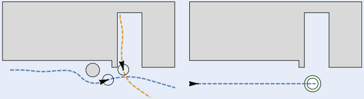
i.e. bound variables $U = \mathrm{BV}(\otimes(\cdot))$ of **no** operator $\otimes$
are free in the substitution on its argument θ (U-admissible)

If you bind a free variable, you go to logic jail!

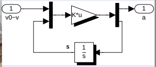
$$\frac{[v := f]p(v) \leftrightarrow p(f)}{[v := -x][x' = v]\, x \geq 0 \leftrightarrow [x' = -x]\, x \geq 0}$$

Clash

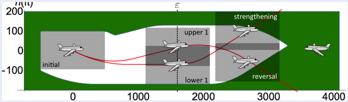
Obstacle Avoidance + Ground Navigation



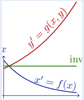
Train Control Brakes



Airborne Collision Avoidance (ACAS X)

Ship Cooling



NSF DARPA

BOSCH SIEMENS

APL
Johns Hopkins
Applied Physics Laboratory

Outline (Dynamic Logic for Dynamical Systems)

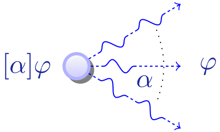
Acknowledgments

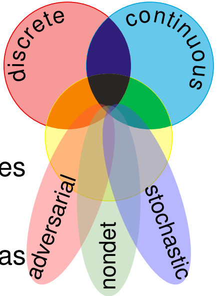
Logical Systems Lab at Carnegie Mellon University, Computer Science
Yong Kiam Tan, Brandon Bohrer, Nathan Fulton, Sarah Loos, Katherine Cordwell
Stefan Mitsch, Khalil Ghorbal, Jean-Baptiste Jeannin, Andrew Sogokon

Logical foundations make a big difference for CPS, and vice versa

differential dynamic logic

$\mathsf{dL} = \mathsf{DL} + \mathsf{HP}$





- Strong analytic foundations
- Practical reasoning advances
- Significant applications
- Catalyze many science areas

1. Multi-dynamical systems
2. Combine simple dynamics
3. Tame complexity
4. Complete axiomatization

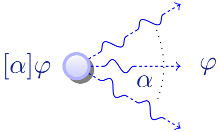
Numerous wonders remain to be discovered

Dynamic Logic for Dynamical Systems

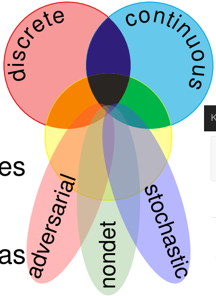
Logical foundations make a big difference for CPS, and vice versa

differential dynamic logic

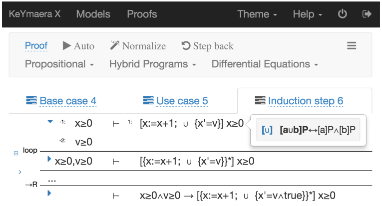
$\mathsf{dL} = \mathsf{DL} + \mathsf{HP}$



- Strong analytic foundations
- Practical reasoning advances
- Significant applications
- Catalyze many science areas



KeYmaera X

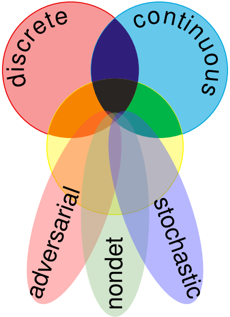


Numerous wonders remain to be discovered

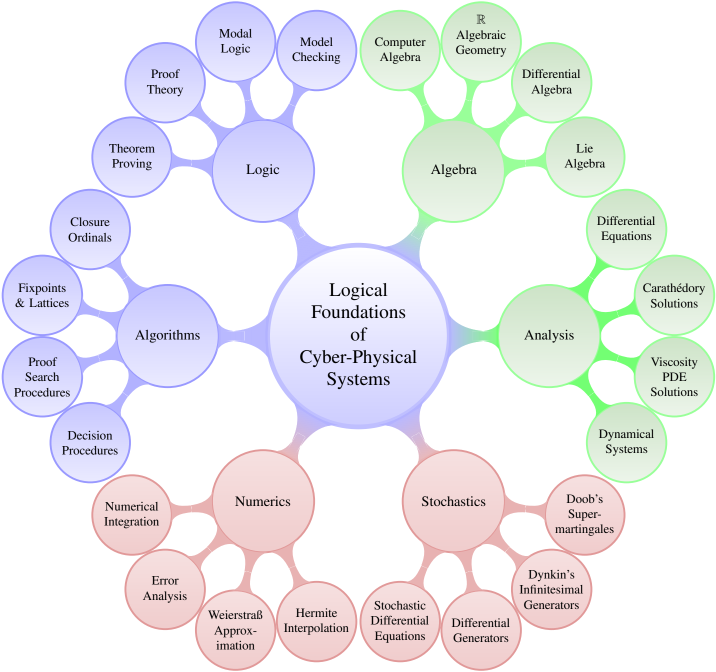
Future CPS Challenges

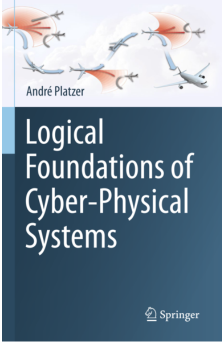
Numerous wonders remain to be discovered

