Computing Differential Invariants of Hybrid Systems as Fixedpoints

André Platzer[1,2] Edmund M. Clarke[2]

[1]University of Oldenburg, Department of Computing Science, Germany

[2]Carnegie Mellon University, Computer Science Department, Pittsburgh, PA

Computer Aided Verification, Princeton, July 2008

Carnegie Mellon

Carl von Ossietzky Universität Oldenburg

Albert-Ludwigs-Universität Freiburg

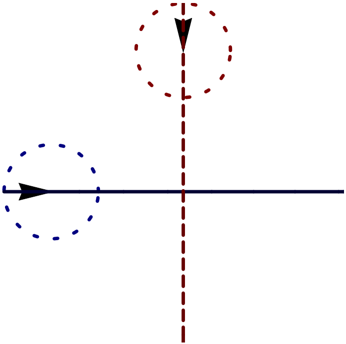
Universität des Saarlandes

Deutsche Forschungsgemeinschaft DFG

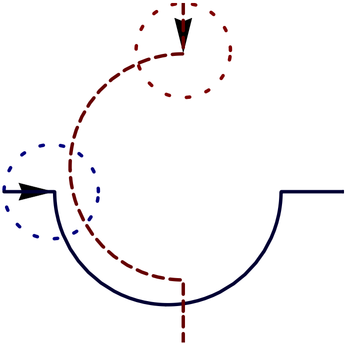
Outline

1. Motivation
2. Compositional Verification Logic d$\mathcal{L}$
3. Decompositional Inductive Verification of Hybrid Systems
    - Verification by Symbolic Decomposition
    - Discrete Induction
    - Differential Induction
4. Computing Differential Invariants by Combining Local Fixedpoints
    - Local Fixedpoints & Differential Saturation
    - Global Fixedpoints & Interplay
5. Case Studies & Experimental Results
6. Conclusions & Future Work

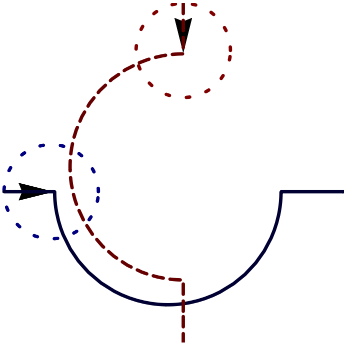
Hybrid Systems & Air Traffic Control



Hybrid Systems & Air Traffic Control



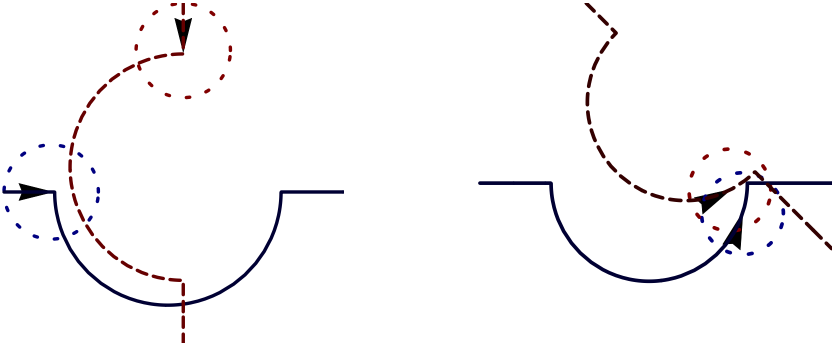
Hybrid Systems & Air Traffic Control



Hybrid Systems

continuous evolution along differential equations + discrete change

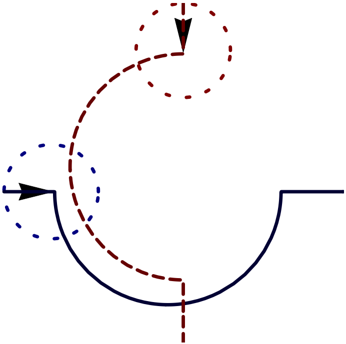
Verification of Hybrid Systems & Air Traffic Control

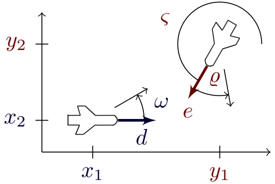


Hybrid Systems

continuous evolution along differential equations + discrete change

Verification of Hybrid Systems & Air Traffic Control



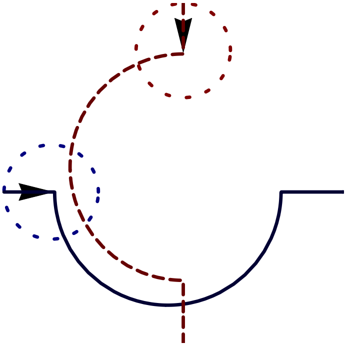


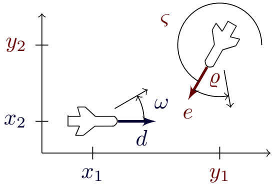
$$\begin{bmatrix} x_1' = -v_1 + v_2 \cos\vartheta + \omega x_2 \\ x_2' = v_2 \sin\vartheta - \omega x_1 \\ \vartheta' = \varrho - \omega \end{bmatrix}$$

Hybrid Systems

continuous evolution along differential equations + discrete change

Verification of Hybrid Systems & Air Traffic Control



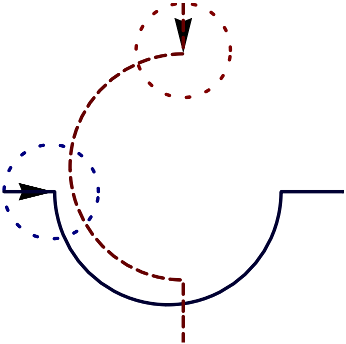


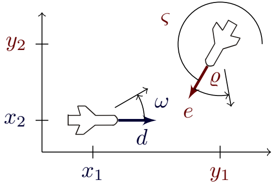
$$\begin{bmatrix} x_1' = -v_1 + v_2 \cos\vartheta + \omega x_2 \\ x_2' = v_2 \sin\vartheta - \omega x_1 \\ \vartheta' = \varrho - \omega \end{bmatrix}$$

Example (“Solving” differential equations)

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

Verification of Hybrid Systems & Air Traffic Control



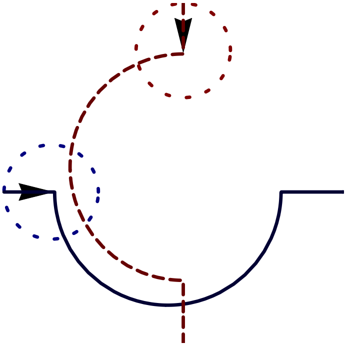


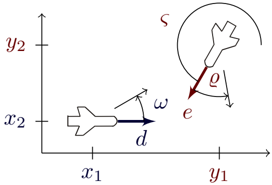
$$\begin{bmatrix} x_1' = -v_1 + v_2 \cos\vartheta + \omega x_2 \\ x_2' = v_2 \sin\vartheta - \omega x_1 \\ \vartheta' = \varrho - \omega \end{bmatrix}$$

Example ("Solving" differential equations)

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

Verification of Hybrid Systems & Air Traffic Control

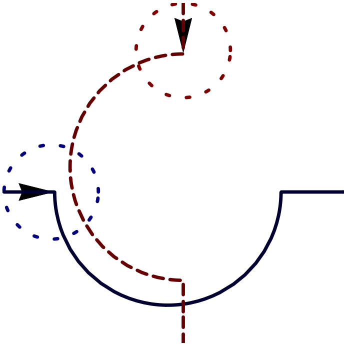


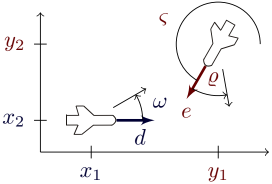


$$\begin{bmatrix} x_1' = -v_1 + v_2 \cos\vartheta + \omega x_2 \\ x_2' = v_2 \sin\vartheta - \omega x_1 \\ \vartheta' = \varrho - \omega \end{bmatrix}$$

Symbolic Verification	Numerical Verification
× constant/nilpotent dynamics	✓ challenging dynamics
× otherwise “no” solutions	× approximation errors
✓ sound	× unsound, . . . see [PC07]

Verification of Hybrid Systems & Air Traffic Control





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

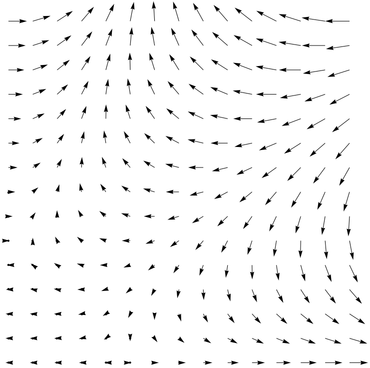
How To Get What We Really Need?

