Bellerophon: Tactical Theorem Proving for Hybrid Systems

Nathan Fulton, Stefan Mitsch, Brandon Bohrer, André Platzer
Carnegie Mellon University

Cyber-Physical Systems

Cyber-Physical Systems combine computation and control.

Hybrid Systems model combinations of discrete and continuous dynamics.

Bellerophon

Verifying hybrid systems is hard.

Bellerophon

Verifying hybrid systems is hard.
Bellerophon demonstrates how to tackle hybrid systems with tactics:

Bellerophon

Verifying hybrid systems is hard.
Bellerophon demonstrates how to tackle hybrid systems with tactics:

- Build on a sound core.

Bellerophon

Verifying hybrid systems is hard.
Bellerophon demonstrates how to tackle hybrid systems with tactics:

- Build on a sound core.
- Implement high-level primitives for hybrid systems proofs.

Bellerophon

Verifying hybrid systems is hard.
Bellerophon demonstrates how to tackle hybrid systems with tactics:

- Build on a sound core.
- Implement high-level primitives for hybrid systems proofs.
- Automate common constructions (for ODEs and control software)

Bellerophon

Theorem	Bellerophon LOC	Conceptual Proof Steps	Hybrid Systems Axiom Applications
Static Safety	12	71	30,355
Passive-Friendly Safety	45	140	68,620
Orientation Safety	15	108	173,989
Pass Intersection Liveness	234	440	61,878

KeYmaera X: Trustworthy Foundations

Interactive Reachability Analysis

- ➢ Bellerophon combinator language
- ➢ Bellerophon standard library for hybrid systems
- ➢ Demonstration

Bellerophon for Automation and Tooling

Conclusions & Resources

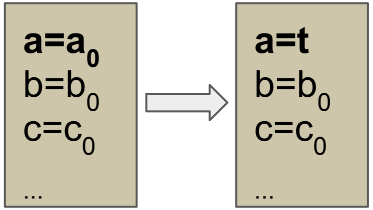
Trustworthy Foundations

KeYmaera X enables trustworthy automation for hybrid systems analysis:

- A well-defined **logical foundations,**
- implemented in a **small trustworthy core**
- that ensures correctness of **automation** and **tooling**.

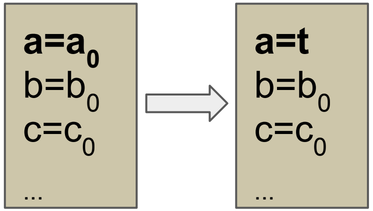
Hybrid Programs

$a := t$

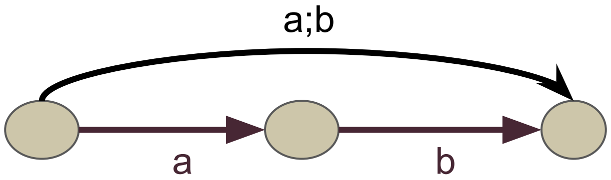


Hybrid Programs

a := t

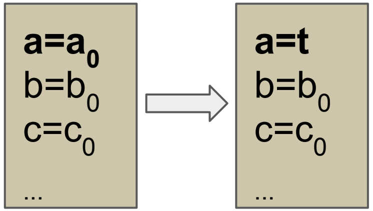


a;b

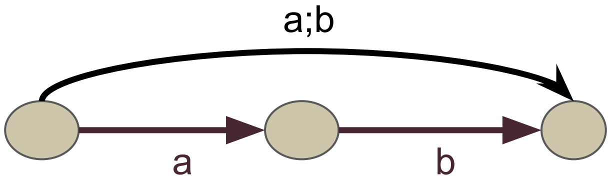


Hybrid Programs

a := t



a;b



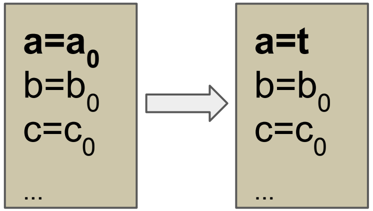
?P

If P is true: no change

If P is false: terminate

Hybrid Programs

a := t

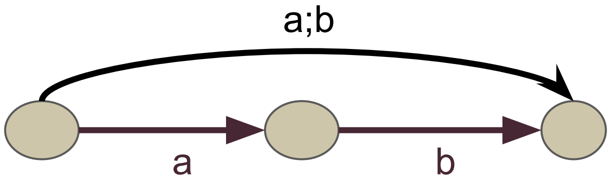


?P

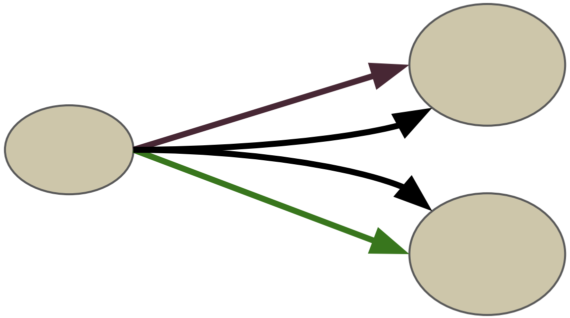
If P is true: no change

If P is false: terminate

a;b

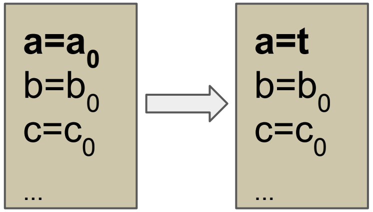


a U b



Hybrid Programs

a := t

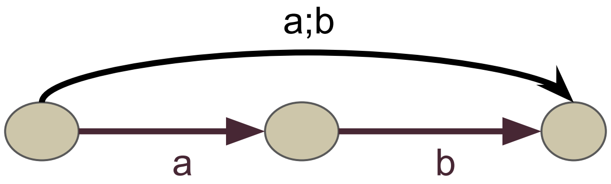


?P

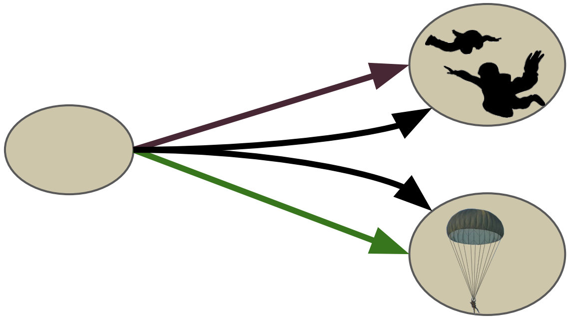
If P is true: no change

If P is false: terminate

a;b

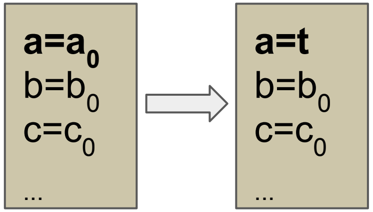


a U b



Hybrid Programs

a := t

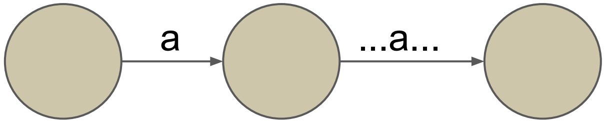


?P

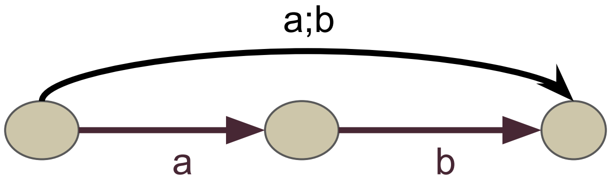
If P is true: no change

If P is false: terminate

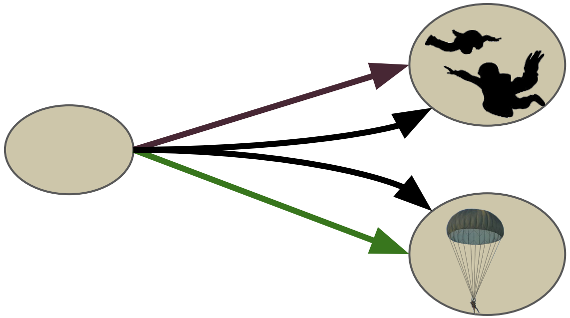
a*



a;b

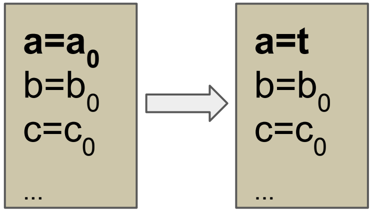


a ∪ b

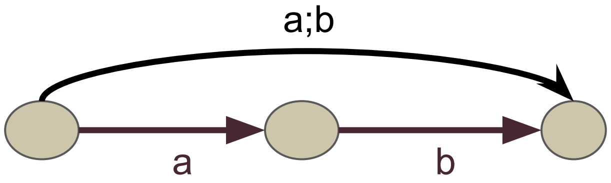


Hybrid Programs

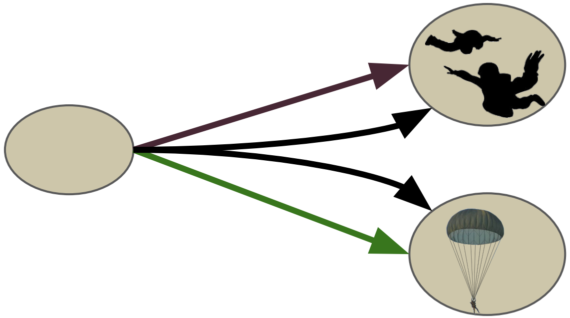
a := t



a;b



a U b

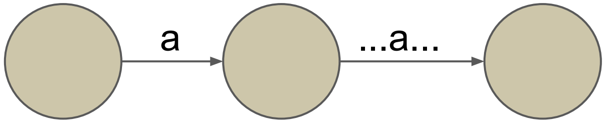


?P

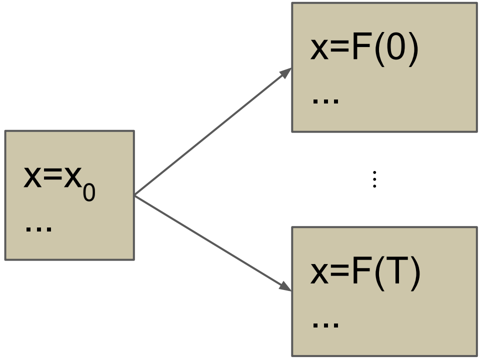
If P is true: no change

If P is false: terminate

a*



x'=f



Reachability Specifications

[a]P	“after every execution of a, P”
<a>P	“after some execution of a, P”