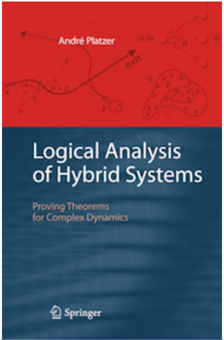
- Verified CPS implementations by ModelPlex FMSD'16
- Verified CPS execution PLDI'18
- CPS proof and tactic languages+libraries ITP'17
- Big CPS built from safe components STTT'18
- Real arithmetic, scalable and verified FM'21
- Automatic ODE proofs JACM'20
- Invariant generation FMSD'21
- Safe AI autonomy in CPS AAAI'18 TACAS'19
- Correct model transformation FM'14
- Refinement + system property proofs LICS'16
- CPS information flow LICS'18
- Hybrid games TOCL'15



CPSs deserve proofs as safety evidence!


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


André Platzer
Logical Foundations of Cyber-Physical Systems
Springer


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

André Platzer.
Logic & proofs for cyber-physical systems.
In Nicola Olivetti and Ashish Tiwari, editors, *IJCAR*, volume 9706 of *LNCS*, pages 15–21, Cham, 2016. Springer.
doi:10.1007/978-3-319-40229-1_3.

André Platzer.
Logics of dynamical systems.
In LICS [16], pages 13–24.
doi:10.1109/LICS.2012.13.

André Platzer.
Logical Foundations of Cyber-Physical Systems.
Springer, Cham, 2018.
URL: http://www.springer.com/978-3-319-63587-3, doi:10.1007/978-3-319-63588-0.

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.
A complete uniform substitution calculus for differential dynamic logic.
J. Autom. Reas., 59(2):219–265, 2017.
doi:10.1007/s10817-016-9385-1.

André Platzer.
The complete proof theory of hybrid systems.
In LICS [16], pages 541–550.
doi:10.1109/LICS.2012.64.

André Platzer.
Differential game logic.
ACM Trans. Comput. Log., 17(1):1:1–1:51, 2015.
doi:10.1145/2817824.

Stefan Mitsch, Khalil Ghorbal, David Vogelbacher, and André Platzer.
Formal verification of obstacle avoidance and navigation of ground robots.
I. J. Robotics Res., 36(12):1312–1340, 2017.
doi:10.1177/0278364917733549.

André Platzer and Jan-David Quesel.
European Train Control System: A case study in formal verification.
In Karin Breitman and Ana Cavalcanti, editors, *ICFEM*, volume 5885 of *LNCS*, pages 246–265, Berlin, 2009. Springer.
doi:10.1007/978-3-642-10373-5_13.

Stefan Mitsch, Marco Gario, Christof J. Budnik, Michael Golm, and André Platzer.
Formal verification of train control with air pressure brakes.
In Alessandro Fantechi, Thierry Lecomte, and Alexander Romanovsky, editors, *RSSRail*, volume 10598 of *LNCS*, pages 173–191. Springer, 2017.
doi:10.1007/978-3-319-68499-4_12.

Jean-Baptiste Jeannin, Khalil Ghorbal, Yanni Kouskoulas, Aurora Schmidt, Ryan Gardner, Stefan Mitsch, and André Platzer.
A formally verified hybrid system for safe advisories in the next-generation airborne collision avoidance system.
STTT, 19(6):717–741, 2017.
doi:10.1007/s10009-016-0434-1.

Stefan Mitsch and André Platzer.
ModelPlex: Verified runtime validation of verified cyber-physical system models.
Form. Methods Syst. Des., 49(1-2):33–74, 2016.
Special issue of selected papers from RV'14.
doi:10.1007/s10703-016-0241-z.

Nathan Fulton, Stefan Mitsch, Brandon Bohrer, and André Platzer.
Bellerophon: Tactical theorem proving for hybrid systems.
In Mauricio Ayala-Rincón and César A. Muñoz, editors, *ITP*, volume 10499 of *LNCS*, pages 207–224. Springer, 2017.
doi:10.1007/978-3-319-66107-0_14.

Matias Scharager, Katherine Cordwell, Stefan Mitsch, and André Platzer.
Verified quadratic virtual substitution for real arithmetic.
In Marieke Huisman, Corina S. Pasareanu, and Naijun Zhan, editors, *FM*, volume 13047 of *LNCS*, pages 200–217. Springer, 2021.
doi:10.1007/978-3-030-90870-6_11.

André Platzer.

Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics.
Springer, Heidelberg, 2010.
URL: http://www.springer.com/978-3-642-14508-7, doi:10.1007/978-3-642-14509-4.

Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on, Los Alamitos, 2012. IEEE.