- ✓ challenging dynamics, e.g., curved flight
- ✓ automatic verification
- ✓ sound

Idea: Exploit Vector Field of Differential Equations

“Definition” (Differential Invariant)

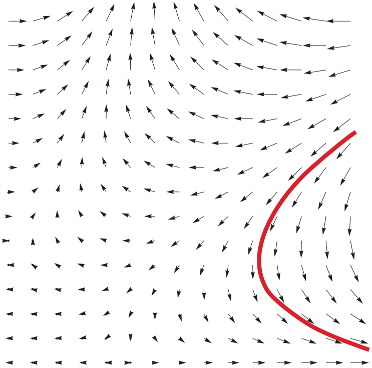
“Property that remains true in the direction of the dynamics”



Idea: Exploit Vector Field of Differential Equations

"Definition" (Differential Invariant)

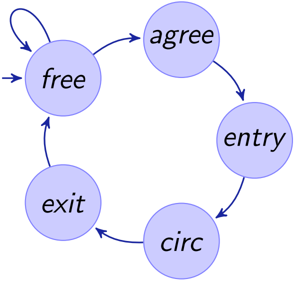
"Property that remains true in the direction of the dynamics"

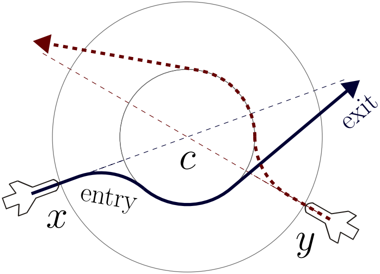


Idea: Exploit Vector Field of Differential Equations

"Definition" (Differential Invariant)

"Property that remains true in the direction of the dynamics"

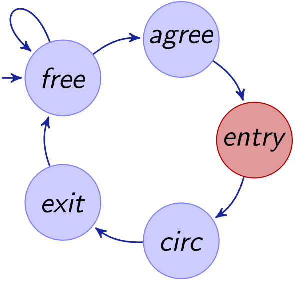




Idea: Exploit Vector Field of Differential Equations

“Definition” (Differential Invariant)

“Property that remains true in the direction of the dynamics”

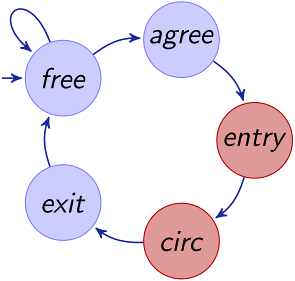


- How to find diff. invariants?

Idea: Exploit Vector Field of Differential Equations

"Definition" (Differential Invariant)

"Property that remains true in the direction of the dynamics"

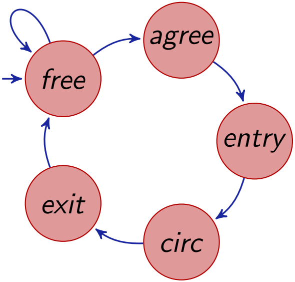


- How to find diff. invariants?
- How do diff. invariants fit together?

Idea: Exploit Vector Field of Differential Equations

"Definition" (Differential Invariant)

"Property that remains true in the direction of the dynamics"

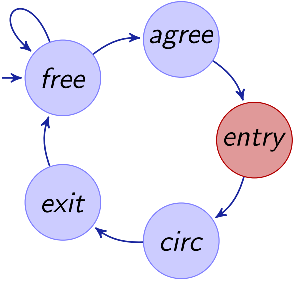


- How to find diff. invariants?
- How do diff. invariants fit together?
- Find all at once? 10000-dim

Idea: Exploit Vector Field of Differential Equations

"Definition" (Differential Invariant)

"Property that remains true in the direction of the dynamics"

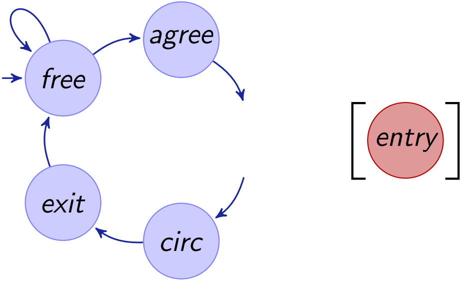


- How to find diff. invariants?
- How do diff. invariants fit together?
- Find local diff. invariants?

Idea: Exploit Vector Field of Differential Equations

"Definition" (Differential Invariant)

"Property that remains true in the direction of the dynamics"

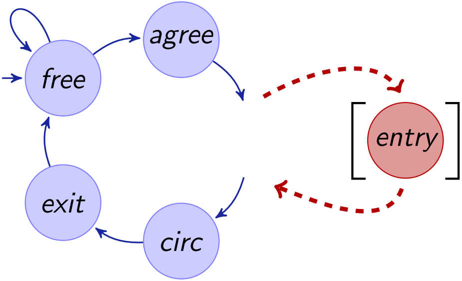


- How to find diff. invariants?
- How do diff. invariants fit together?
- Find local diff. invariants?

Idea: Exploit Vector Field of Differential Equations

"Definition" (Differential Invariant)

"Property that remains true in the direction of the dynamics"

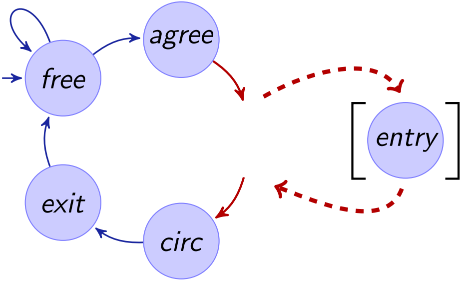


- How to find diff. invariants?
- How do diff. invariants fit together?
- Find local diff. invariants?
- How to put local differential invariants together?

Idea: Exploit Vector Field of Differential Equations

"Definition" (Differential Invariant)

"Property that remains true in the direction of the dynamics"

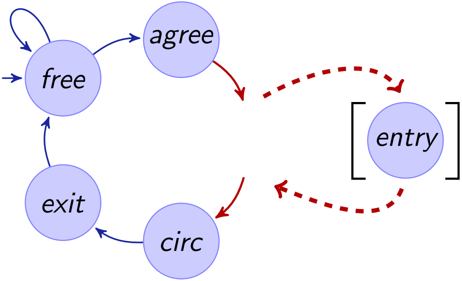


- How to find diff. invariants?
- How do diff. invariants fit together?
- Find local diff. invariants?
- How to put local differential invariants together?
- How do discrete transitions fit?

Idea: Exploit Vector Field of Differential Equations

"Definition" (Differential Invariant)

"Property that remains true in the direction of the dynamics"



- How to find diff. invariants?
- How do diff. invariants fit together?
- Find local diff. invariants?
- How to put local differential invariants together?
- How do discrete transitions fit?
- What does "fit" really mean?

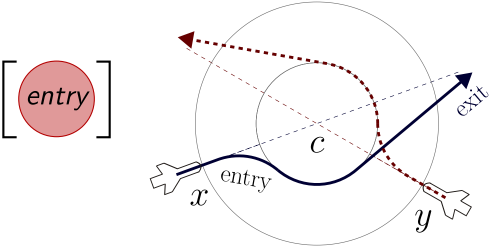
Outline

1. Motivation
2. Compositional Verification Logic d$\mathcal{L}$
3. Decompositional Inductive Verification of Hybrid Systems
    - Verification by Symbolic Decomposition
    - Discrete Induction
    - Differential Induction
4. Computing Differential Invariants by Combining Local Fixedpoints
    - Local Fixedpoints & Differential Saturation
    - Global Fixedpoints & Interplay
5. Case Studies & Experimental Results
6. Conclusions & Future Work

Outline

1. Motivation
2. Compositional Verification Logic d$\mathcal{L}$
3. Decompositional Inductive Verification of Hybrid Systems
   - Verification by Symbolic Decomposition
   - Discrete Induction
   - Differential Induction
4. Computing Differential Invariants by Combining Local Fixedpoints
   - Local Fixedpoints & Differential Saturation
   - Global Fixedpoints & Interplay
5. Case Studies & Experimental Results
6. Conclusions & Future Work

Verification Logic for Hybrid (Sub-)Systems: d$\mathcal{L}$

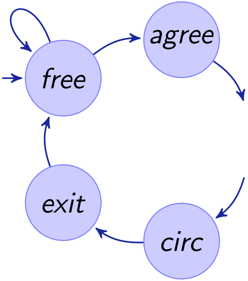


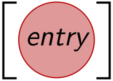
Example

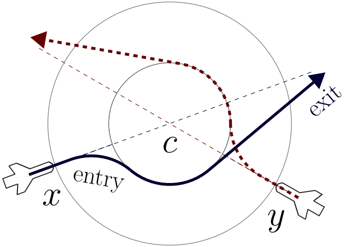
$$\mathit{safe} \land \mathit{far} \quad \rightarrow \quad [\mathit{entry}](\mathit{safe} \land \mathit{tangential})$$

$$\text{where} \quad \mathit{safe} \quad \equiv (x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2$$

Verification Logic for Hybrid (Sub-)Systems: d$\mathcal{L}$



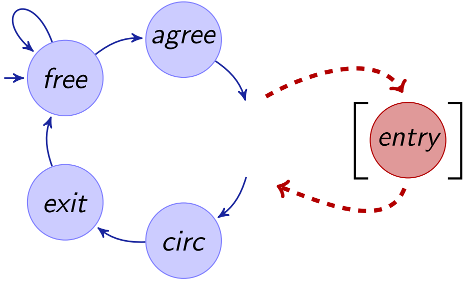


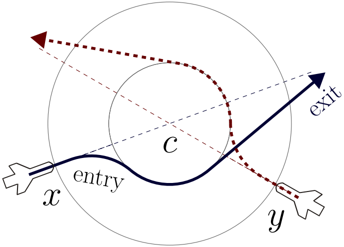


Example

$$\begin{aligned} \mathit{safe} \land \mathit{far} &\rightarrow [\mathit{entry}](\mathit{safe} \land \mathit{tangential}) \\ \mathit{safe} \land \mathit{tangential} &\rightarrow [\mathit{other\ subsystem}]\mathit{safe} \\ \text{where}\quad \mathit{safe} &\equiv (x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2 \end{aligned}$$

Verification Logic for Hybrid (Sub-)Systems: d$\mathcal{L}$





Example

$$\left.\begin{aligned} \mathit{safe} \land \mathit{far} &\rightarrow [\mathit{entry}](\mathit{safe} \land \mathit{tangential}) \\ \mathit{safe} \land \mathit{tangential} &\rightarrow [\mathit{other\ subsystem}]\mathit{safe} \end{aligned}\right\} \text{conjunction}$$

$$\text{where} \quad \mathit{safe} \equiv (x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2$$

