Compositional and Distributed Verification of Distributed Hybrid Systems

David Renshaw, Ping Hou, André Platzer, Sarah M. Loos

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA

NSF Expeditions in Computing

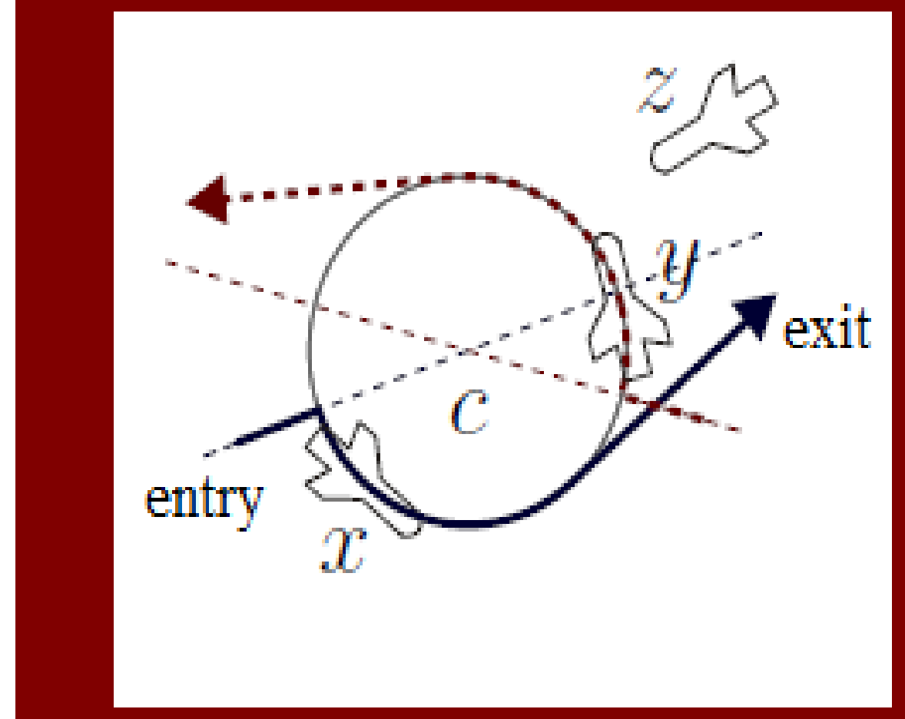
Overview

Issue: Distributed hybrid systems present extraordinarily challenging problems for verification. On top of the difficulties associated with distributed systems, they also exhibit continuous dynamics. Handling the arithmetic challenges can be extremely expensive. How do we succinctly specify a compositional and distributed verification strategy, and how do we control the computational cost?

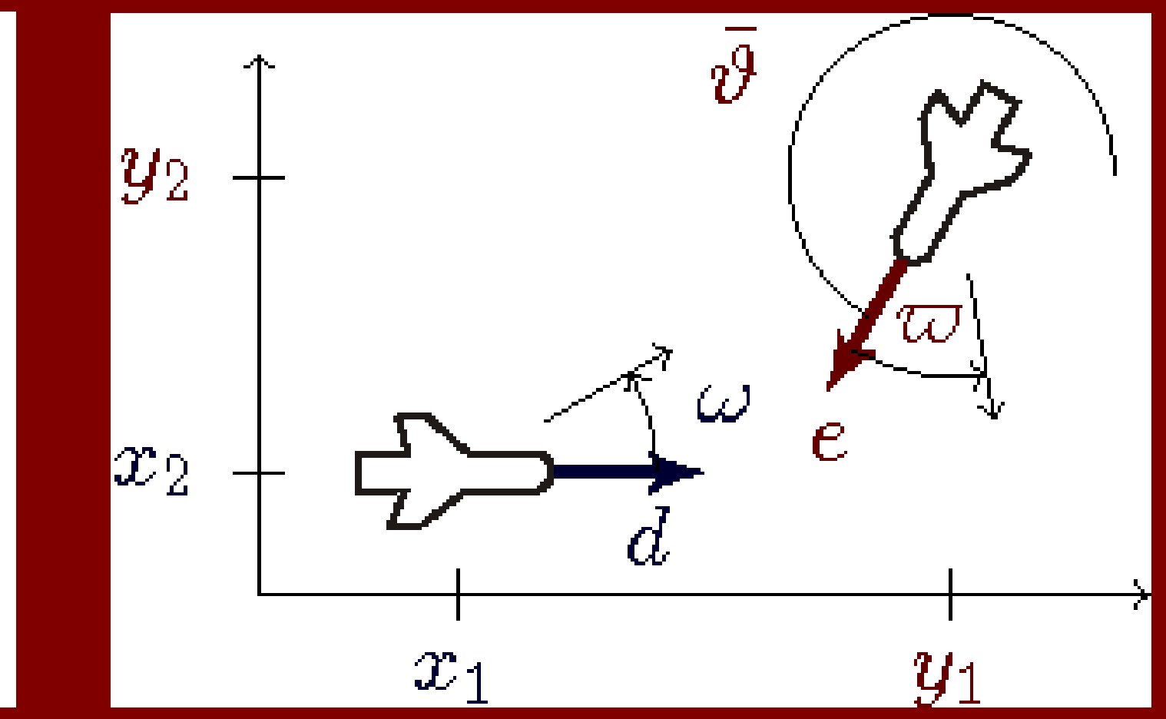
Objective: Develop a compositional and distributed verification tool for distributed hybrid systems, which has a distributed verification engine. Using a distributed verifying backend enables us to overcome the high computational complexity of distributed hybrid systems verification.