Reachability Specifications

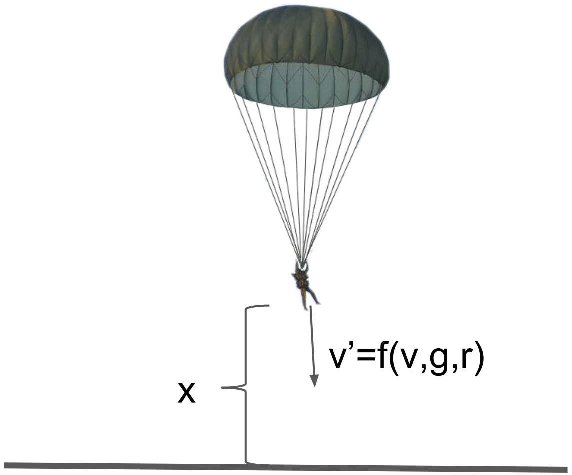
$[a]P$ “after every execution of a, P”

$\langle a \rangle P$ “after some execution of a, P”

$$init \rightarrow [\{x := u(x);\ x' = f(x)\}^*]safe$$

Hello, World

```
{
  {?Dive ∪ r := r_p};
  t:=0;
  {x' = v,
   V' = f(v,g,r), t'=1
   & 0≤x & t≤T}
}*
```

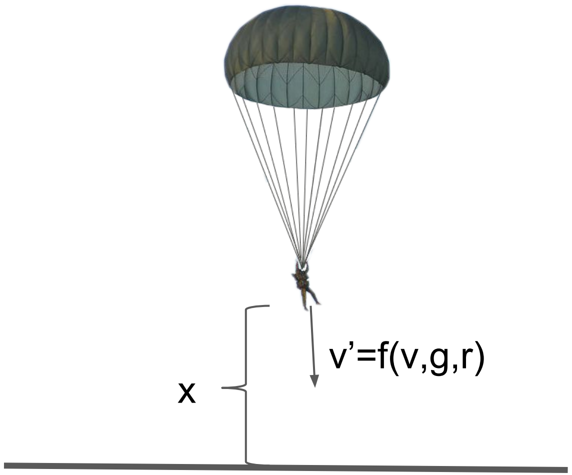


Control: Continue diving if safe, else open parachute.

Plant: Downward velocity determined by gravity, air resistance.

Hello, World

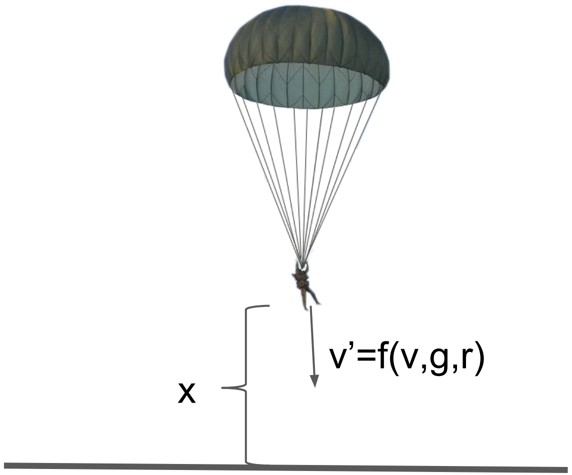
```
{
  {?Dive ∪ r := r_p};
  t:=0;
  {x' = v,
   V' = f(v,g,r), t'=1
   & 0≤x & t≤T}
}*
```



Control: Continue diving if safe, else open parachute.
Plant: Downward velocity determined by gravity, air resistance.

Hello, World

```
{
  {?Dive U r := r_p};
  t:=0;
  {x' = v,
   V' = f(v,g,r), t'=1
   & 0≤x & t≤T}
}*
```

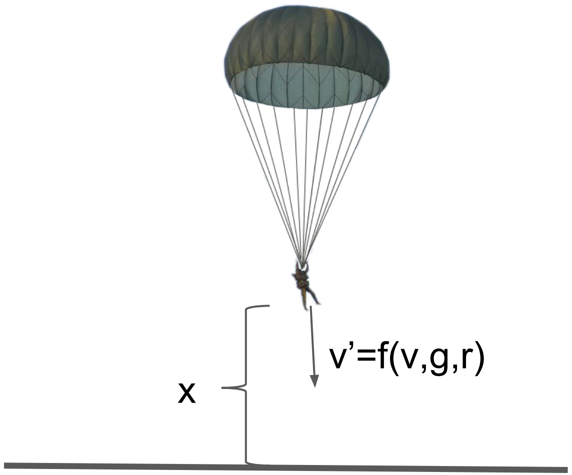


Control: Continue diving if safe, else open parachute.

Plant: Downward velocity determined by gravity, air resistance.

Hello, World

```
{
   {?Dive ∪ r := r_p};
   t:=0;
   {x′ = v,
    V′ = f(v,g,r), t′=1
    & 0≤x & t≤T}
}*
```

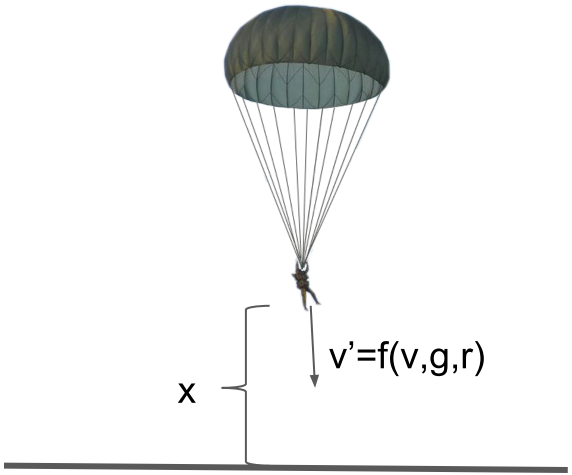


Control: Continue diving if safe, else open parachute.

Plant: Downward velocity determined by gravity, air resistance.

Hello, World

```
{
   {?Dive ∪ r := r_p};
   t:=0;
   {x' = v,
    v' = f(v,g,r), t'=1
     & 0≤x & t≤T}
}*
```

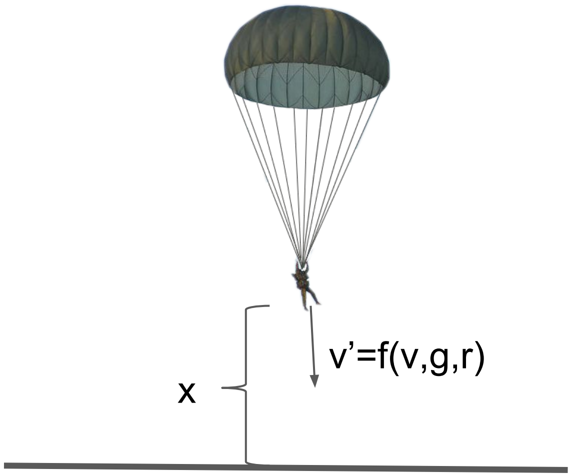


Control: Continue diving if safe, else open parachute.

Plant: Downward velocity determined by gravity, air resistance.

Hello, World

```
{
   {?Dive ∪ r := r_p};
   t:=0;
   {x' = v,
    V' = f(v,g,r), t'=1
    & 0≤x & t≤T}
}*
```

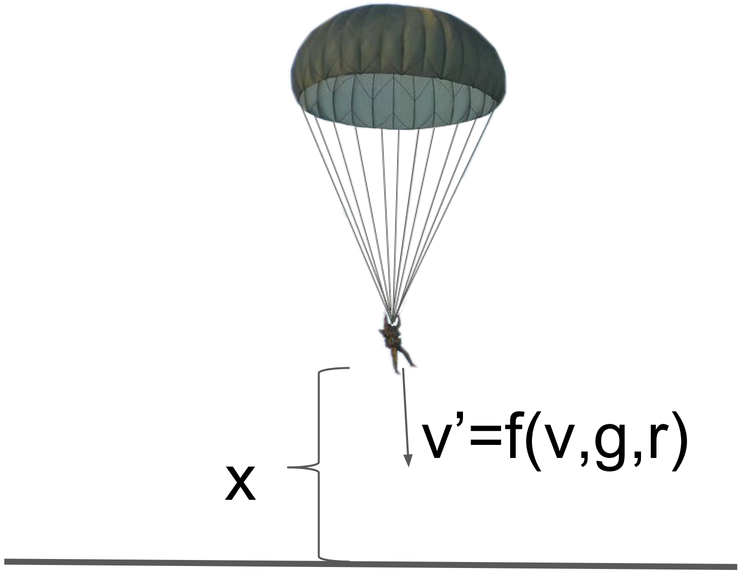


Control: Continue diving if safe, else open parachute.