Verification Logic for Compositional Hybrid Programs

Definition (d$\mathcal{L}$ Formula ϕ)

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

with terms θ_1, θ_2 of nonlinear real arithmetic $(+, \cdot)$

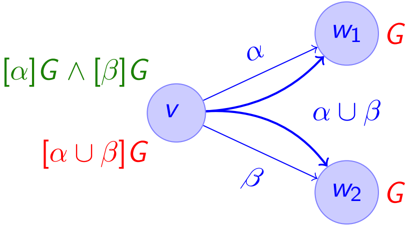
Definition (Hybrid program α)

$x' = f(x) \wedge H$	(continuous evolution)	
$x := f(x)$	(discrete jump)	jump & test
$?H$	(conditional execution)	
$\alpha; \beta$	(seq. composition)	Kleene algebra
$\alpha \cup \beta$	(nondet. choice)	
α^*	(nondet. repetition)	

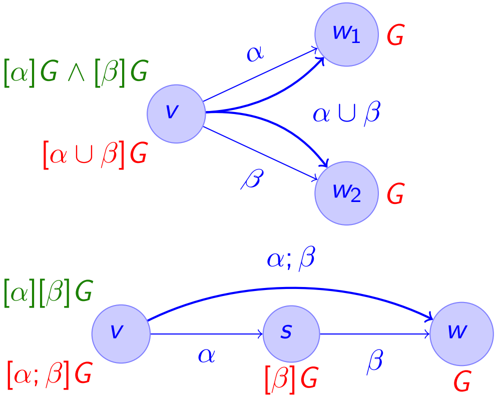
Outline

1. Motivation
2. Compositional Verification Logic d$\mathcal{L}$
3. Decompositional Inductive Verification of Hybrid Systems
   - Verification by Symbolic Decomposition
   - Discrete Induction
   - Differential Induction
4. Computing Differential Invariants by Combining Local Fixedpoints
   - Local Fixedpoints & Differential Saturation
   - Global Fixedpoints & Interplay
5. Case Studies & Experimental Results
6. Conclusions & Future Work

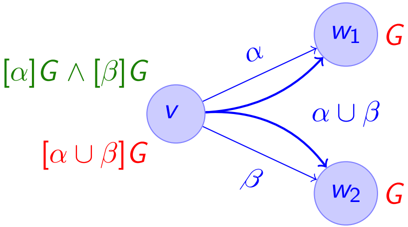
Verification by Symbolic Decomposition

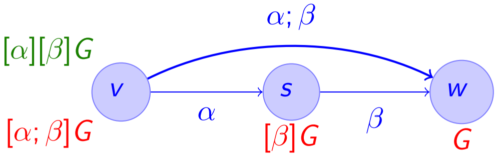


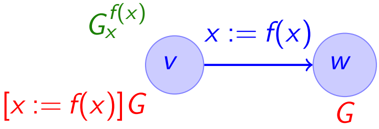
Verification by Symbolic Decomposition



Verification by Symbolic Decomposition







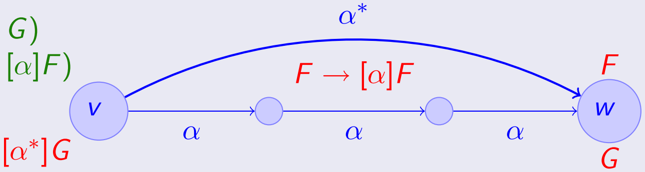
Verification by Discrete and Differential Induction

Definition (Discrete Invariant F)

F

$\forall_{cl}(F \rightarrow G)$

$\forall_{cl}(F \rightarrow [\alpha]F)$



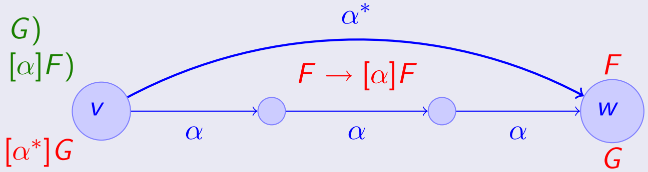
Verification by Discrete and Differential Induction

Definition (Discrete Invariant F)

F

$\forall_{cl}(F \rightarrow G)$

$\forall_{cl}(F \rightarrow [\alpha]F)$

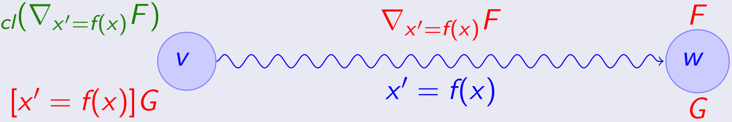


Definition (Differential Invariant F)

F

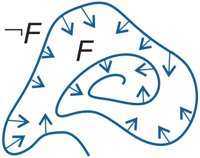
$\forall_{cl}(F \rightarrow G)$

$\forall_{cl}(\nabla_{x'=f(x)} F)$



Verification by Discrete and Differential Induction

$$\nabla_{x_1'=f_1(x)\wedge\ldots\wedge x_n'=f_n(x)} F \quad \text{is} \quad \bigwedge_{(b\geq c)\in F}\left(\sum_{i=1}^{n}\frac{\partial b}{\partial x_i}f_i(x) \geq \sum_{i=1}^{n}\frac{\partial c}{\partial x_i}f_i(x)\right)$$

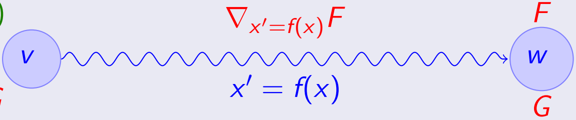


Definition (Differential Invariant F)

F

$\forall_{cl}(F \rightarrow G)$

$\forall_{cl}(\nabla_{x'=f(x)}F)$



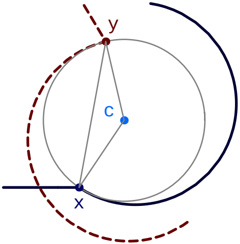
$[x' = f(x)]G$

Outline

1. Motivation
2. Compositional Verification Logic $d\mathcal{L}$
3. Decompositional Inductive Verification of Hybrid Systems
   - Verification by Symbolic Decomposition
   - Discrete Induction
   - Differential Induction
4. Computing Differential Invariants by Combining Local Fixedpoints
   - Local Fixedpoints & Differential Saturation
   - Global Fixedpoints & Interplay
5. Case Studies & Experimental Results
6. Conclusions & Future Work

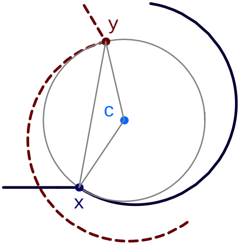
Differential Induction for Roundabout Mode

$$\overline{[x_1' = d_1 \wedge d_1' = -\omega d_2 \wedge x_2' = d_2 \wedge d_2' = \omega d_1 \ldots](x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2}$$



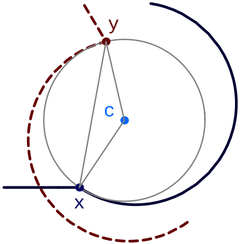
Differential Induction for Roundabout Mode

$$\dfrac{\overline{\frac{\partial\|x-y\|^2}{\partial x_1}x_1' + \frac{\partial\|x-y\|^2}{\partial y_1}y_1' + \frac{\partial\|x-y\|^2}{\partial x_2}x_2' + \frac{\partial\|x-y\|^2}{\partial y_2}y_2' \geq \frac{\partial p^2}{\partial x_1}x_1' \ldots}}{[x_1' = d_1 \land d_1' = -\omega d_2 \land x_2' = d_2 \land d_2' = \omega d_1 \ldots](x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2}$$



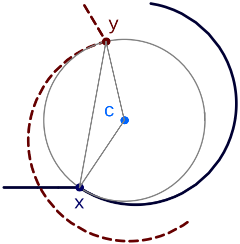
Differential Induction for Roundabout Mode

$$\frac{\frac{\partial\|x-y\|^2}{\partial x_1}x_1' + \frac{\partial\|x-y\|^2}{\partial y_1}y_1' + \frac{\partial\|x-y\|^2}{\partial x_2}x_2' + \frac{\partial\|x-y\|^2}{\partial y_2}y_2' \geq \frac{\partial p^2}{\partial x_1}x_1' \dots}{[x_1' = d_1 \land d_1' = -\omega d_2 \land x_2' = d_2 \land d_2' = \omega d_1 \dots](x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2}$$



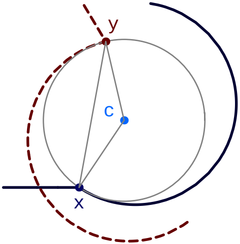
Differential Induction for Roundabout Mode

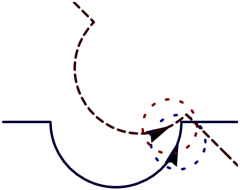
$$\frac{\overline{\frac{\partial\|x-y\|^2}{\partial x_1} d_1 + \frac{\partial\|x-y\|^2}{\partial y_1} e_1 + \frac{\partial\|x-y\|^2}{\partial x_2} d_2 + \frac{\partial\|x-y\|^2}{\partial y_2} e_2 \geq \frac{\partial p^2}{\partial x_1} d_1 \ldots}}{[x_1' = d_1 \land d_1' = -\omega d_2 \land x_2' = d_2 \land d_2' = \omega d_1 \ldots](x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2}$$