Distributed Hybrid Systems Verification

Discrete, Continuous, Distributed:



Distributed roundabout maneuvers



Flight dynamics

Initially safe → [(step)*] No collision

step: free dynamics; discrete coordination; coordinated dynamics

continuous dynamics (quantified differential equation): $\forall i\ x(i)' = d(i)$

discrete dynamics (quantified control decisions): $\forall i\ d(i) \coloneqq$ if .. then a else $-b$ fi

dimensional dynamics (appearance): $n \coloneqq$ new Aircraft

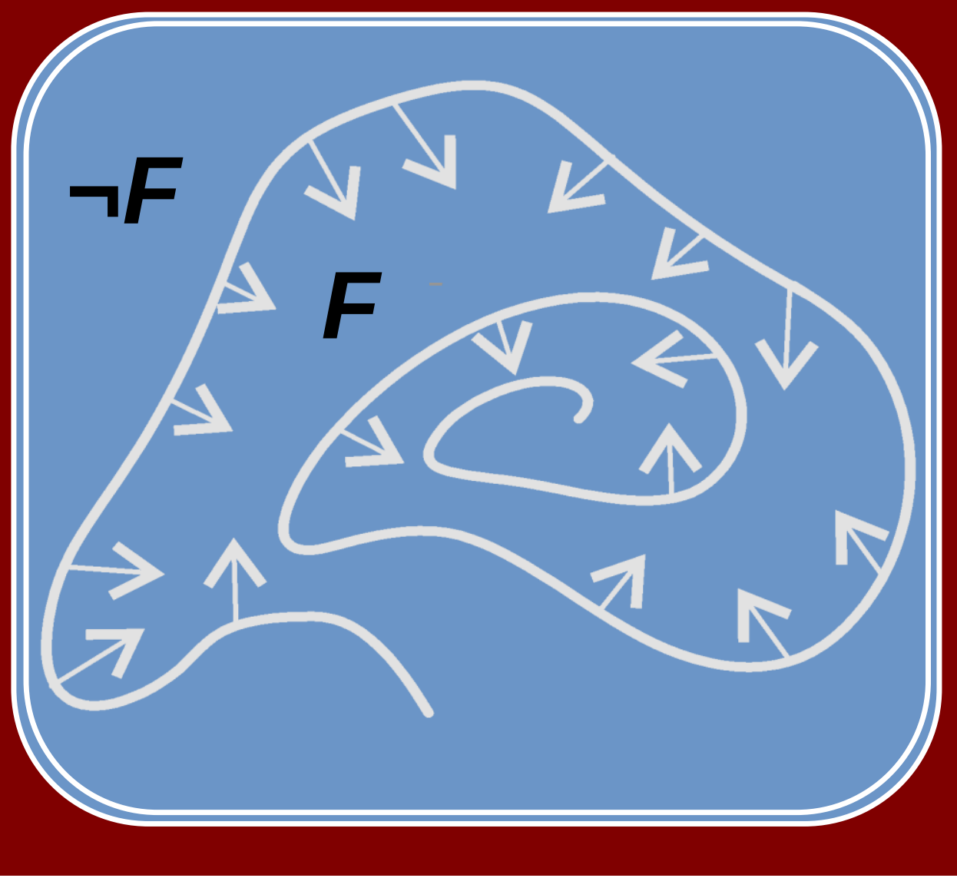
continuous dynamics of an aircraft i with an angular velocity $\omega(i)$:

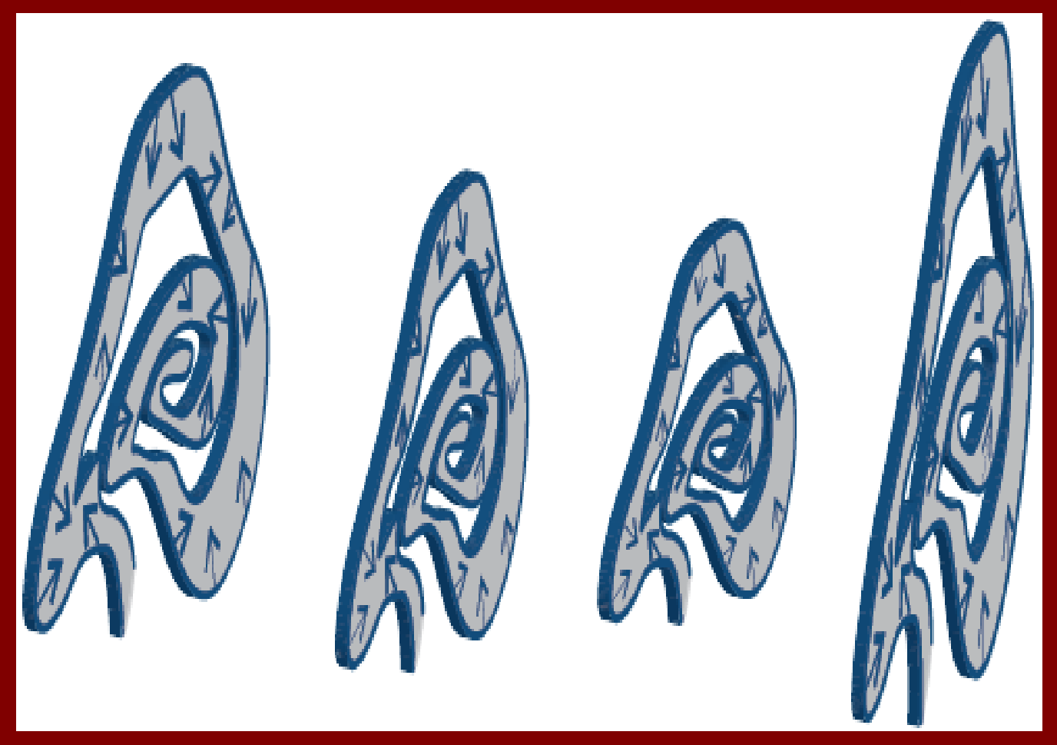
$$F_{\omega(i)}(i)\text{: } x_1(i)' = d_1(i), x_2(i)' = d_2(i), d_1(i)' = -\omega(i) d_2(i), d_2(i)' = \omega(i) d_1(i)$$