Plant: Downward velocity determined by gravity, air resistance.

Reachability Specifications

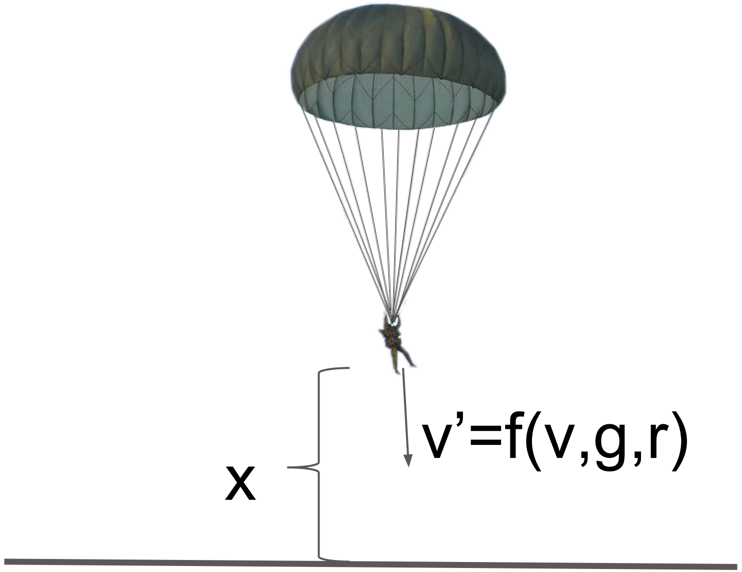
```
(Dive & g>0 & ...)→
[{
   {?Dive ∪ r := r_p};
   {x' = v,
    V' = f(v,g,r)
    & 0≤x}
}*] (x=0→m≤v)
```



Reachability Specifications

```
(Dive & g>0 & …)→
[{
  {?Dive ∪ r := r_p};
  {x' = v,
   V' = f(v,g,r)
   & 0≤x}
}*] (x=0→m≤v)
```

If the parachuter is on the ground, their speed is safe ($m \le v \le 0$)

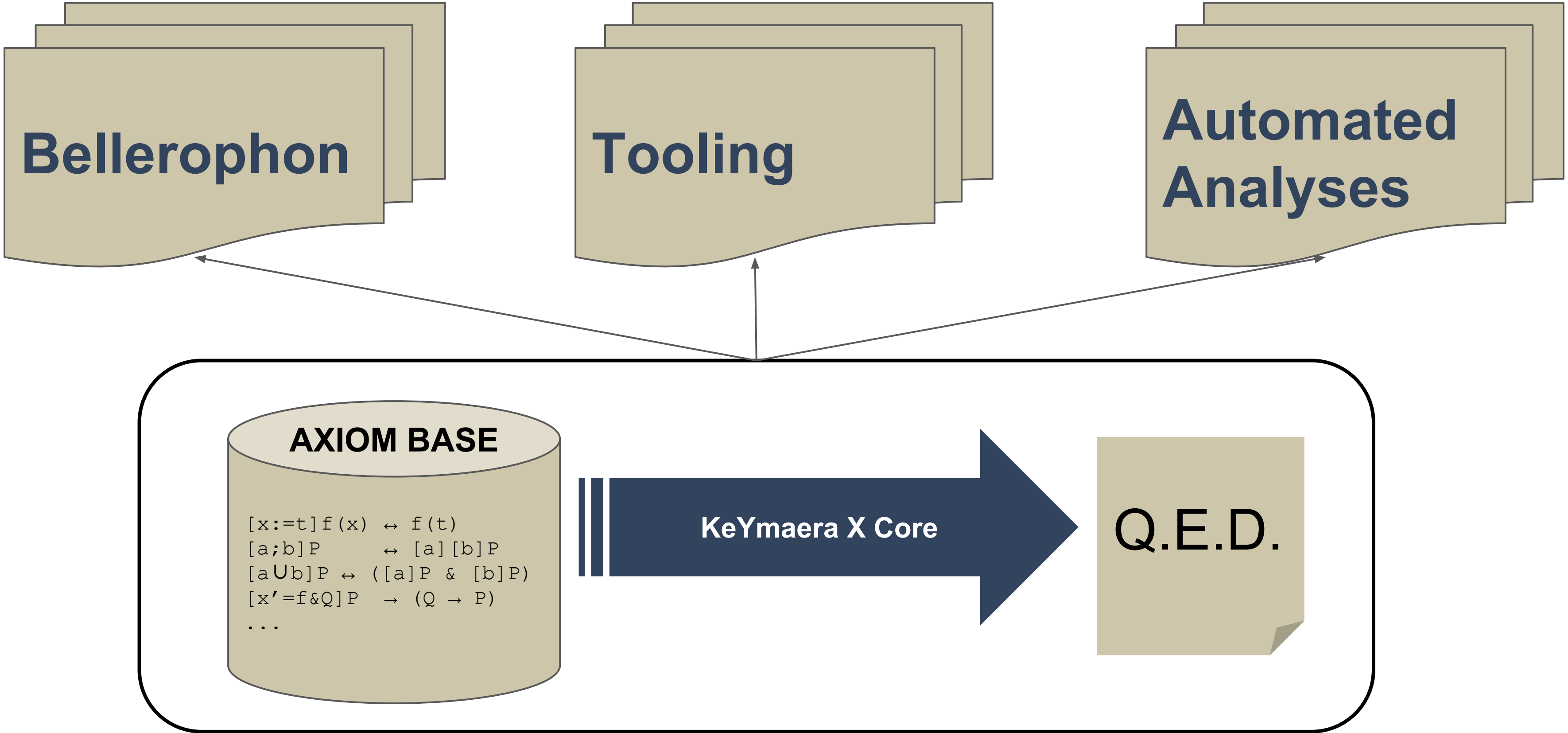


Dynamical Axioms

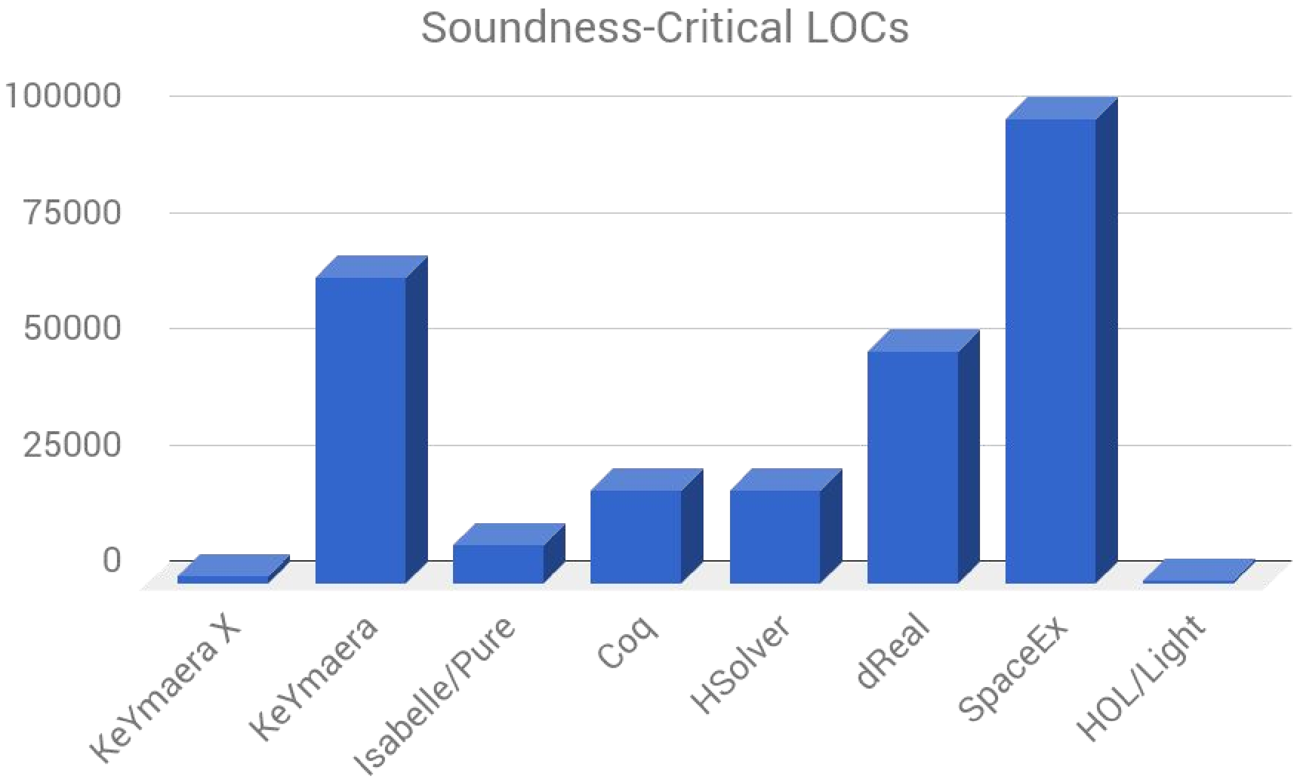
```
[x:=t]f(x)  ↔  f(t)
[a;b]P      ↔  [a][b]P
[a∪b]P      ↔  ([a]P & [b]P)
[x'=f&Q]P   →  (Q → P)
...
```

Trusted Core

Trustworthy Implementations



Prover Core Comparison



Bellerophon

Bellerophon enables interactive verification and tool development:

Bellerophon

Bellerophon enables interactive verification and tool development:

- A **standard library** of common proof techniques.