Differential Induction for Roundabout Mode

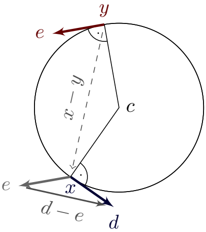
$$\dfrac{\dfrac{\overline{2(x_1-y_1)(d_1-e_1)+2(x_2-y_2)(d_2-e_2)\geq 0}}{\frac{\partial\|x-y\|^2}{\partial x_1}d_1+\frac{\partial\|x-y\|^2}{\partial y_1}e_1+\frac{\partial\|x-y\|^2}{\partial x_2}d_2+\frac{\partial\|x-y\|^2}{\partial y_2}e_2\geq\frac{\partial p^2}{\partial x_1}d_1\ldots}}{[x_1'=d_1\land d_1'=-\omega d_2\land x_2'=d_2\land d_2'=\omega d_1\ldots](x_1-y_1)^2+(x_2-y_2)^2\geq p^2}$$

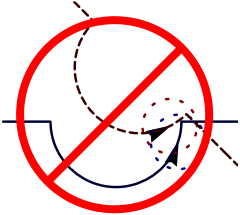




Differential Induction for Roundabout Mode

$$\begin{array}{l}\hline 2(x_1-y_1)(d_1-e_1)+2(x_2-y_2)(d_2-e_2)\geq 0 \\ \hline \frac{\partial\|x-y\|^2}{\partial x_1}d_1+\frac{\partial\|x-y\|^2}{\partial y_1}e_1+\frac{\partial\|x-y\|^2}{\partial x_2}d_2+\frac{\partial\|x-y\|^2}{\partial y_2}e_2\geq\frac{\partial p^2}{\partial x_1}d_1\dots \\ \hline [x_1'=d_1\land d_1'=-\omega d_2\land x_2'=d_2\land d_2'=\omega d_1\dots](x_1-y_1)^2+(x_2-y_2)^2\geq p^2\end{array}$$





$$\begin{array}{l}\hline [d_1'=-\omega d_2\land e_1'=-\omega e_2\land x_2'=d_2\land d_2'=\omega d_1..]d_1-e_1=-\omega(x_2-y_2)\end{array}$$

Differential Induction for Roundabout Mode

$$\dfrac{\dfrac{\dfrac{}{2(x_1-y_1)(d_1-e_1)+2(x_2-y_2)(d_2-e_2)\geq 0}}{\frac{\partial\|x-y\|^2}{\partial x_1}d_1+\frac{\partial\|x-y\|^2}{\partial y_1}e_1+\frac{\partial\|x-y\|^2}{\partial x_2}d_2+\frac{\partial\|x-y\|^2}{\partial y_2}e_2\geq\frac{\partial p^2}{\partial x_1}d_1\dots}}{[x_1'=d_1\land d_1'=-\omega d_2\land x_2'=d_2\land d_2'=\omega d_1\dots](x_1-y_1)^2+(x_2-y_2)^2\geq p^2}$$

Proposition (Differential saturation)

F differential invariant of $[x'=\theta\land H]G$, then

$$[x'=\theta\land H]G \quad \text{iff} \quad [x'=\theta\land H\land F]G$$

$$\dfrac{}{[d_1' = -\omega d_2\land e_1' = -\omega e_2\land x_2'=d_2\land d_2'=\omega d_1..]d_1-e_1=-\omega(x_2-y_2)}$$

Differential Induction for Roundabout Mode

$$\begin{array}{l}
2(x_1-y_1)(-\omega(x_2-y_2))+2(x_2-y_2)\omega(x_1-y_1)\geq 0\\
\hline
2(x_1-y_1)(d_1-e_1)+2(x_2-y_2)(d_2-e_2)\geq 0\\
\hline
\frac{\partial\|x-y\|^2}{\partial x_1}d_1+\frac{\partial\|x-y\|^2}{\partial y_1}e_1+\frac{\partial\|x-y\|^2}{\partial x_2}d_2+\frac{\partial\|x-y\|^2}{\partial y_2}e_2\geq\frac{\partial p^2}{\partial x_1}d_1\ldots\\
\hline
[x_1'=d_1\land d_1'=-\omega d_2\land x_2'=d_2\land d_2'=\omega d_1\ldots](x_1-y_1)^2+(x_2-y_2)^2\geq p^2
\end{array}$$

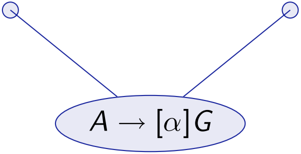
Proposition (Differential saturation)

F differential invariant of $[x'=\theta\land H]G$, then

$$[x'=\theta\land H]G \quad \text{iff} \quad [x'=\theta\land H\land F]G$$

$$\frac{}{[d_1'=\,-\omega d_2\land e_1'=\,-\omega e_2\land x_2'=d_2\land d_2'=\omega d_1..]d_1-e_1=-\omega(x_2-y_2)}$$

Differential Invariants as Fixedpoints

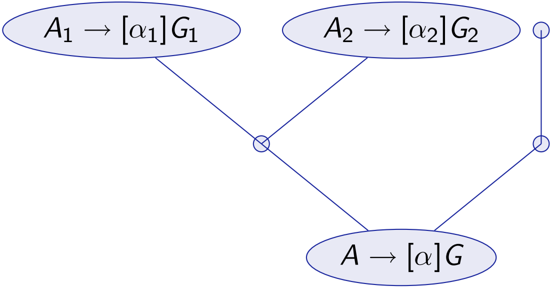


$A \rightarrow [\alpha]G$

[Clarke'79]