continuous dynamics of all aircraft i : $\forall i\ F_{\omega(i)}(i)$

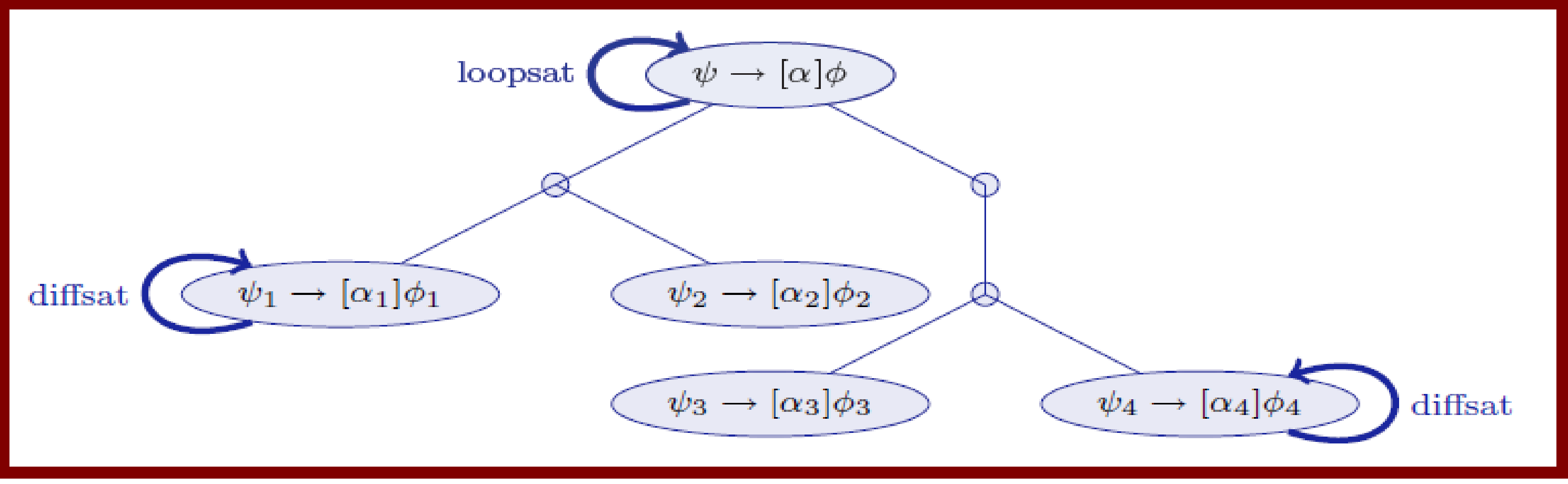
Differential Invariants:

Differential invariant ***F***





Quantified differential invariant

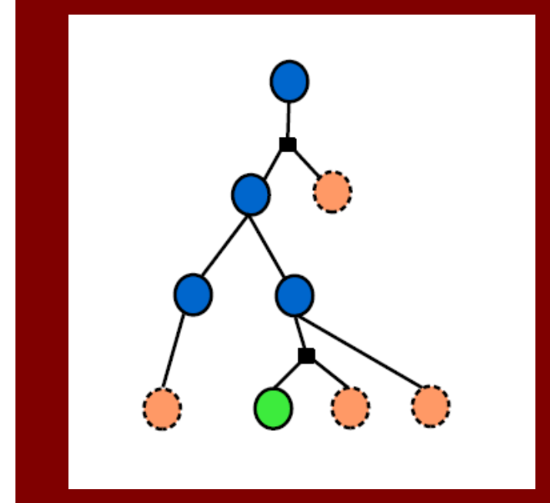


loopsat
$\psi \to [\alpha]\phi$
diffsat
$\psi_1 \to [\alpha_1]\phi_1$
$\psi_2 \to [\alpha_2]\phi_2$
$\psi_3 \to [\alpha_3]\phi_3$
$\psi_4 \to [\alpha_4]\phi_4$
diffsat

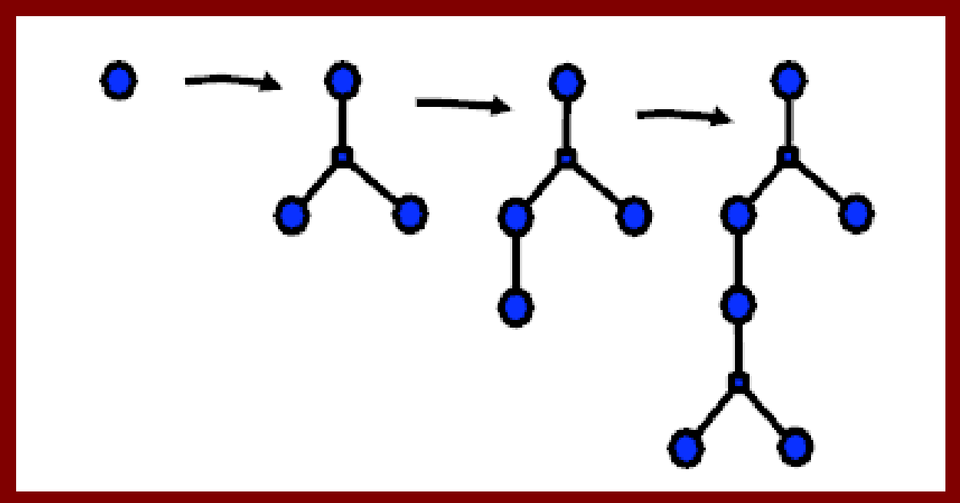
Fixed-points verification for invariants