Bellerophon

Bellerophon enables interactive verification and tool development:

- A **standard library** of common proof techniques.
- A **combinator language/library** for **decomposing** theorems and **composing** proof strategies.

Bellerophon

Standard Library

Tactic	Meaning
`prop`	Applies propositional reasoning exhaustively.
`unfold`	Symbolically executes discrete, loop-free programs.
`loop(J, i)`	Applies loop invariance axiom to position i.
`dI,dG,dC,dW`	Reasoning principles for differential equations.

Bellerophon

Standard Library

Tactic	Meaning
`prop`	Applies propositional reasoning exhaustively.
`unfold`	Symbolically executes discrete, loop-free programs.
`loop(J, i)`	Applies loop invariance axiom to position i.
`dI,dG,dC,dW`	Reasoning principles for differential equations.

1000+

Bellerophon

Combinators

1000+

Tactic	Meaning
`prop`	Applies propositional reasoning exhaustively.
`unfold`	Symbolically executes discrete, loop-free programs.
`loop(J, i)`	Applies loop invariance axiom to position i, extends J with constants.
`dI,dG,dC,dW`	Reasoning principles for differential equations.

Combinator	Meaning
A ; B	Execute A on current goal, then execute B on the result.
A | B	Try executing A on current goal. If A fails, execute B on current goal.
A*	Run A until it no longer applies.
$A<(B_1,B_2, \ldots ,B_N)$	Execute A on current goal to create N subgoals. Run B_i on subgoal i.

Isolating Interesting Questions

(Dive & g>0 & …)→

[{

}*] **(x=0→m≤v)**

Isolating Interesting Questions

(Dive & g>0 & …) → [{

}*] (x=0→m≤v)

prop ; loop(J,1)

(Dive & g>0 & …) → J

Loop invariant holds initially

J → [

] J

Loop invariant is preserved

J → x=0→m≤v

Loop invariant implies safety

Isolating Interesting Questions

(Dive & g>0
& …)→
[{

}*] **(x=0→m≤v)**

prop ; loop(J,1)