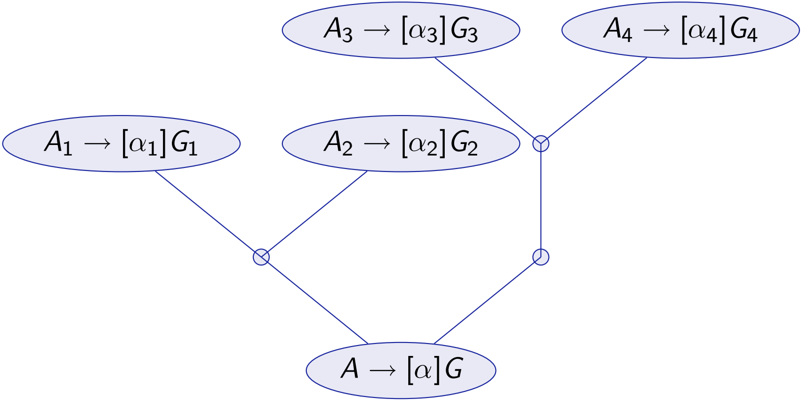
▸ Details

Differential Invariants as Fixedpoints



for $\cup$, ; , := do decompose

Differential Invariants as Fixedpoints



$A_3 \rightarrow [\alpha_3]G_3$

$A_4 \rightarrow [\alpha_4]G_4$

$A_1 \rightarrow [\alpha_1]G_1$

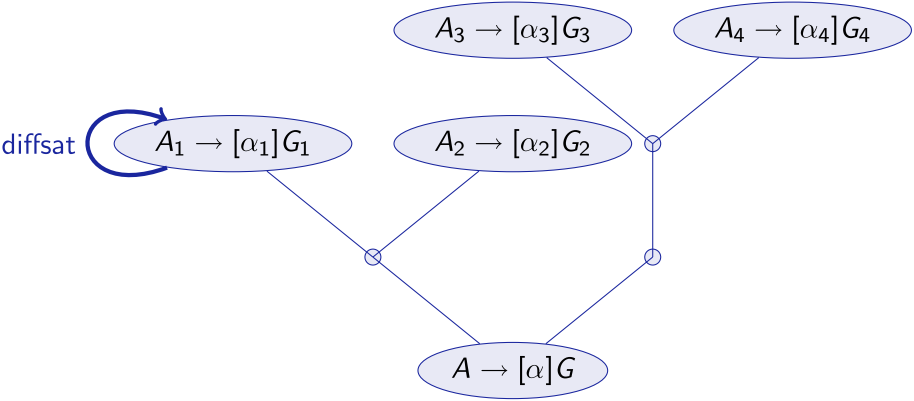
$A_2 \rightarrow [\alpha_2]G_2$

$A \rightarrow [\alpha]G$

for $\cup\,, ;\,, :=$ do decompose

▸ Details

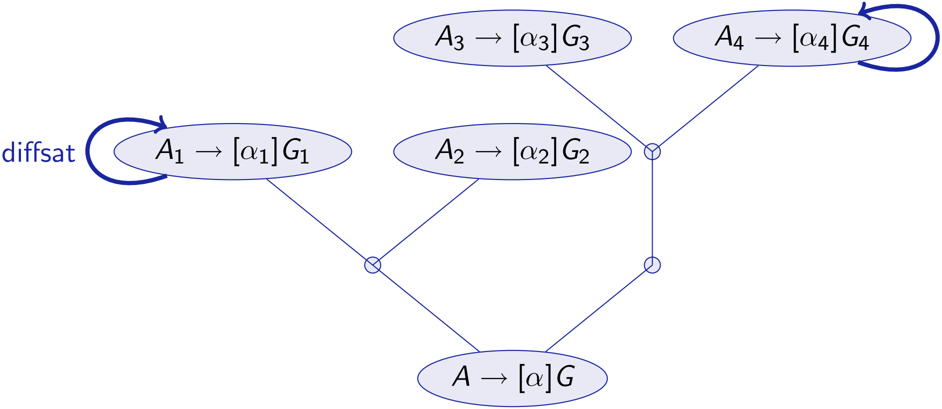
Differential Invariants as Fixedpoints



for $\cup\,,;\,,:=$	do decompose
for $x' = f(x)$	do diffsat

▸ Details

Differential Invariants as Fixedpoints



$A_3 \rightarrow [\alpha_3]G_3$

$A_4 \rightarrow [\alpha_4]G_4$

diffsat

$A_1 \rightarrow [\alpha_1]G_1$

$A_2 \rightarrow [\alpha_2]G_2$

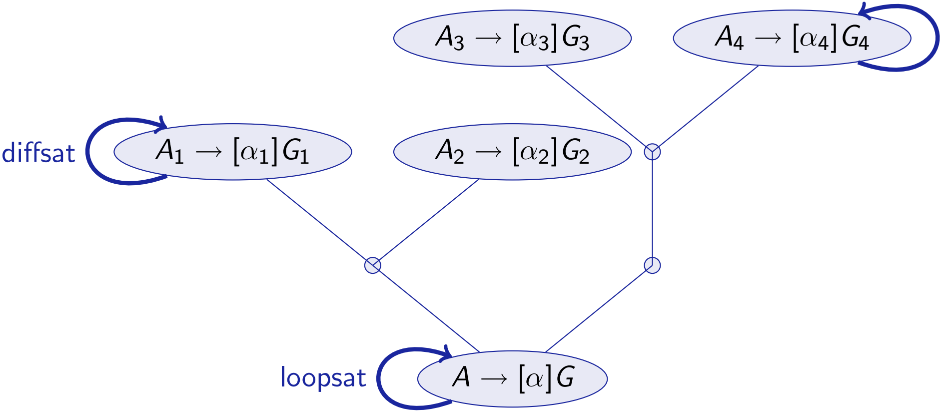
$A \rightarrow [\alpha]G$

for $\cup\, , ; , :=$ do decompose

for $x' = f(x)$ do diffsat

▸ Details

Differential Invariants as Fixedpoints



$A_3 \rightarrow [\alpha_3]G_3$

$A_4 \rightarrow [\alpha_4]G_4$

diffsat

$A_1 \rightarrow [\alpha_1]G_1$

$A_2 \rightarrow [\alpha_2]G_2$

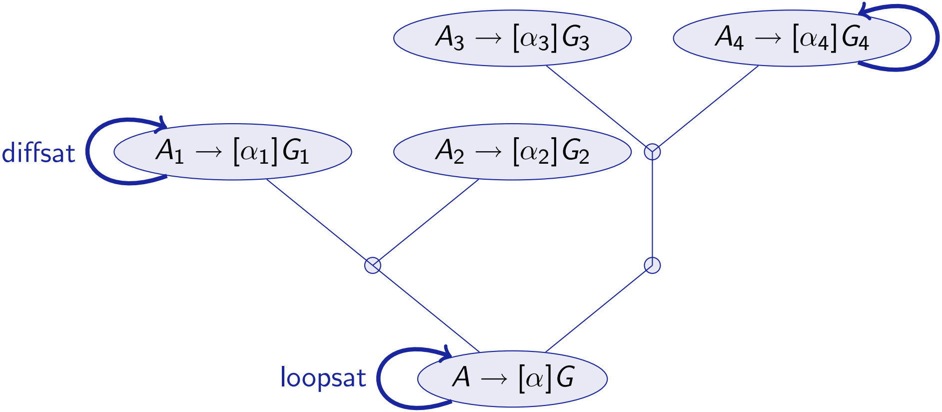
loopsat

$A \rightarrow [\alpha]G$

for $\cup , ; , :=$ do decompose
for $x' = f(x)$ do diffsat
for α^* do loopsat

▸ Details

Differential Invariants as Fixedpoints



$A_3 \rightarrow [\alpha_3]G_3$

$A_4 \rightarrow [\alpha_4]G_4$

diffsat

$A_1 \rightarrow [\alpha_1]G_1$

$A_2 \rightarrow [\alpha_2]G_2$

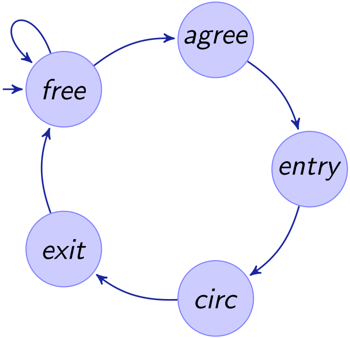
loopsat

$A \rightarrow [\alpha]G$

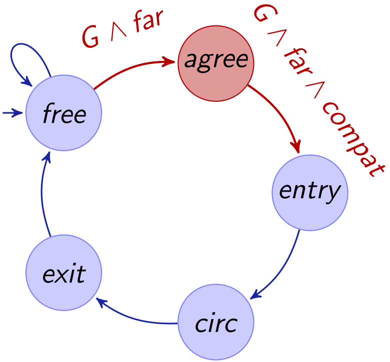
$$\left.\begin{array}{ll} \text{for } \cup\,,;\,,:= & \text{do decompose}\\ \text{for } x' = f(x) & \text{do diffsat}\\ \text{for } \alpha^* & \text{do loopsat}\end{array}\right\}\text{ repeat until fixedpoint}$$

▸ Details

Fixedpoint Iterations for Air Traffic Control



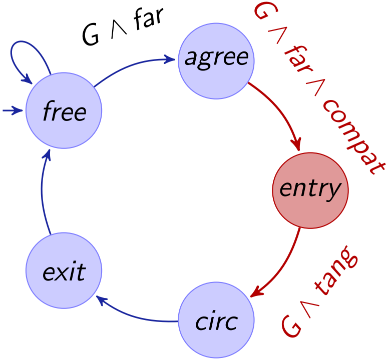
Fixedpoint Iterations for Air Traffic Control



Example (d$\mathcal{L}$ formula of verification subgoal)

$$safe \land far \;\rightarrow\; [agree](safe \land far \land compatible)$$

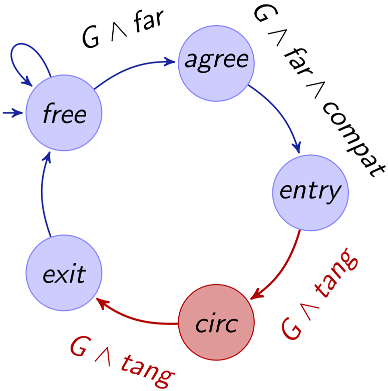
Fixedpoint Iterations for Air Traffic Control



Example (d$\mathcal{L}$ formula of verification subgoal)

$$safe \land far \land compatible \;\rightarrow\; [entry](safe \land tangential)$$

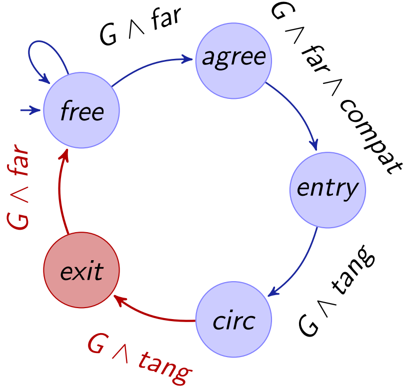
Fixedpoint Iterations for Air Traffic Control



Example (d$\mathcal{L}$ formula of verification subgoal)

$$safe \land tangential \rightarrow [circ](safe \land tangential)$$

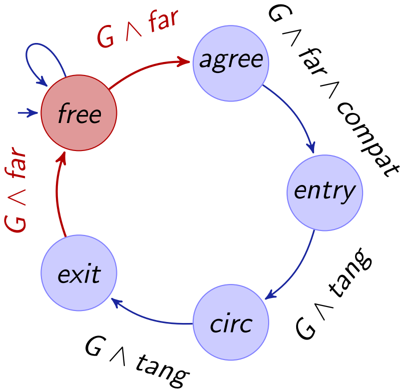
Fixedpoint Iterations for Air Traffic Control



Example (d$\mathcal{L}$ formula of verification subgoal)

$$safe \land tangential \rightarrow [exit](safe \land far)$$

Fixedpoint Iterations for Air Traffic Control



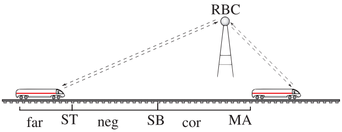
Example (d$\mathcal{L}$ formula of verification subgoal)

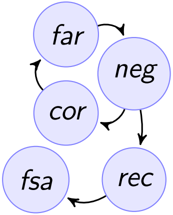
$$safe \land far \rightarrow [free](safe \land far)$$

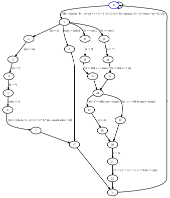
Outline

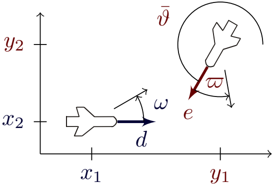
1. Motivation
2. Compositional Verification Logic $d\mathcal{L}$
3. Decompositional Inductive Verification of Hybrid Systems
   - Verification by Symbolic Decomposition
   - Discrete Induction
   - Differential Induction
4. Computing Differential Invariants by Combining Local Fixedpoints
   - Local Fixedpoints & Differential Saturation
   - Global Fixedpoints & Interplay
5. Case Studies & Experimental Results
6. Conclusions & Future Work

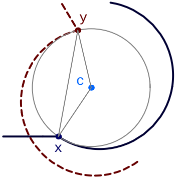
Case Studies

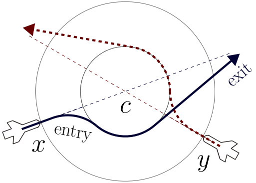


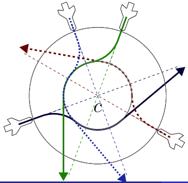












Experimental Results

Case Study	Time(s)	Mem(Mb)	Proof Steps	Dimension
Roundabout (2) ▸	14	8	117	13
Roundabout (3)	387	42	182	18
Roundabout (4) ▸	730	39	234	23
Roundabout (5)	1964	88	317	28
bounded speed entry	20	34	28	12
flyable entry (simplif.)	6	10	98	8
ETCS-kernel	27	28	53	9
ETCS-safety ▸	183	87	169	15
ETCS binary	56	27	147	15
ETCS controllability	1	6	17	5
RBC controllability	1	7	45	16