KeYmaeraD

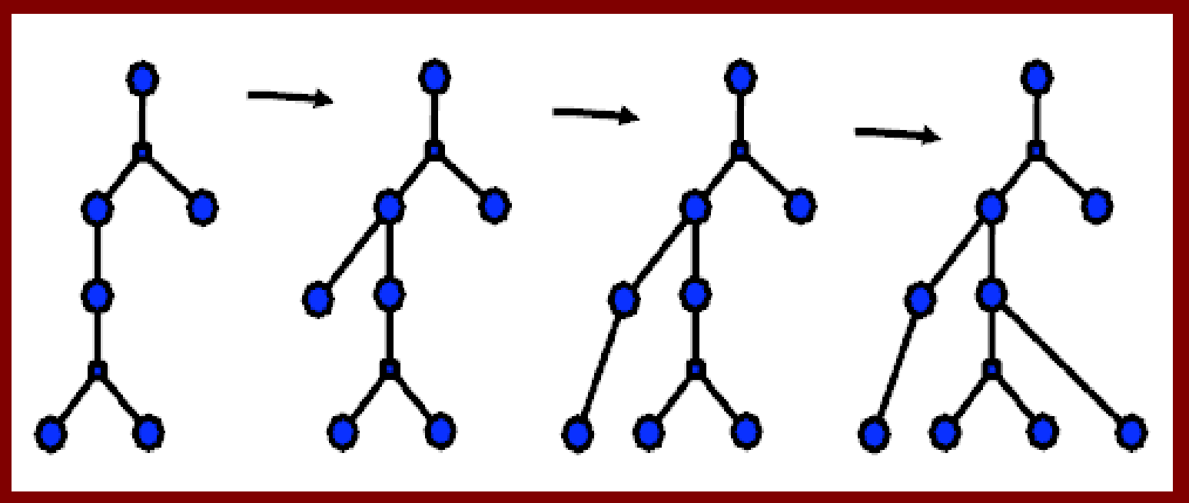
Verifying in KeYmaeraD:



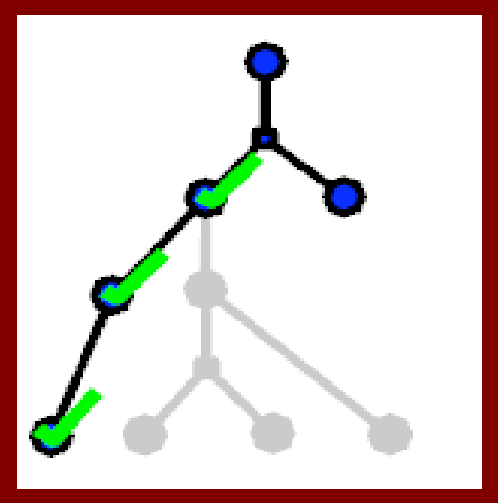
Verification state



Decompose subproblems

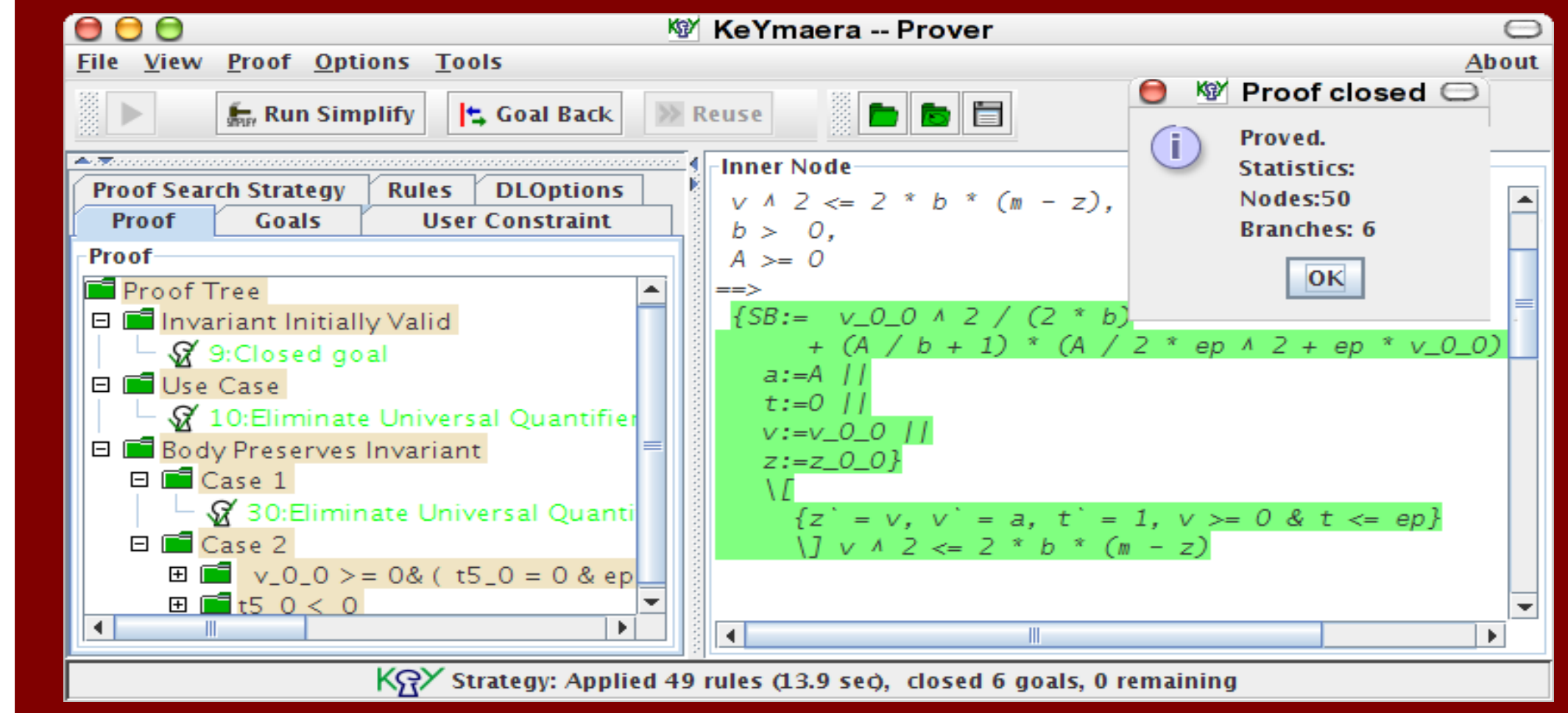


Or-branching for choices and decomposition



Closing an or-branching

KeYmaera and KeYmaeraD:



Proof closed

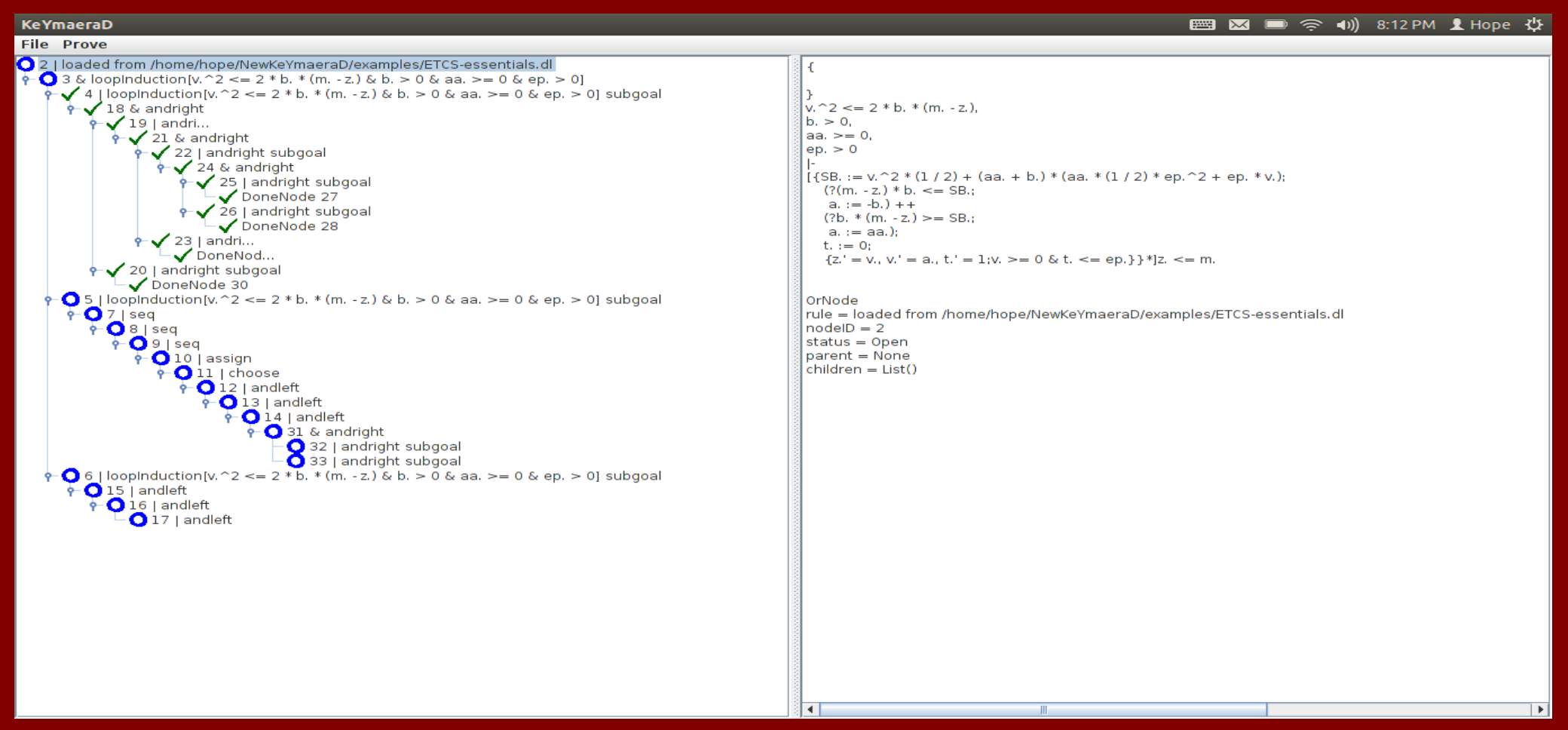
Proved.
Statistics:
Nodes:50
Branches: 6

```
Proof
Proof Tree
  Invariant Initially Valid
    9:Closed goal
  Use Case
    10:Eliminate Universal Quantifier
  Body Preserves Invariant
    Case 1
      30:Eliminate Universal Quanti
    Case 2
      v_0_0 >= 0& ( t5_0 = 0 & ep
      t5_0 < 0
```