(Dive &
g>0 &
…)→
J

Loop invariant holds initially

J→[

]J

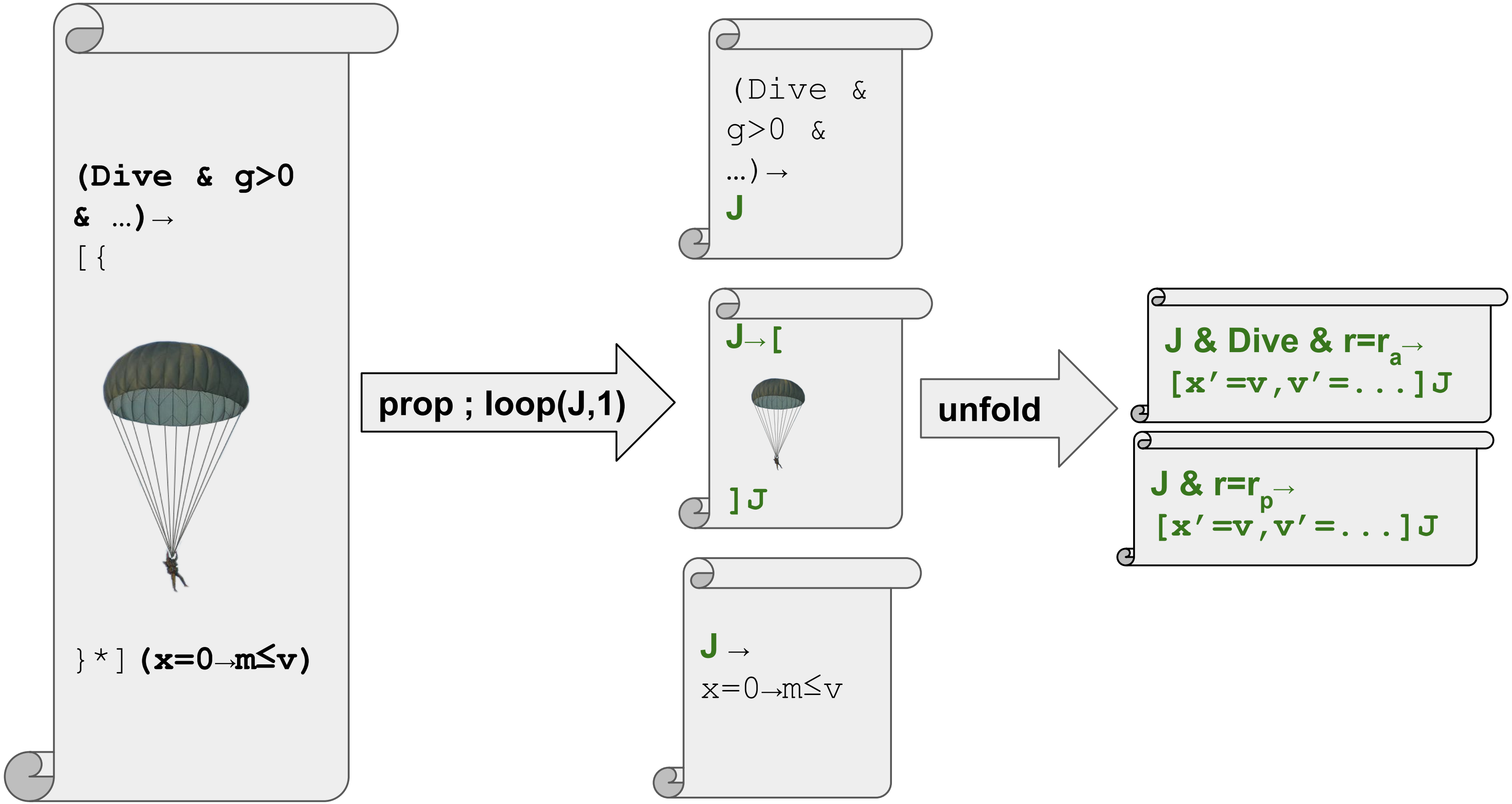
Loop invariant is preserved

J →
x=0→m≤v

Loop invariant implies safety

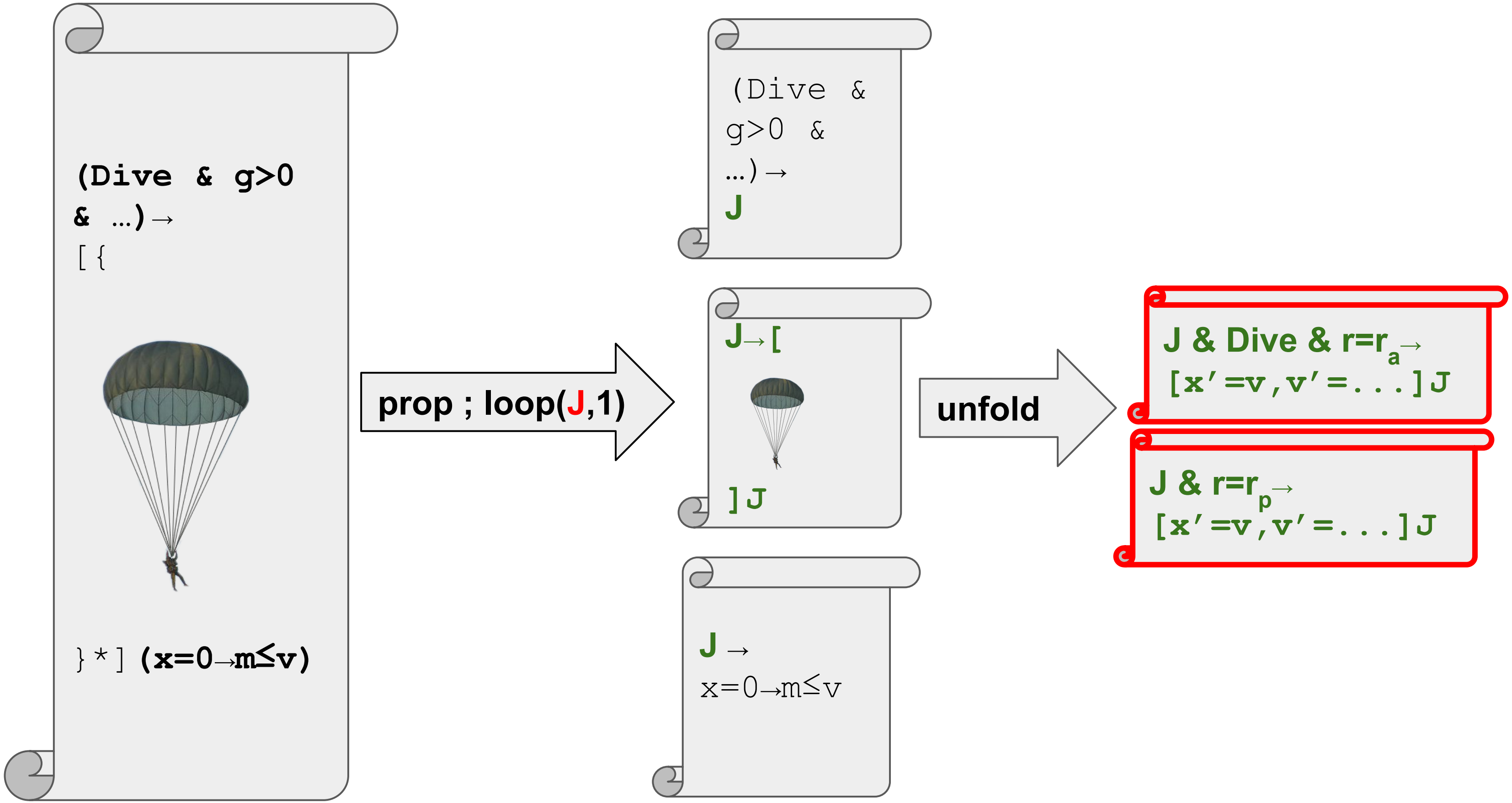
Bellerophon

Isolating Interesting Questions



Bellerophon

Isolating Interesting Questions



Isolating Interesting Questions

```
prop ;  loop(J, 1) <(
   QE, /* Real arith. solver */
   QE,
   unfold ; <(
      … /* parachute open case */
      … /* parachute closed case */
   )
)
```

Trustworthy Standard Library at High Abstraction Level

```
J → [{ctrl; plant}*]J
```

J = v > -sqrt(g/pr) > m & …

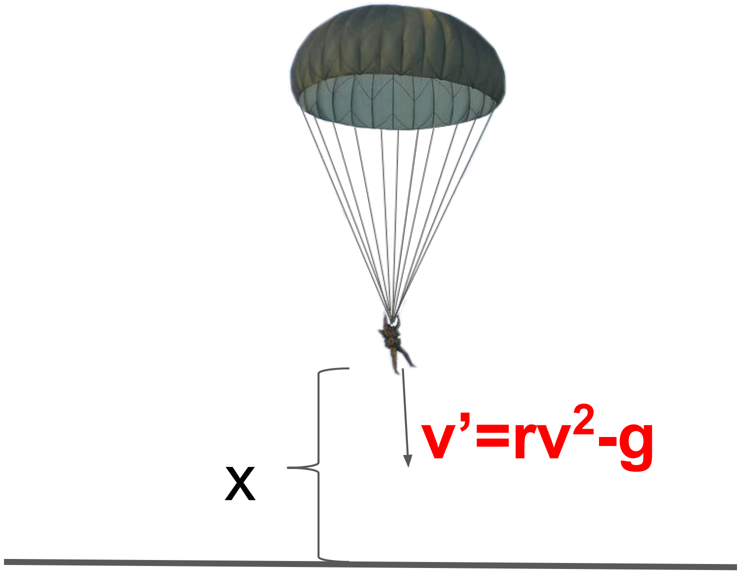
Parachute Open Case:

$v \geq v_0 - gt$

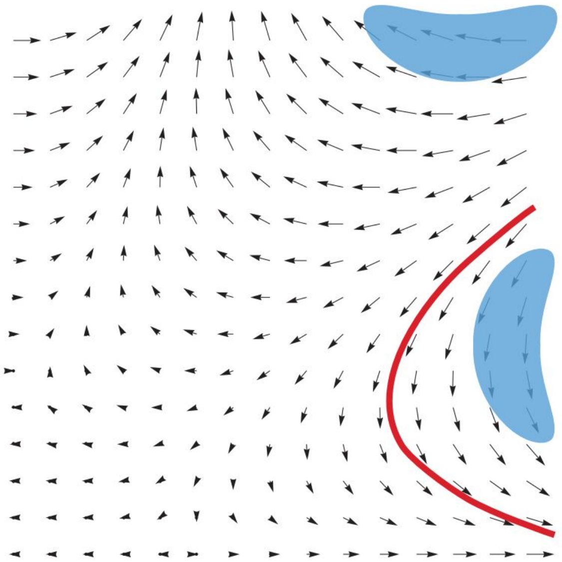
$\geq v_0 - gT$

$> -\mathrm{sqrt}(g/pr)$

Inductive invariants



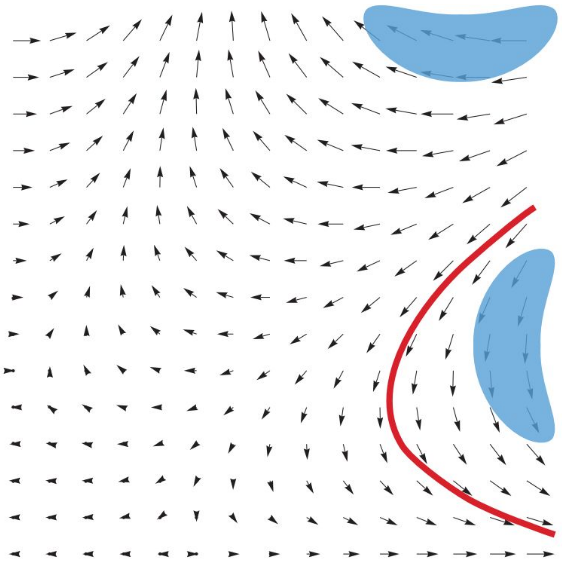
From Axioms to Proof Steps



DI Axiom:

[{x'=f&Q}]P↔([?Q]P←(Q→[{x'=f&Q}]P'))

From Axioms to Proof Steps



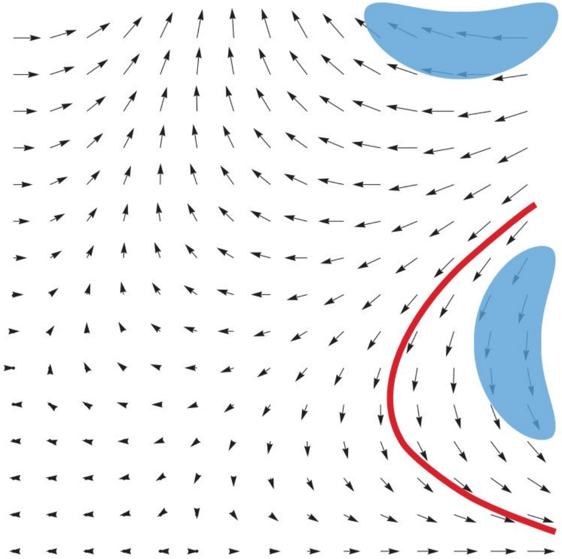
DI Axiom:

$[\{x'=f\&Q\}]P \leftrightarrow ([?Q]P \leftarrow (Q \rightarrow [\{x'=f\&Q\}]P'))$

Example:

$[v'=r_p v^2-g, t'=1] v \geq v_0 - gt$

From Axioms to Proof Steps



DI Axiom:

[{x'=f&Q}]P↔([?Q]P←(Q→[{x'=f&Q}]P'))

Example:

$[v'=r_p v^2-g, t'=1]v \geq v_0 - gt$ ↔

… ↔

$[v':=r_p v^2-g][t':=1]v' \geq -g*t'$ ↔

$r_p v^2-g \geq -g$ ↔

$r_p \geq 0$

From Axioms to Proof Steps

dI Tactic:

Side derivation:

$(v \geq v_0 - gt)' \leftrightarrow$

$(v)' \geq (v_0 - gt)' \leftrightarrow$

$(v)' \geq (v_0 - gt)' \leftrightarrow$

$(v)' \geq (v_0)' - (gt)' \leftrightarrow$

$(v)' \geq (v_0)' - (t(g)' + g(t')) \leftrightarrow$

$v' \geq v_0' - (tg' + gt')$

$H = r_p \geq 0 \ \& \ r_a \geq 0 \ \& \ g > 0 \ \& \ \ldots$

DI Axiom:

$[\{x'=f \& Q\}]P \leftrightarrow ([?Q]P \leftarrow (Q \rightarrow [\{x'=f \& Q\}]P'))$

Example:

$[v'=r_p v^2-g, t'=1] v \geq v_0 - gt \leftrightarrow$

… $\leftrightarrow$

$[v' := r_p v^2-g][t' := 1] v' \geq -g * t' \leftrightarrow$

$r_p v^2-g \geq -g \leftrightarrow$

$H \rightarrow r_p \geq 0$

Automation and Tooling

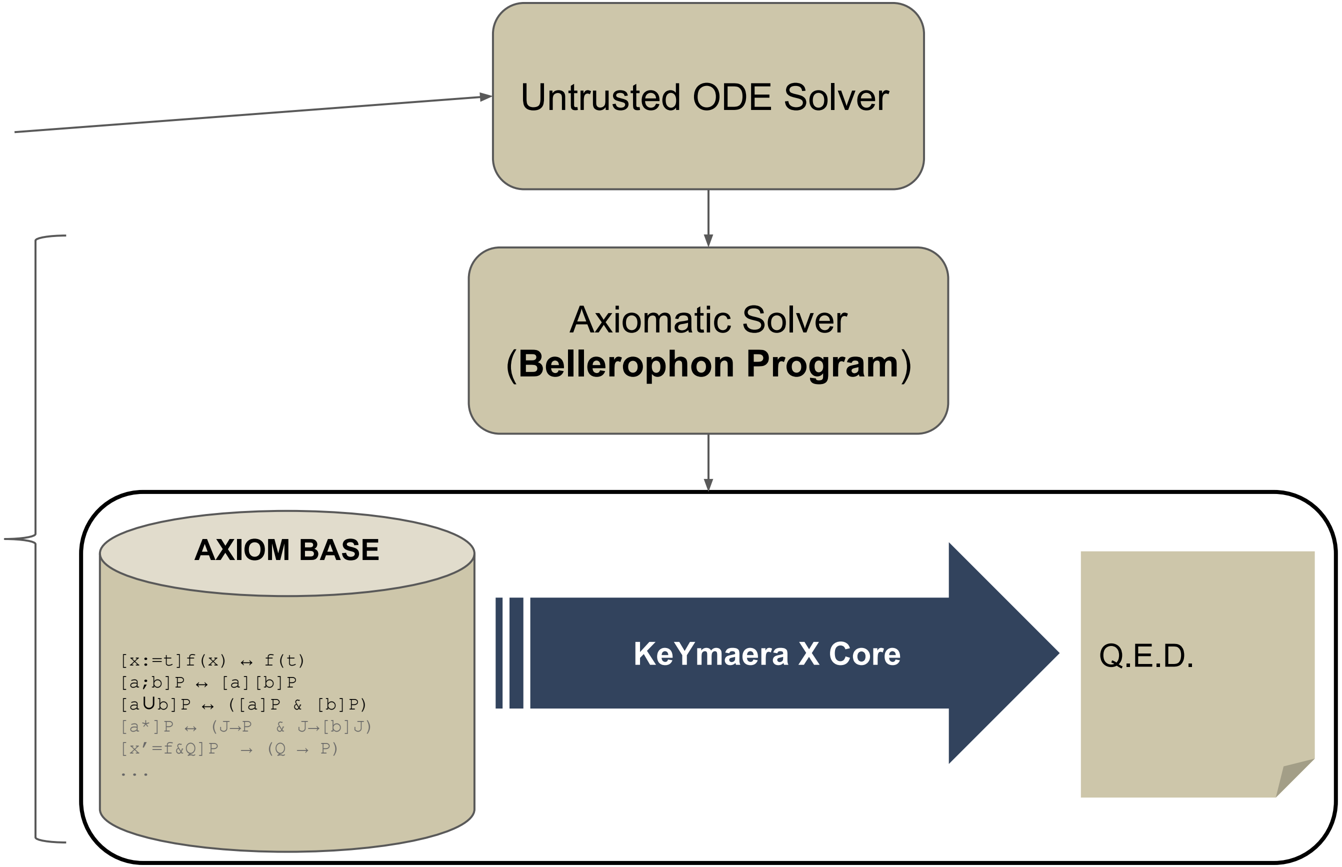
Hybrid Systems Analyses can be built on top of KeYmaera X.

Examples:

- ODE Solver
- Runtime Monitoring

Solving Differential Equations

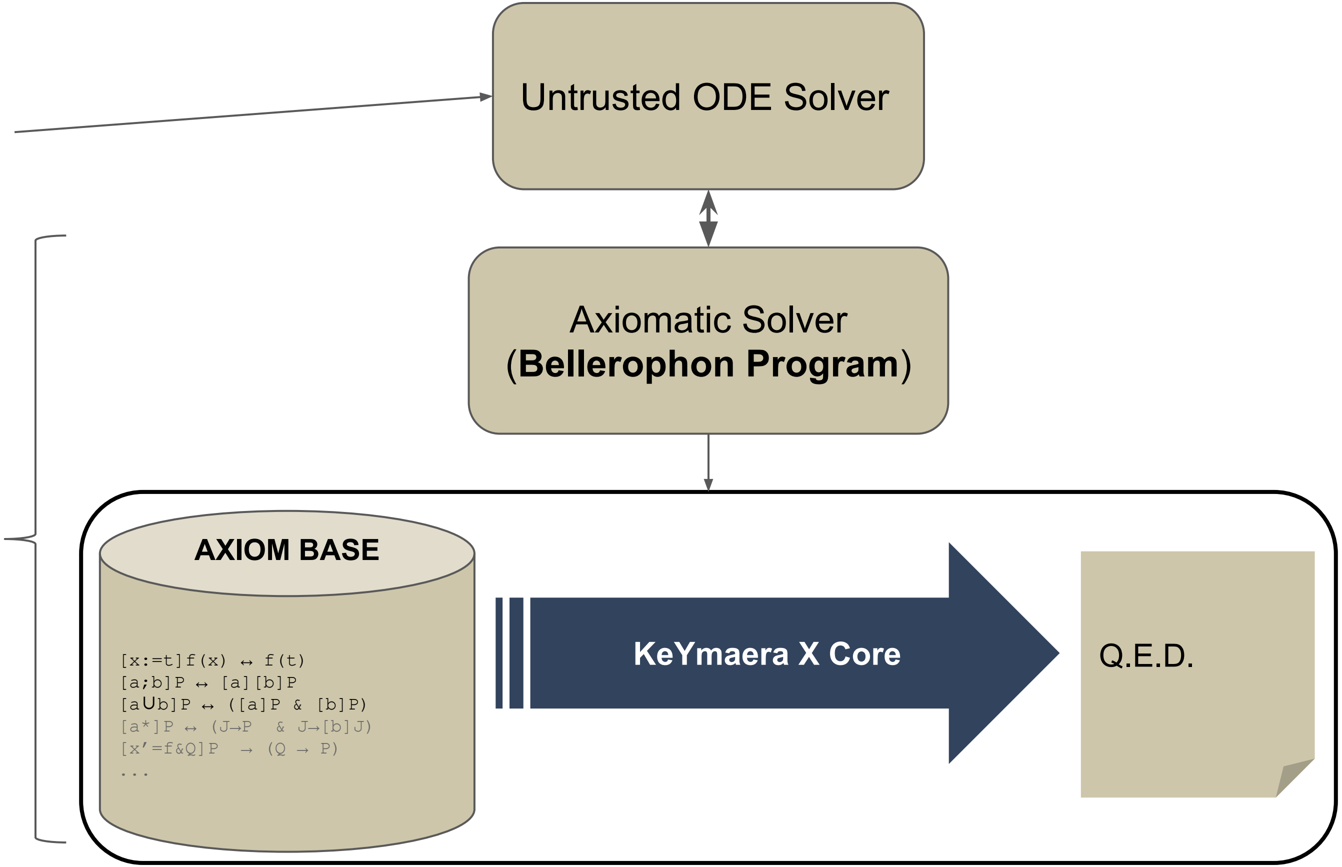
1. Use untrusted code to find a conjecture.
2. Prove the conjecture systematically, leveraging standard library.



Solving Differential Equations

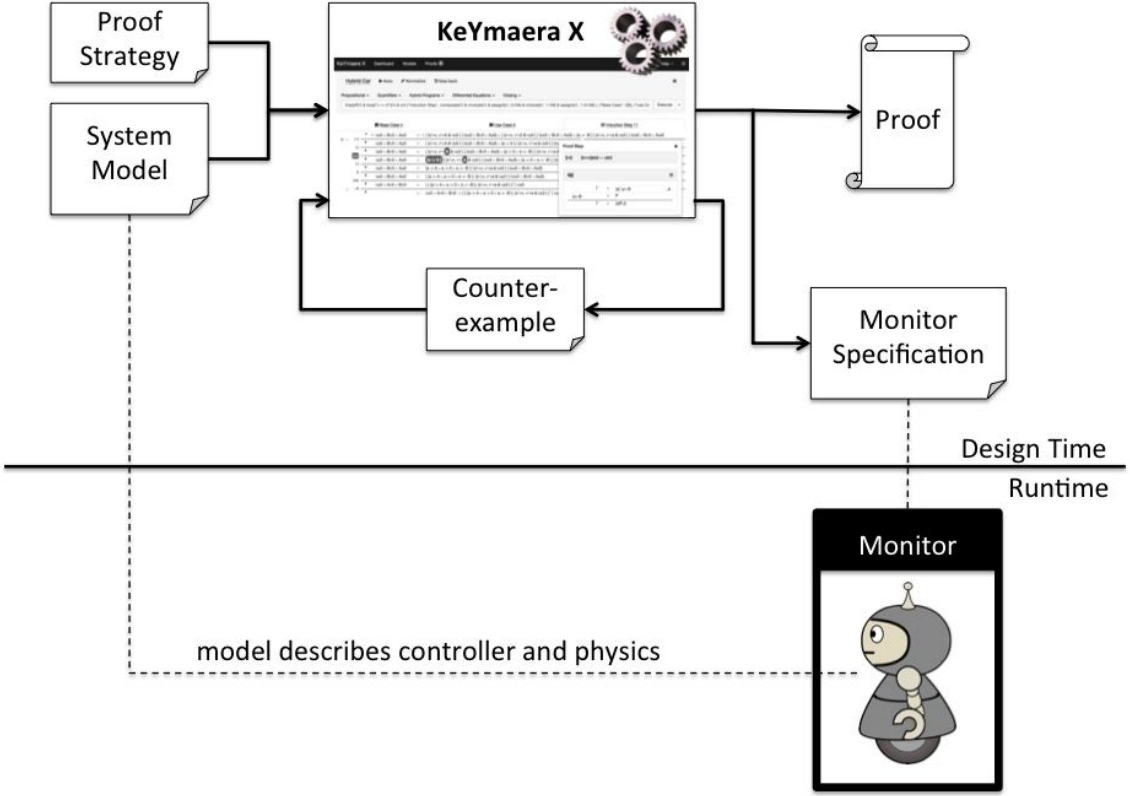
1. Use untrusted code to find a conjecture.