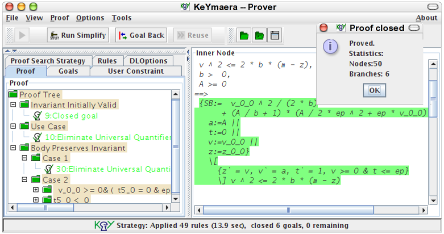
Outline

1. Motivation
2. Compositional Verification Logic d$\mathcal{L}$
3. Decompositional Inductive Verification of Hybrid Systems
    - Verification by Symbolic Decomposition
    - Discrete Induction
    - Differential Induction
4. Computing Differential Invariants by Combining Local Fixedpoints
    - Local Fixedpoints & Differential Saturation
    - Global Fixedpoints & Interplay
5. Case Studies & Experimental Results
6. Conclusions & Future Work

Conclusions

Verifying hybrid systems with challenging dynamics:

- Verification by decomposition: differential dynamic logic d$\mathcal{L}$
- Differential invariants instead of reachability along solutions
- Computing differential invariants as fixedpoints
- Differential saturation procedure
- Exploit locality in system designs
- Verify challenging aircraft control
- Sound “by construction”



```
v ^ 2 <= 2 * b * (m - z),
b >  0,
A >= 0
==>
{SB:=  v_0_0 ^ 2 / (2 * b)
     + (A / b + 1) * (A / 2 * ep ^ 2 + ep * v_0_0)
  a:=A ||
  t:=0 ||
  v:=v_0_0 ||
  z:=z_0_0}
  \[
    {z` = v, v` = a, t` = 1, v >= 0 & t <= ep}
    \] v ^ 2 <= 2 * b * (m - z)
```

Proved.
Statistics:
Nodes:50
Branches: 6

Strategy: Applied 49 rules (13.9 sec), closed 6 goals, 0 remaining

KeYmaera

Future Work

- Compare differential invariants with classical state reachability?
  - Particularly good for hybrid systems with parameterized dynamics
  - Single initial state ⇒ simulation more appropriate
- Case studies
  - Successful for aircraft and train control
  - Performance for other case studies?

Outline

7. Background Material
   - Related Work
   - Formal Semantics
   - Differential Invariants
   - Differential Saturation Procedure
   - Case Studies
   - Hybrid Automata Embedding

	Op	Par	T	Cl	Tec	Aut	Cex	Dim	
HenzingerH94, HyTech	✓	×	✓	×	✓	✓	✓		LHA
LafferrierePY99	✓	×	✓	×	✓		✓		forgetful reset
Fränzle99	✓	×	✓	×	✓		✓	×	robust systems
CKrogh03, CheckMate	✓	×	✓	×	✓	✓	✓		polyhedral
Frehse05, PHAVer	✓	×	✓	×	✓	✓	✓	8	LHA (+affine)
MysorePM05	✓	×	✓	×	✓	●	✓	4	bounded prefix
TomlinPS98,MBT05	○	×	×	×	○	○	●	4	HJB numPDE
RatschanS07, HSolver	✓	×		×	✓	✓	×	4	interval
MannaS98, STeP	✓			×	✓	○	×	7	inv↦VCG, flat
ÁbrahámSH01, PVS	●			×	●	○	×	≈9	HA↪PVS, -“-
ZhouRH92, EDC	×	●	✓	..	×	×	×	×	no maths
DavorenN00, $\mathrm{L}\mu$	×	×		✓	○	×	×	×	prop. H-semantics
RönkköRS03, HGC	✓	×	×	×	×	×	×	×	HGC↪HOL
SSManna04	●	○		×	✓		×	4/1	equational system
CTiwari05	●	○		×	✓		×	6/0	linear, -”-
PrajnaJP07, barrier	●	×		×	●		×	3	needs 10000-dim
$d\mathcal{L}$ & dTL	✓	✓	✓	✓	✓	●	×	28	expr., compos.

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

Definition (Kripke state)

$v : V \to \mathbb{R}$ with set of variables V

Return

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

Definition (Formulas ϕ)

$$\begin{array}{lcll} v \models [\alpha]\phi & :\iff & w \models \phi & \text{for all } w \text{ with } (v,w) \in \rho(\alpha) \\ v \models \langle\alpha\rangle\phi & :\iff & w \models \phi & \text{for some } w \text{ with } (v,w) \in \rho(\alpha) \end{array}$$

Definition (Hybrid programs α)

$$\begin{array}{rcl} \rho(x' = f(x)) & = & \{(\varphi(0), \varphi(r)) \ : \ \varphi \models x' = f(x) \text{ for duration } r\} \\ (v,w) \in \rho(x := \theta) & :\iff & w = v[x \mapsto [\![\theta]\!]_v] \\ \rho(?\chi) & = & \{(v,v) \ : \ v \models \chi\} \\ \rho(\alpha \cup \gamma) & = & \rho(\alpha) \cup \rho(\gamma) \\ \rho(\alpha;\gamma) & = & \rho(\alpha) \circ \rho(\gamma) \\ (v,w) \in \rho(\alpha^*) & :\iff & \text{there is } \ v \xrightarrow{\rho(\alpha)} v_1 \xrightarrow{\rho(\alpha)} v_2 \cdots\cdots\cdots \xrightarrow{\rho(\alpha)} w \end{array}$$

◂ Return

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

Definition (Formulas ϕ)

$$\begin{array}{lcll} v \models [\alpha]\phi & :\Longleftrightarrow & w \models \phi & \text{for all } w \text{ with } (v,w) \in \rho(\alpha) \\ v \models \langle\alpha\rangle\phi & :\Longleftrightarrow & w \models \phi & \text{for some } w \text{ with } (v,w) \in \rho(\alpha) \end{array}$$

Definition (Hybrid programs α)

$$\begin{array}{rcl} \rho(x' = f(x)) & = & \{(\varphi(0), \varphi(r)) \; : \; \varphi \models x' = f(x) \text{ for duration } r\} \\ (v,w) \in \rho(x := \theta) & :\Longleftrightarrow & w = v[x \mapsto [\![\theta]\!]_v] \\ \rho(?\chi) & = & \{(v,v) \; : \; v \models \chi\} \\ \rho(\alpha \cup \gamma) & = & \rho(\alpha) \cup \rho(\gamma) \\ \rho(\alpha;\gamma) & = & \rho(\alpha) \circ \rho(\gamma) \end{array}$$

$$(v,w) \in \rho(\alpha^*) :\Longleftrightarrow \text{there is } v \xrightarrow{\rho(\alpha)} v_1 \xrightarrow{\rho(\alpha)} v_2 \cdots\cdots\cdots \xrightarrow{\rho(\alpha)} w$$

◂ Return

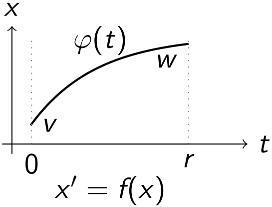
Differential Dynamic Logic d$\mathcal{L}$: Formal Semantics

Definition (Hybrid programs α)

$$\rho(x' = f(x)) \quad = \quad \{(\varphi(0), \varphi(r)) \;:\; \varphi \models x' = f(x) \text{ for duration } r\}$$

$$\text{with } [\![x']\!]_{\varphi(\zeta)} = \frac{\mathrm{d}\varphi(t)(x)}{\mathrm{d}t}(\zeta)$$

- there is $\varphi : [0, r] \to \text{State}$ with $\varphi(0) = v, \varphi(r) = w$
- $[\![x]\!]_{\varphi(\zeta)}$ is continuous in ζ on $[0, r]$
- $\frac{\mathrm{d}\,[\![x]\!]_{\varphi(t)}}{\mathrm{d}t}(\zeta) = [\![f(x)]\!]_{\varphi(\zeta)}$ for $\zeta \in (0, r)$
- $[\![y]\!]_{\varphi(\zeta)} = [\![y]\!]_v$ otherwise



Return

Differential Induction Principle

$$\sigma_1 \quad \mapsto \quad [\![F]\!]_{\sigma_1}$$

◂ Return

Differential Induction Principle

$$\begin{array}{lcl}\sigma_1 & \mapsto & [\![F]\!]_{\sigma_1} \\ \sigma_2 & \mapsto & [\![F]\!]_{\sigma_2}\end{array}$$

◂ Return

Differential Induction Principle

$$\begin{aligned} \sigma_1 &\quad\mapsto\quad [\![F]\!]_{\sigma_1} \\ \sigma_2 &\quad\mapsto\quad [\![F]\!]_{\sigma_2} \end{aligned}$$

In the limit:

$$\frac{\mathrm{d}\,[\![F]\!]_{\sigma}}{\mathrm{d}\sigma}$$

◂ Return

Differential Induction Principle

$$\begin{array}{lcl} \sigma_1 & \mapsto & [\![F]\!]_{\sigma_1} \\ \sigma_2 & \mapsto & [\![F]\!]_{\sigma_2} \end{array}$$

In the limit:

$$\frac{\mathrm{d}\,[\![F]\!]_{\sigma(t)}}{\mathrm{d}t}$$

where $\frac{\mathrm{d}\sigma(t)}{\mathrm{d}t}$ according to ODE

◂ Return

Differential Induction Principle

$$\begin{array}{rcl} \sigma_1 & \mapsto & [\![F]\!]_{\sigma_1} \\ \sigma_2 & \mapsto & [\![F]\!]_{\sigma_2} \end{array}$$