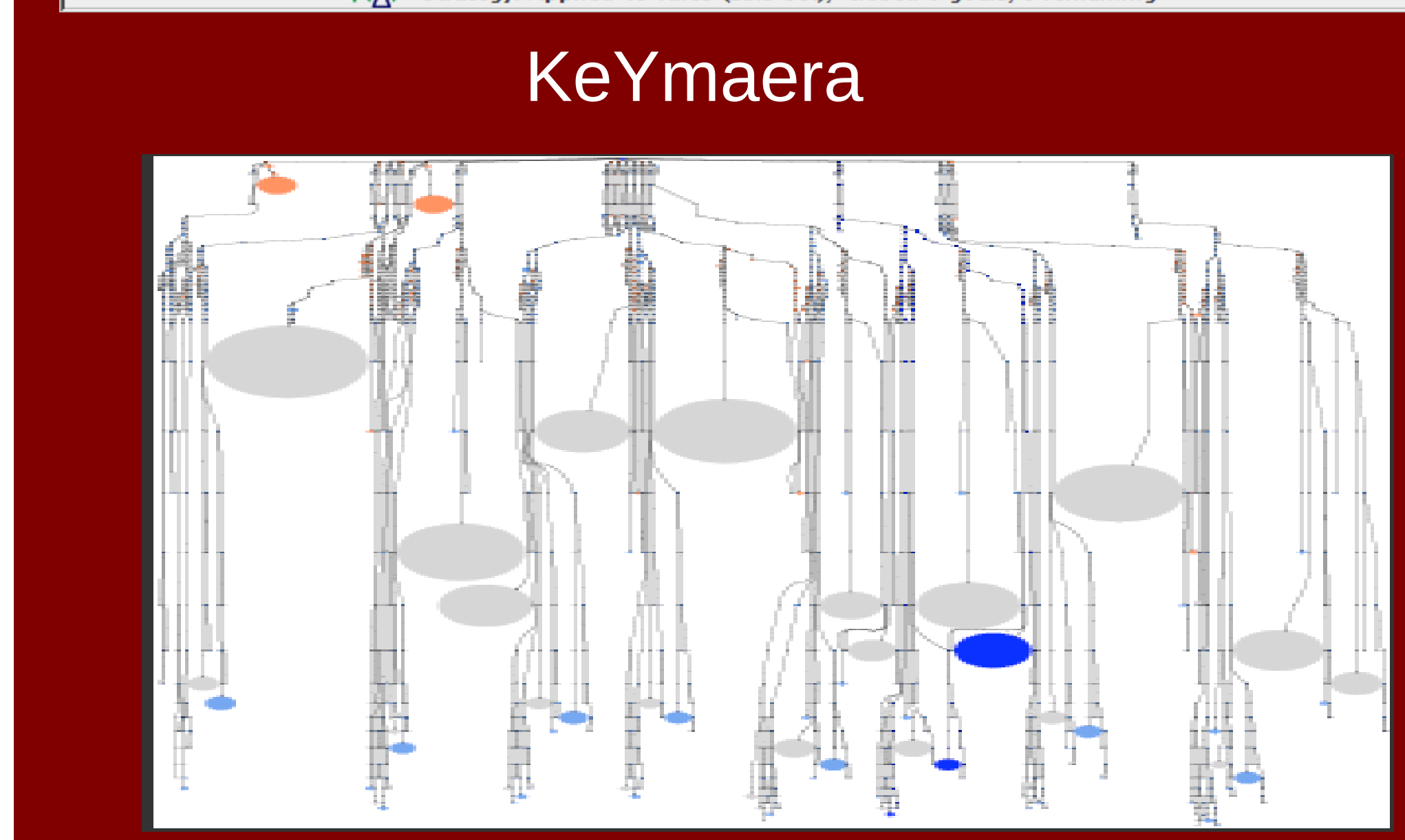
```
v ^ 2 <= 2 * b * (m - z),
b >  0,
A >= 0
==>
{SB:=  v_0_0 ^ 2 / (2 * b)
     + (A / b + 1) * (A / 2 * ep ^ 2 + ep * v_0_0)
  a:=A ||
  t:=0 ||
  v:=v_0_0 ||
  z:=z_0_0}
  \[
   {z` = v, v` = a, t` = 1, v >= 0 & t <= ep}
  \] v ^ 2 <= 2 * b * (m - z)
```