2. Prove the conjecture systematically, leveraging standard library.



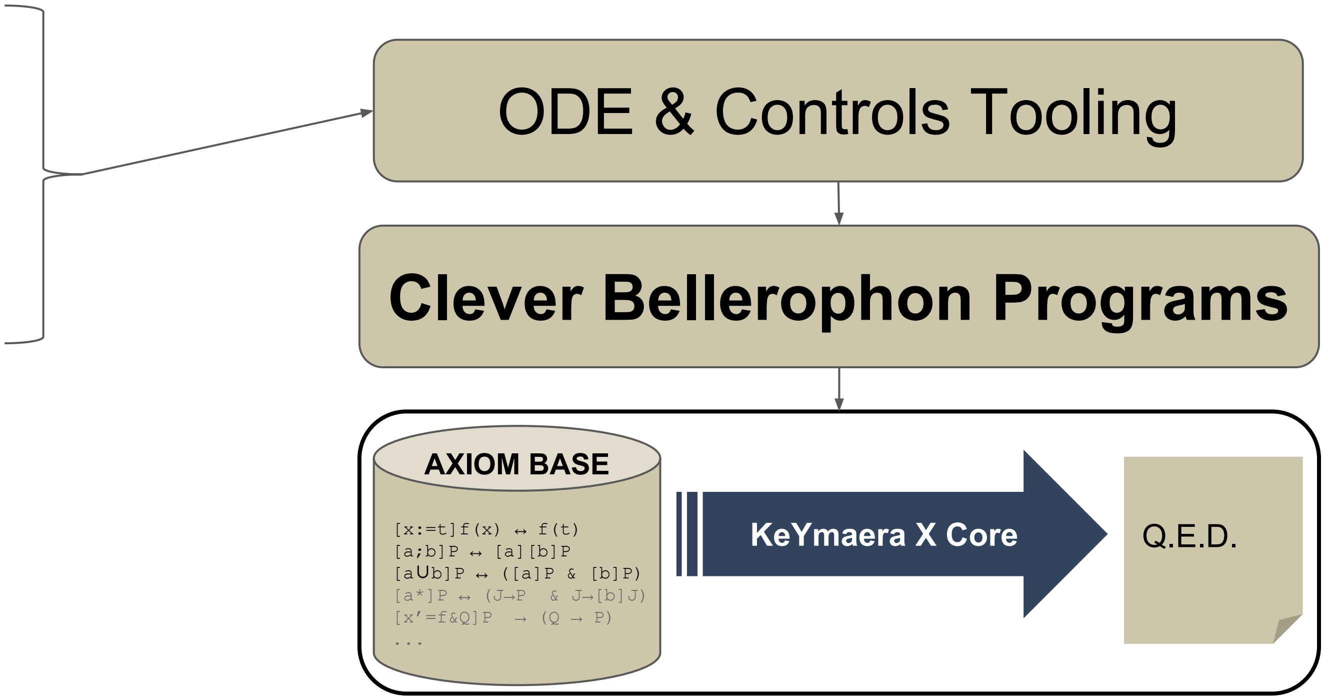
Automation and Tooling

ModelPlex Tactic



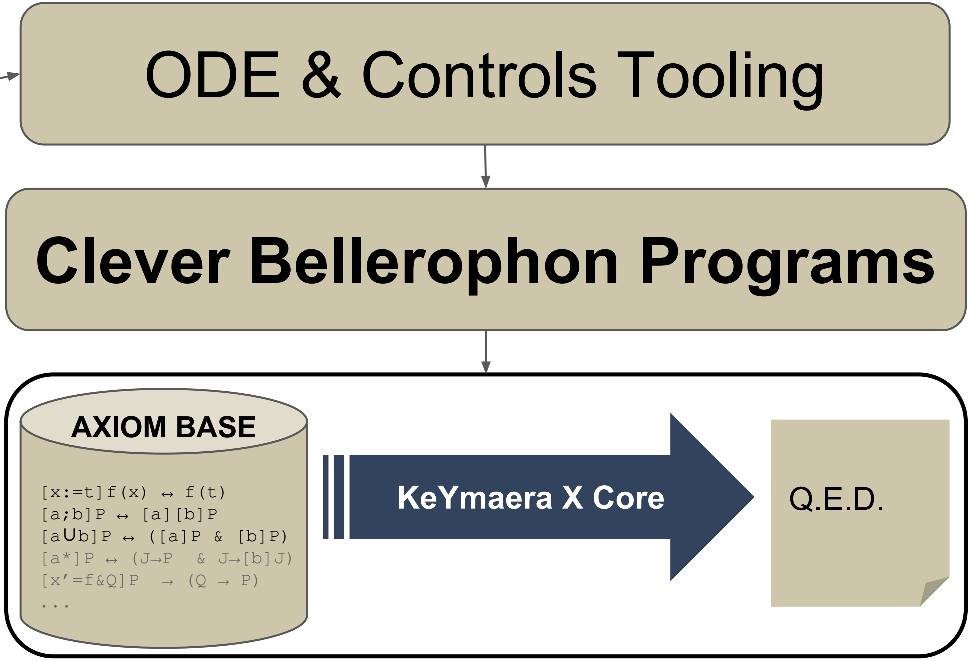
Other Proof Automation & Tooling

- **Taylor Series**
- **Bifurcations**
- Limit Cycles
- Numerical tools
- ...



Other Proof Automation & Tooling

- **Taylor Series**
- **Bifurcations**
- Limit Cycles
- Numerical tools
- ...



Other Tooling:

- Component-based Verification
- Web UI

Conclusion

There is a wide gap between **sound foundations** for hybrid systems and **practical interactive theorem proving technology** for cyber-physical systems verification.

Conclusion

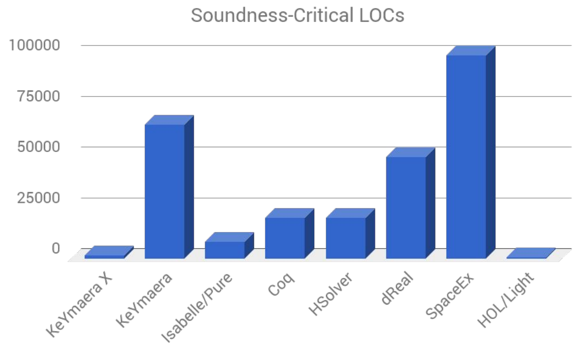
There is a wide gap between **sound foundations** for hybrid systems and **practical interactive theorem proving technology** for cyber-physical systems verification.

Bellerophon demonstrates how to verify hybrid systems using tactics.

Conclusion

There is a wide gap between **sound foundations** for hybrid systems and **practical interactive theorem proving technology** for cyber-physical systems verification.

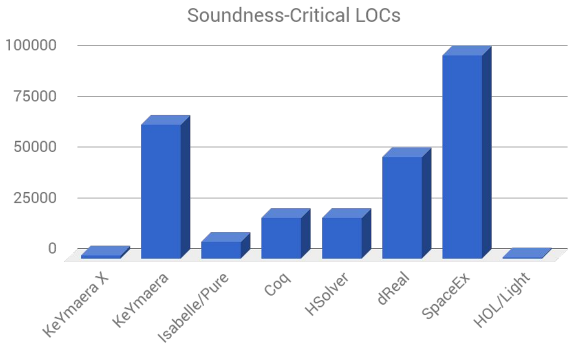
Bellerophon demonstrates how to verify hybrid systems using tactics.



Conclusion

There is a wide gap between **sound foundations** for hybrid systems and **practical interactive theorem proving technology** for cyber-physical systems verification.

Belleroph on demonstrates how to verify hybrid systems using tactics.



dI Tactic:

Side derivation:
$(v \geq v_0 - gt)' \leftrightarrow$
$\ldots \leftrightarrow$
$\ldots \leftrightarrow$
$\ldots$

$H = r_p \geq 0$ & $r_a \geq 0$ & $g > 0$ & ...