In the limit:

$$\frac{\mathrm{d}\,[\![F]\!]_{\sigma(t)}}{\mathrm{d}t}(\zeta) = [\![\nabla_{x'=f(x)} F]\!]_{\bar{\sigma}(\zeta)}$$

where $\frac{\mathrm{d}\sigma(t)}{\mathrm{d}t}$ according to ODE

◂ Return

Differential Induction Principle

$$\begin{aligned} \sigma_1 &\mapsto [\![F]\!]_{\sigma_1} \\ \sigma_2 &\mapsto [\![F]\!]_{\sigma_2} \end{aligned}$$

In the limit:

$$\frac{\mathrm{d}\,[\![F]\!]_{\sigma(t)}}{\mathrm{d}t}(\zeta) = [\![\nabla_{x'=f(x)} F]\!]_{\bar{\sigma}(\zeta)}$$

where $\frac{\mathrm{d}\sigma(t)}{\mathrm{d}t}$ according to ODE

Lemma (Derivation lemma)

Valuation is a differential homomorphism

◂ Return

Derivations and Differentiation

Definition (Syntactic total derivation $D : \mathrm{Trm}(\Sigma \cup \Sigma') \to \mathrm{Trm}(\Sigma \cup \Sigma')$)

$$\begin{aligned}
D(r) &= 0 && \text{if } r \text{ is a (rigid) number symbol}\\
D(x^{(n)}) &= x^{(n+1)} && \text{if } x \in \Sigma \text{ is non-rigid, } n \geq 0\\
D(a+b) &= D(a) + D(b)\\
D(a \cdot b) &= D(a) \cdot b + a \cdot D(b)\\
D(a/b) &= (D(a) \cdot b - a \cdot D(b))/b^2
\end{aligned}$$

$$\begin{aligned}
D(F) &\equiv \bigwedge_{i=1}^{m} D(F_i) && \{F_1, \dots, F_m\} \text{ all literals of } F\\
D(a \geq b) &\equiv D(a) \geq D(b) && \text{accordingly for } <, >, \leq, =
\end{aligned}$$

◂ Return

Derivations and Differentiation

Lemma (Derivation lemma)

Valuation is a differential homomorphism: for all flows φ *all* $\zeta \in [0, r]$

$$\frac{\mathrm{d}\,[\![\theta]\!]_{\varphi(t)}}{\mathrm{d}t}(\zeta) = [\![D(\theta)]\!]_{\bar{\varphi}(\zeta)}$$

Lemma (Differential substitution principle)

If $\varphi \models x_i' = \theta_i \wedge H$*, then* $\varphi \models \mathcal{D} \leftrightarrow (H \rightarrow \mathcal{D}_{x_i'}^{\theta_i})$ *for all* $\mathcal{D}$*.*

Definition (Differential Invariant)

$$(H \rightarrow F') \;\equiv\; H \rightarrow D(F)_{x_i'}^{\theta_i} \qquad \text{for } [x_i' = \theta_i \wedge H]F$$

◂ Return

Differential Invariants and Variants

Counterexample

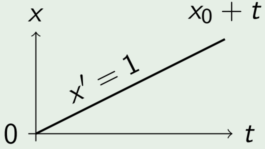
$$\frac{\vdash \forall x\,(x^2 \le 0 \rightarrow 2x \cdot 1 \le 0)}{x^2 \le 0 \ \ \vdash [x' = 1]x^2 \le 0}$$

$$\frac{\vdash \forall x\,(x > 0 \rightarrow -x < 0)}{\vdash \langle x' = -x \rangle x \le 0}$$

Differential Invariants and Variants

Counterexample

$$\frac{\vdash \forall x\,(x^2 \leq 0 \rightarrow 2x \cdot 1 \leq 0)}{x^2 \leq 0 \ \vdash [x' = 1]x^2 \leq 0}$$

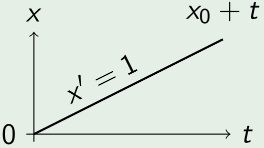


$$\frac{\vdash \forall x\,(x > 0 \rightarrow -x < 0)}{\vdash \langle x' = -x \rangle x \leq 0}$$

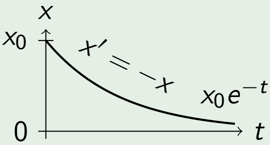
Differential Invariants and Variants

Counterexample

$$\dfrac{\vdash \forall x\,(x^2 \le 0 \rightarrow 2x \cdot 1 \le 0)}{x^2 \le 0 \;\;\vdash [x' = 1]x^2 \le 0}$$



$$\dfrac{\vdash \forall x\,(x > 0 \rightarrow -x < 0)}{\vdash \langle x' = -x\rangle x \le 0}$$



Differential Saturation Procedure

refine d$\mathcal{L}$ verification calculus to automatic verification fixedpoint algorithm

⇣

```
function prove($A \vdash [\mathcal{D} \wedge H]G$):
2: if prove($\forall_{cl}(H \rightarrow G)$) then
     return true /* property proven */
   for each $F \in$ Candidates($A \vdash [\mathcal{D} \wedge H]G$, $H$) do
     if prove($A \wedge H \vdash F$) and prove($\forall_{cl}(H \rightarrow \nabla_{\mathcal{D}} F)$) then
       $H$ := $H \wedge F$ /* refine by differential invariant*/
       goto 2;     /* repeat fixedpoint loop */
   end for
   return "not provable using candidates"
```

◂ Return

Full European Train Control System (ETCS)

provable automatically!

$$\begin{array}{lll}
\text{spec} & : & \tau.v^2 - \mathfrak{m}.d^2 \leq 2b(\mathfrak{m}.e - \tau.p) \land \tau.v \geq 0 \land \mathfrak{m}.d \geq 0 \land b > 0 \\
 & & \rightarrow\ [\text{ETCS}](\tau.p \geq \mathfrak{m}.e \rightarrow \tau.v \leq \mathfrak{m}.d) \\
\text{ETCS} & : & (\text{train} \cup \text{rbc})^* \\
\text{train} & : & \text{spd};\ \text{atp};\ \text{move} \\
\text{spd} & : & \ (?\tau.v \leq \mathfrak{m}.r;\ \tau.a := *;\ ? -b \leq \tau.a \leq A) \\
 & & \cup(?\tau.v \geq \mathfrak{m}.r;\ \tau.a := *;\ ?0 > \tau.a \geq -b) \\
\text{atp} & : & SB := \frac{\tau.v^2 - \mathfrak{m}.d^2}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon\ \tau.v\right); \\
 & & \ (?(\mathfrak{m}.e - \tau.p \leq SB \lor rbc.message = emergency);\ \tau.a := -b) \\
 & & \cup(?\mathfrak{m}.e - \tau.p \geq SB \land rbc.message \neq emergency) \\
\text{move} & : & t := 0;\ (\tau.p' = \tau.v, \tau.v' = \tau.a, t' = 1 \land \tau.v \geq 0 \land t \leq \varepsilon) \\
\text{rbc} & : & \ (rbc.message := emergency) \\
 & & \cup\ \big(\mathfrak{m}_0 := \mathfrak{m}; \mathfrak{m} := *; \\
 & & \quad ?\mathfrak{m}.r \geq 0 \land \mathfrak{m}.d \geq 0 \land \mathfrak{m}_0.d^2 - \mathfrak{m}.d^2 \leq 2b(\mathfrak{m}.e - \mathfrak{m}_0.e)\big)
\end{array}$$

Full European Train Control System (ETCS)

```
 state = 0,
 2 * b * (m - z) >= v ^ 2 - d ^ 2,
 v >= 0, d >= 0, v >= 0, ep >  0, b >  0, amax >  0, d >= 0
==>
    v <= vdes
 -> \forall R a_3;
      (   a_3 >= 0 & a_3 <= amax
       ->   (      m - z
                <= (amax / b + 1) * ep * v
                 + (v ^ 2 - d ^ 2) / (2 * b)
                 + (amax / b + 1) * amax * ep ^ 2 / 2
             -> \forall R t0;
                  (   t0 >= 0
                   -> \forall R ts0;  (0 <= ts0 & ts0 <= t0 -> -b * ts0 + v >= 0 & ts0 + 0 <= ep)
                   ->      2 * b * (m - 1 / 2 * (-b * t0 ^ 2 + 2 * t0 * v + 2 * z))
                        >= (-b * t0 + v) ^ 2
                         - d ^ 2
                      & -b * t0 + v >= 0
                      & d >= 0))
          & (      m - z
                >  (amax / b + 1) * ep * v
                 + (v ^ 2 - d ^ 2) / (2 * b)
                 + (amax / b + 1) * amax * ep ^ 2 / 2
             -> \forall R t2;
                  (   t2 >= 0
                   -> \forall R ts2;  (0 <= ts2 & ts2 <= t2 -> a_3 * ts2 + v >= 0 & ts2 + 0 <= ep)
                   ->      2 * b * (m - 1 / 2 * (a_3 * t2 ^ 2 + 2 * t2 * v + 2 * z))
                        >= (a_3 * t2 + v) ^ 2
                         - d ^ 2
                      & a_3 * t2 + v >= 0
                      & d >= 0)))
```

Tangential Roundabout Collision Avoidance Maneuver

provable automatically!