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

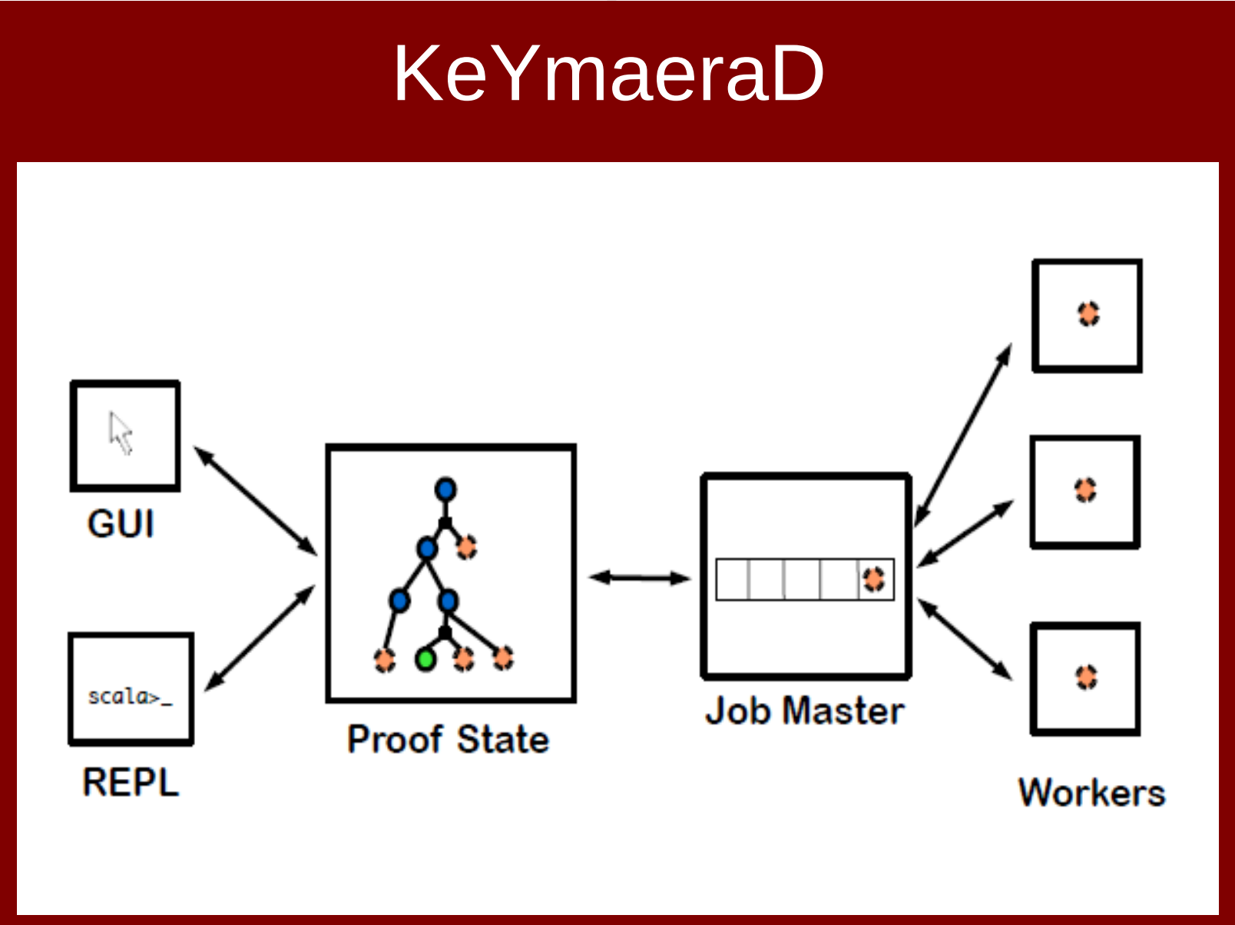
KeYmaera



KeYmaeraD



The cost of verification is concentrated in the tree's leaves



KeYmaeraD tool architecture

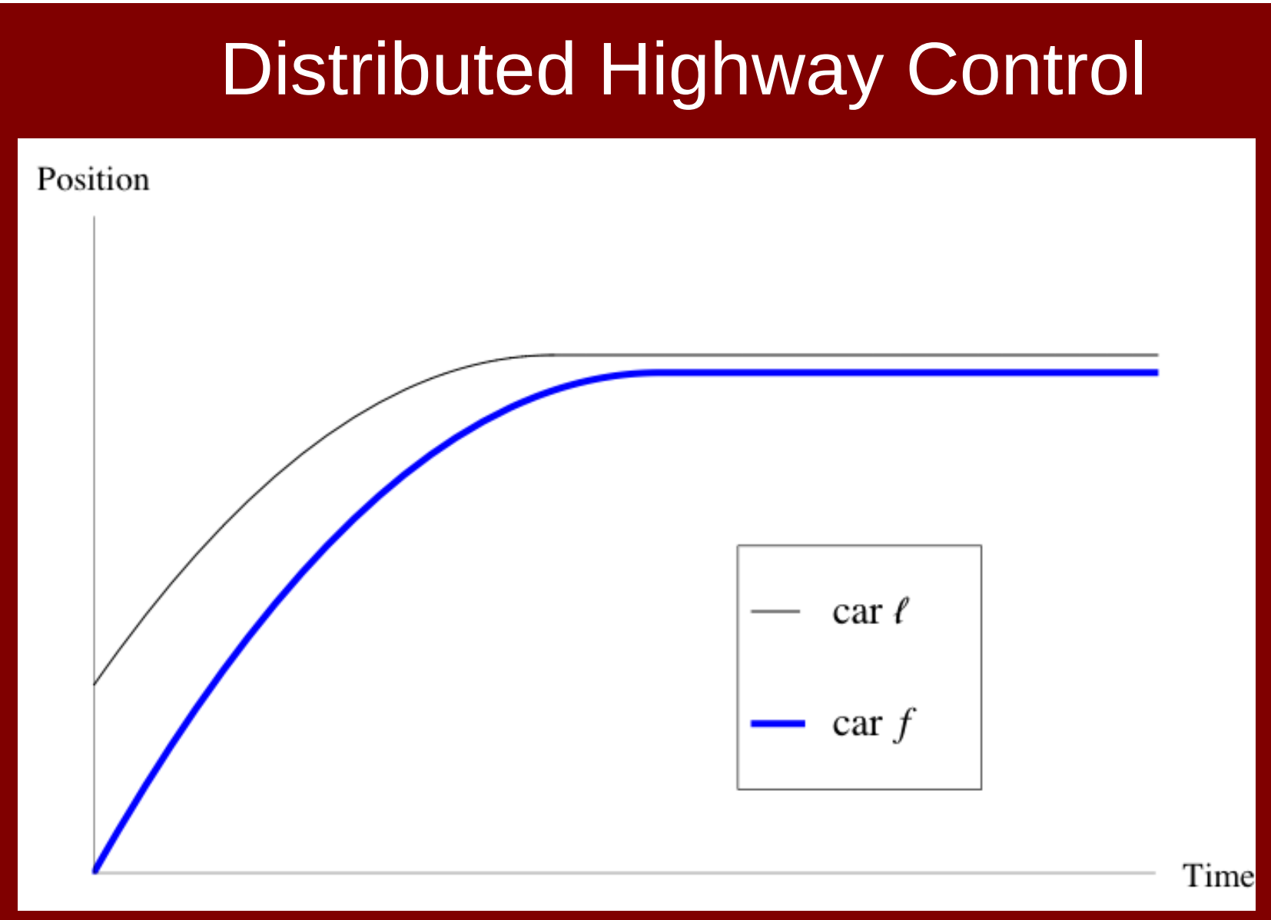
Case Study:

Distributed Highway Control

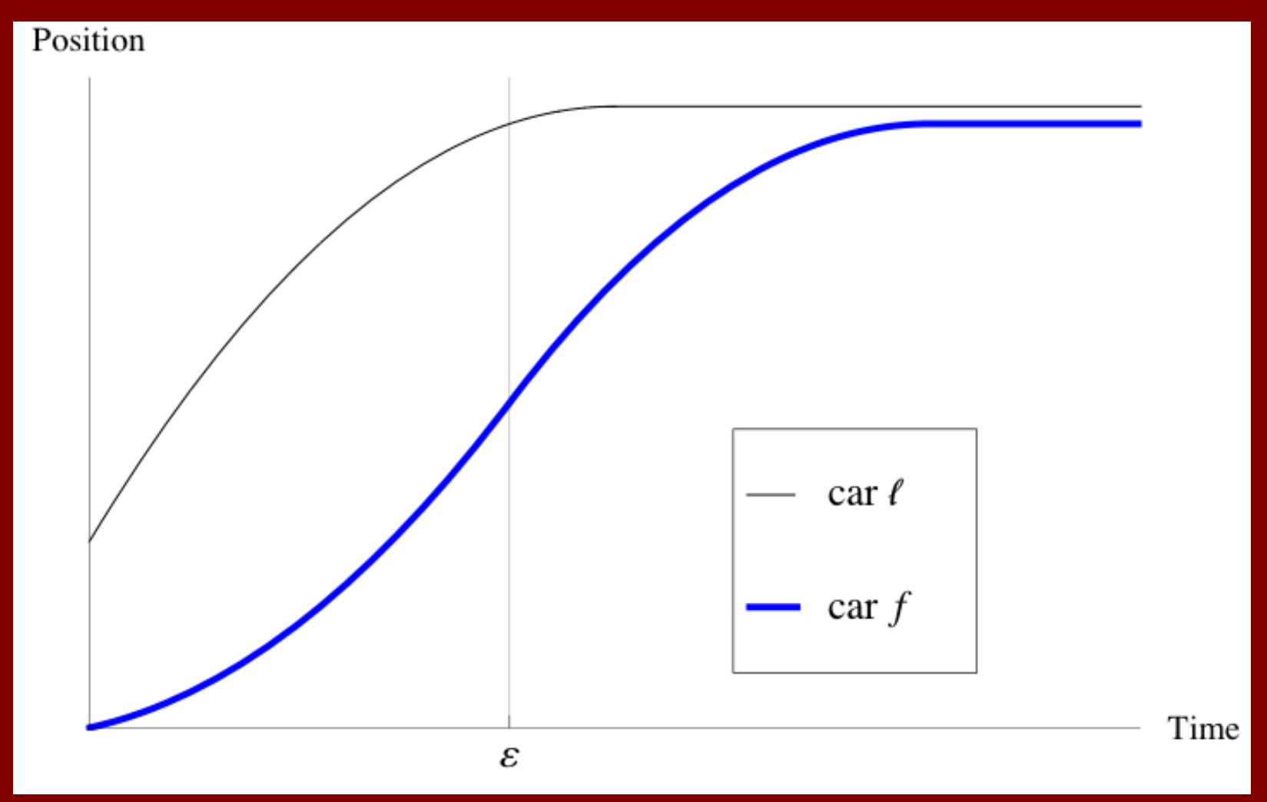
Initially safe → [(step)*] No collision

invariant → [(step)*] invariant

step: exit ∪ enter ∪ (control; dynamics)



Loop invariant



Safe to accelerate?

Verification Statistics:

synchronous control:
verification state: 1134 nodes
one worker time: 40 seconds
two workers time: 33 seconds
2.86GHz Core 2 Duo

asynchronous control:
verification state: 7154 nodes
one worker time: 640 seconds
four workers time: 195 seconds
2.83GHz intel Core2 quad core