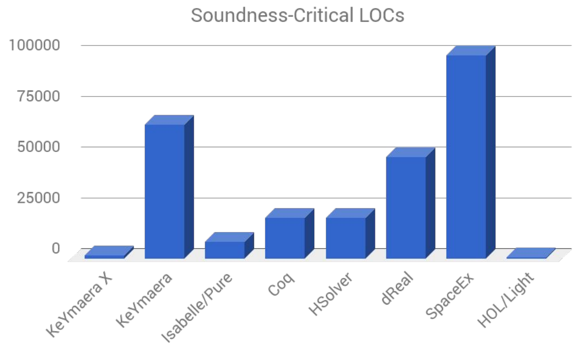
DI Axiom:
$[\{x'=f\&Q\}]P \leftrightarrow ([?Q]P \leftarrow (Q \rightarrow [\{x'=f\&Q\}]P'))$

Example:
$[v'=r_p v^2-g, t'=1]v \geq v_0 - gt \quad \leftrightarrow$
$\ldots \quad \leftrightarrow$
$[v':=r_p v^2-g][t':=1]v' \geq -g*t' \quad \leftrightarrow$
$r_p v^2-g \geq -g \quad \leftrightarrow$
$H \rightarrow r_p \geq 0$

Conclusion

There is a wide gap between **sound foundations** for hybrid systems and **practical interactive theorem proving technology** for cyber-physical systems verification.

Belleraphon demonstrates how to verify hybrid systems using tactics.



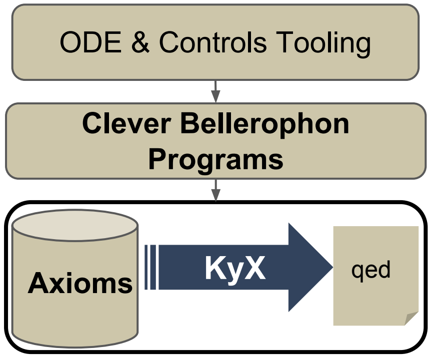
dI Tactic:

Side derivation:
$(v \geq v_0 - gt)' \leftrightarrow$
$\ldots \leftrightarrow$
$\ldots \leftrightarrow$
$\ldots$

$H = r_p \geq 0$ & $r_a \geq 0$ & $g > 0$ & ...

DI Axiom:
$[\{x'=f \& Q\}]P \leftrightarrow ([?Q]P \leftarrow (Q \rightarrow [\{x'=f \& Q\}]P'))$

Example:
$[v'=r_p v^2-g, t'=1]v \geq v_0 - gt \leftrightarrow$
$\ldots \leftrightarrow$
$[v':=r_p v^2-g][t':=1]v' \geq -g*t' \leftrightarrow$
$r_p v^2-g \geq -g \leftrightarrow$
$H \rightarrow r_p \geq 0$



Conclusion

There is a wide gap between **sound foundations** for hybrid systems and **practical interactive theorem proving technology** for cyber-physical systems verification.

Bellerophon demonstrates how to verify hybrid systems using tactics.

Project Website (start here)	keymaeraX.org
Online Demo	web.keymaeraX.org
Open Source (GPL)	github.com/ls-lab/KeYmaeraX-release

Thanks: 15-424 students, **Jean-Baptiste Jeannin**, Khalil Ghorbal, **Yanni Kouskoulas** et al., and many others!

Developers:

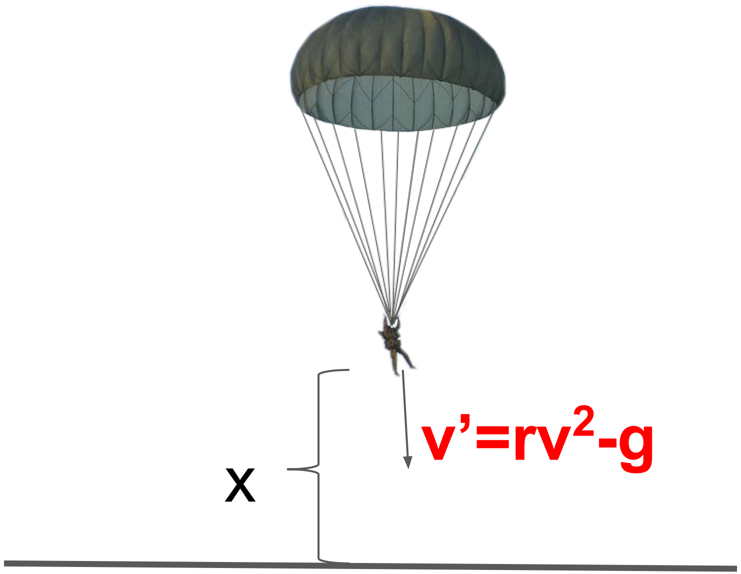
- Stefan Mitsch
- Nathan Fulton
- André Platzer
- Brandon Bohrer
- Jan-David Quesel
- Yong Kiam Tan
- Markus Völp

Differential Ghosts

Parachute Closed:

$$J \ \& \ t=0 \ \& \ r=r_p \ \rightarrow$$
$$[x'=v, v'=rv^2-g \ \& \ 0 \leq x \ \& \ t \leq T] v>-\mathrm{sqrt}(g/pr) \ > \ m$$

Proof requires a **differential ghost** because the property is **not inductive**.



Differential Ghosts

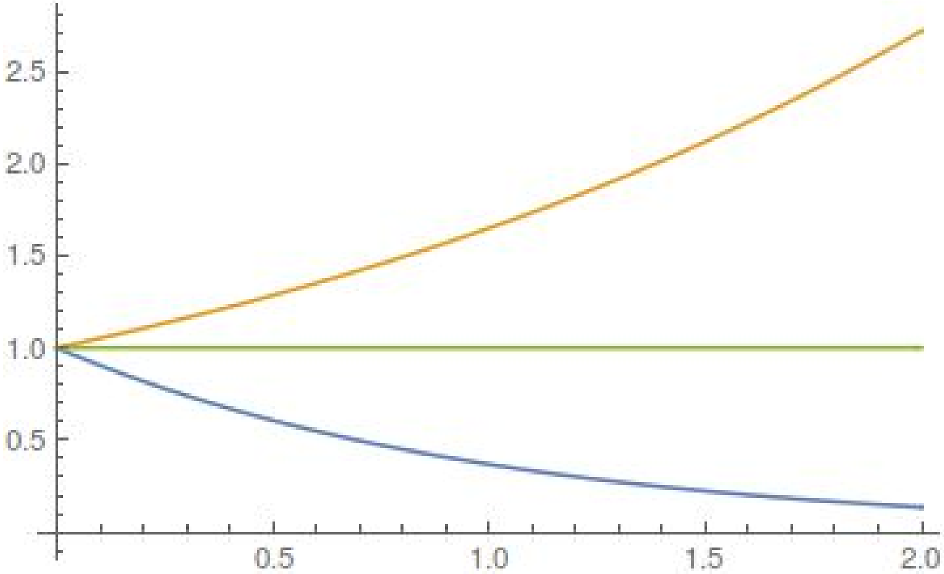
An example differential ghost.

$$x>0 \rightarrow [x'=-x]x>0$$

Differential Ghosts

An example differential ghost.

```
x>0 → [x'=-x]x>0
Ghost:      y'=y/2
Conserved: 1=xy²
```



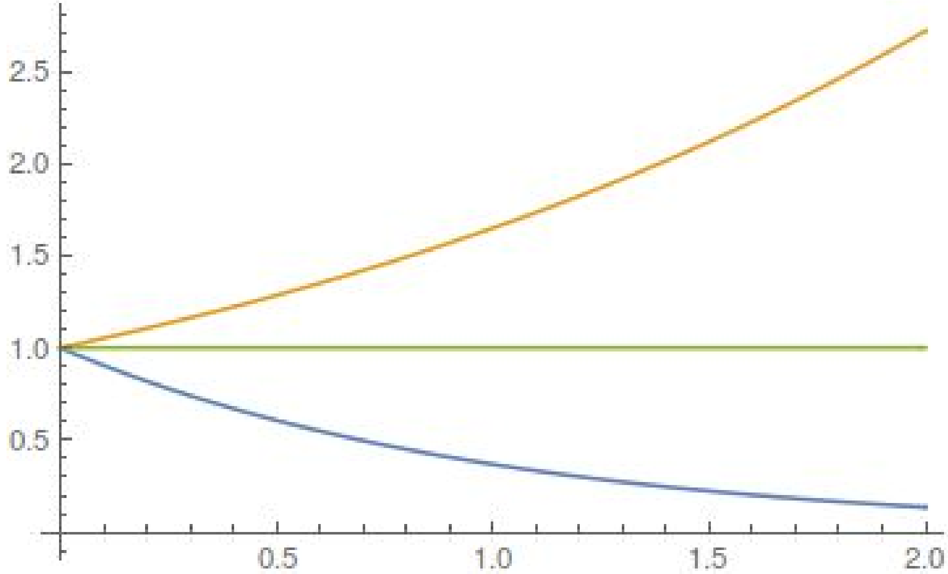
Differential Ghosts

An example differential ghost.

$x>0 \rightarrow [x'=-x]x>0$

Ghost: $y'=y/2$

Conserved: $1=xy^2$



Notice:

$x>0 \leftrightarrow \exists y. 1=xy^2$

Therefore, suffices to show:

$1=xy^2 \rightarrow \exists y. [x'=-x, y'=y/2] 1=xy^2$

Prover Core Comparison

Tool	Trusted LOC (approx.)
KeYmaera X	**1,682 (out of 100,000+)**
KeYmaera	65,989
Isabelle/Pure	8,113
Coq	20,000
HSolver	20,000
dReal	50,000
SpaceEx	100,000