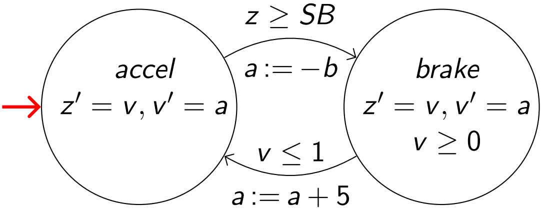
$$\begin{aligned}
\psi &\equiv \phi \to [ATC^*]\phi \\
\phi &\equiv \|x-y\|^2 \geq p^2 \equiv (x_1-y_1)^2 + (x_2-y_2)^2 \geq p^2 \\
ATC &\equiv \mathit{free};\quad \mathit{agree};\quad \mathcal{F}(\omega) \land \mathcal{G}(\omega) \\
\mathit{free} &\equiv \exists\omega\,\mathcal{F}(\omega) \land \exists\varrho\,\mathcal{G}(\varrho) \land \phi \\
\mathit{agree} &\equiv \exists u\,\omega := u;\ \exists c\,(d := \omega(x-c)^\perp \land e := \omega(y-c)^\perp)
\end{aligned}$$

$$\mathcal{F}(\omega) \equiv \left(\begin{array}{rlll}
& x_1' = v\cos\vartheta & = d_1 & \\
\land & x_2' = v\sin\vartheta & = d_2 & \\
\land & d_1' = v(-\sin\vartheta)\vartheta' & = -\omega d_2 & \\
\land & d_2' = v(\cos\vartheta)\vartheta' & = \quad \omega d_1 &
\end{array}\right) \quad \mathcal{G}(\varrho) \equiv \left(\begin{array}{rl}
& y_1' = e_1 \\
\land & y_2' = e_2 \\
\land & e_1' = -\varrho e_2 \\
\land & e_2' = \quad \varrho e_1
\end{array}\right)$$

provable automatically!

$$
\begin{aligned}
\psi \quad &\equiv \phi \to [ATC^*]\phi \\
\phi \quad &\equiv (x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2 \wedge (y_1 - z_1)^2 + (y_2 - z_2)^2 \geq p^2 \\
&\quad \wedge (x_1 - z_1)^2 + (x_2 - z_2)^2 \geq p^2 \wedge (x_1 - u_1)^2 + (x_2 - u_2)^2 \geq p^2 \\
&\quad \wedge (y_1 - u_1)^2 + (y_2 - u_2)^2 \geq p^2 \wedge (z_1 - u_1)^2 + (z_2 - u_2)^2 \geq p^2 \\
ATC \quad &\equiv free; \quad agree; \\
&\quad x_1' = d_1 \wedge x_2' = d_2 \wedge d_1' = -\omega_x d_2 \wedge d_2' = \omega_x d_1 \\
&\quad \wedge y_1' = e_1 \wedge y_2' = e_2 \wedge e_1' = -\omega_y e_2 \wedge e_2' = \omega_y e_1 \\
&\quad \wedge z_1' = f_1 \wedge z_2' = f_2 \wedge f_1' = -\omega_z f_2 \wedge f_2' = \omega_z f_1 \\
&\quad \wedge u_1' = g_1 \wedge u_2' = g_2 \wedge g_1' = -\omega_u g_2 \wedge g_2' = \omega_u g_1 \\
free \quad &\equiv (\omega_x := *; \quad \omega_y := *; \quad \omega_z := *; \quad \omega_u := *; \\
&\quad x_1' = d_1 \wedge x_2' = d_2 \wedge d_1' = -\omega_x d_2 \wedge d_2' = \omega_x d_1 \\
&\quad \wedge y_1' = e_1 \wedge y_2' = e_2 \wedge e_1' = -\omega_y e_2 \wedge e_2' = \omega_y e_1 \\
&\quad \wedge z_1' = f_1 \wedge z_2' = f_2 \wedge f_1' = -\omega_z f_2 \wedge f_2' = \omega_z f_1 \\
&\quad \wedge u_1' = g_1 \wedge u_2' = g_2 \wedge g_1' = -\omega_u g_2 \wedge g_2' = \omega_u g_1 \wedge \phi)^* \\
agree \quad &\equiv \omega := *; \; c := *; \\
&\quad d_1 := -\omega(x_2 - c_2); \; d_2 := \omega(x_1 - c_1); \\
&\quad e_1 := -\omega(y_1 - c_1); \; e_2 := \omega(y_2 - c_2); \\
&\quad f_1 := -\omega(z_1 - c_1); \; f_2 := \omega(z_2 - c_2); \\
&\quad g_1 := -\omega(u_1 - c_1); \; g_2 := \omega(u_2 - c_2)
\end{aligned}
$$

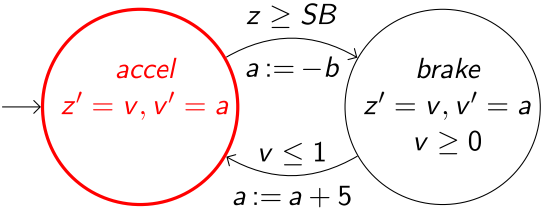
Embedding Hybrid Automata as Hybrid Programs



$$\rightsquigarrow$$

$$
\begin{aligned}
&{\color{red} q := accel};\\
&\big(\quad (?q = accel;\quad z' = v \land v' = a)\\
&\cup\ (?q = accel \land z \geq SB;\quad a := -b;\quad q := brake;\quad ?v \geq 0)\\
&\cup\ (?q = brake;\quad z' = v \land v' = a \land v \geq 0)\\
&\cup\ (?q = brake \land v \leq 1;\quad a := a + 5;\quad q := accel)\big)^*
\end{aligned}
$$

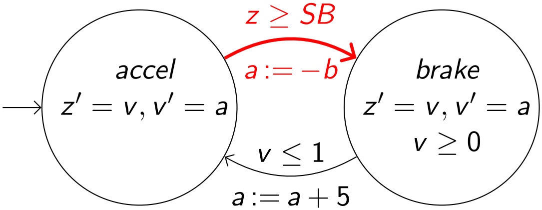
Embedding Hybrid Automata as Hybrid Programs



$$\rightsquigarrow$$

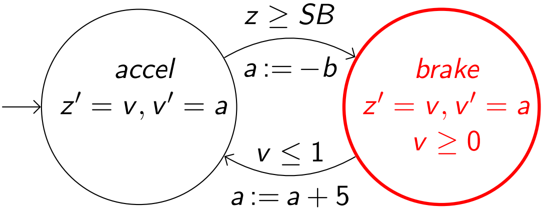
$$\begin{aligned}
&q := accel;\\
&\big(\quad (?q = accel;\quad z' = v \land v' = a)\\
&\cup\ (?q = accel \land z \geq SB;\quad a := -b;\quad q := brake;\quad ?v \geq 0)\\
&\cup\ (?q = brake;\quad z' = v \land v' = a \land v \geq 0)\\
&\cup\ (?q = brake \land v \leq 1;\quad a := a + 5;\quad q := accel)\big)^*
\end{aligned}$$

Embedding Hybrid Automata as Hybrid Programs



$$
\begin{aligned}
&q := accel;\\
&\big(\quad (?q = accel;\quad z' = v \land v' = a)\\
&\cup\ (?q = accel \land z \geq SB;\quad a := -b;\quad q := brake;\quad ?v \geq 0)\\
&\cup\ (?q = brake;\quad z' = v \land v' = a \land v \geq 0)\\
&\cup\ (?q = brake \land v \leq 1;\quad a := a + 5;\quad q := accel)\big)^*
\end{aligned}
$$

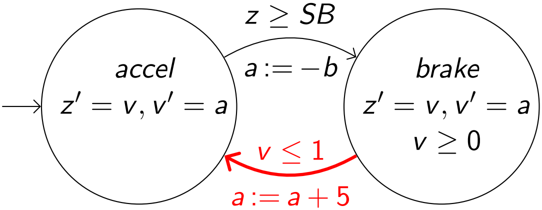
Embedding Hybrid Automata as Hybrid Programs



$\rightsquigarrow$

$$
\begin{aligned}
& q := accel;\\
& \big(\ \ (?q = accel;\quad z' = v \land v' = a)\\
& \cup\ (?q = accel \land z \geq SB;\quad a := -b;\quad q := brake;\quad ?v \geq 0)\\
& \cup\ (?q = brake;\quad z' = v \land v' = a \land v \geq 0)\\
& \cup\ (?q = brake \land v \leq 1;\quad a := a + 5;\quad q := accel)\big)^*
\end{aligned}
$$

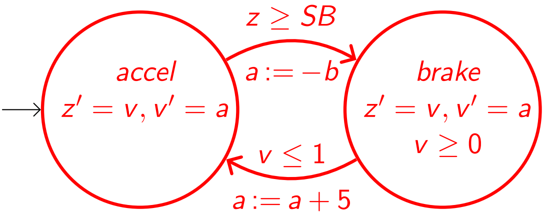
Embedding Hybrid Automata as Hybrid Programs



$\rightsquigarrow$

$$
\begin{aligned}
&q := accel;\\
&(\quad (?q = accel;\quad z' = v \land v' = a)\\
&\cup\ (?q = accel \land z \geq SB;\quad a := -b;\quad q := brake;\quad ?v \geq 0)\\
&\cup\ (?q = brake;\quad z' = v \land v' = a \land v \geq 0)\\
&\cup\ (?q = brake \land v \leq 1;\quad a := a + 5;\quad q := accel))^*
\end{aligned}
$$

Embedding Hybrid Automata as Hybrid Programs



$\rightsquigarrow$

$$
\begin{array}{ll}
q := accel; & \\
\big(& (?q = accel; \quad z' = v \land v' = a) \\
\cup & (?q = accel \land z \geq SB; \quad a := -b; \quad q := brake; \quad ?v \geq 0) \\
\cup & (?q = brake; \quad z' = v \land v' = a \land v \geq 0) \\
\cup & (?q = brake \land v \leq 1; \quad a := a + 5; \quad q := accel)\big)^{*}
\end{array}
$$