Reachability Analysis in the KeYmaera X Theorem Prover

SNR 2017 | Uppsala, Sweden | April 22, 2017

Nathan Fulton

Other System Contributors: Stefan Mitsch, André Platzer, Brandon Bohrer, Yong Kiam Tan, Jan-David Quesel, ...
Trustworthy Foundations

Interactive Reachability Analysis
- Demonstration
- Bellerophon language and library

Automation and Tooling

Conclusions & Resources
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**.
$a := t$

\[
\begin{align*}
a &= a_0 \\
b &= b_0 \\
c &= c_0 \\
\ldots
\end{align*}
\]

\[
\begin{align*}
a &= t \\
b &= b_0 \\
c &= c_0 \\
\ldots
\end{align*}
\]
$a := t$

```
a = a_0
b = b_0
c = c_0
...
```

```
a = t
b = b_0
c = c_0
...
```

$a; b$
```
Trustworthy Foundations
Hybrid Programs

a := t

?P
If P is true: no change
If P is false: terminate
```
a := t

If P is true: no change
If P is false: terminate
Trustworthy Foundations
Hybrid Programs

\[ a := t \]
\[ \begin{array}{l}
\text{a} = a_0 \\
\text{b} = b_0 \\
\text{c} = c_0 \\
\end{array} \quad \rightarrow \quad \begin{array}{l}
\text{a} = t \\
\text{b} = b_0 \\
\text{c} = c_0 \\
\end{array} \]

\[ a \cup b \]

\[ \text{If P is true: no change} \]
\[ \text{If P is false: terminate} \]

\[ a^* \]

\[ a ; b \]

\[ a \cup b \]
**Trustworthy Foundations**

**Hybrid Programs**

\[ a := t \]

\[ a = a_0 \quad b = b_0 \quad c = c_0 \]

...\[ a = t \quad b = b_0 \quad c = c_0 \]

...\[ x' = f \]

If \( P \) is true: no change

If \( P \) is false: terminate

\[ a \cup b \]

\[ x = F(0) \]

...\[ x = x_0 \]

...\[ x = F(T) \]

\[ a ; b \]

...\[ a ; b \]

...\[ a ; b \]

...\[ a ; b \]
\[
\begin{align*}
\{ \text{?Dive U } r & := r_p \}; \\
& t := 0; \\
& \{ x' = v, \\
& \quad V' = f(v, g, r), \quad t' = 1 \\
& \quad \& 0 \leq x \quad \& t \leq T \}
\end{align*}
\]

*Control*: Continue diving if safe, else open parachute.

*Plant*: Downward velocity determined by gravity, air resistance.
\begin{aligned} &\{?\text{Dive} \cup r := r_p\}; \\
& t := 0; \\
& \{x' = v, \ \\
& V' = f(v, g, r), \ t' = 1 \\
& \& 0 \leq x \& t \leq T\} \end{aligned}

\textbf{Control:} Continue diving if safe, else open parachute.

\textbf{Plant:} Downward velocity determined by gravity, air resistance.
\[
\{ \text{?Dive} \cup r := r_p \};
\]
\[
t := 0;
\]
\[
\{ x' = v, \\
V' = f(v, g, r), \quad t' = 1 \}
\]
\[
& \quad 0 \leq x \quad \& \quad t \leq T
\]

\* 

**Control**: Continue diving if safe, else open parachute.

**Plant**: Downward velocity determined by gravity, air resistance.
\[
\begin{align*}
\{ & \text{Dive} \cup r := r_p \}; \\
& t := 0; \\
& \{ x' = v, \\
& \quad V' = f(v, g, r), \quad t' = 1 \\
& \quad & 0 \leq x \quad \& \quad t \leq T \}
\}
\]

**Control:** Continue diving if safe, else open parachute.

**Plant:** Downward velocity determined by gravity, air resistance.
\{
  \{ ?Dive \cup r := r_p \};
  t := 0;
  \{ x' = v, \\
      V' = f(v, g, r), \thinspace t' = 1 \\
      \& \thinspace 0 \leq x \thinspace \& \thinspace t \leq T \}
\} *

**Control:** Continue diving if safe, else open parachute.

**Plant:** Downward velocity determined by gravity, air resistance.
[a]P  "after every execution of a, P"
<a>P  "after some execution of a, P"
(Dive & g>0 & ...) →

\[
\{ \\
{ \text{?Dive} \cup r := r_p } ; \\
{ x' = v } , \\
{ V' = f(v, g, r) } \\
& 0 \leq x \\
\} \ast (x=0 \rightarrow m \leq v)
\]
(Dive & g>0 & ...) →
{
  {?Dive \cup r := r_p};
  {x' = v, V' = f(v, g, r)
   \& 0 \leq x}
\}
(*) (x=0 \rightarrow m \leq v)

If the parachuter is on the ground, their speed is safe (m \leq v \leq 0)
Introduction to Differential Dynamic Logic

**Dynamical Axioms**

\[
[x:=t]f(x) \leftrightarrow f(t)
\]
\[
[a;b]P \leftrightarrow [a][b]P
\]
\[
[a\cup b]P \leftrightarrow ([a]P \& [b]P)
\]
\[
[a^*]P \leftrightarrow (J\rightarrow P \& J\rightarrow [b]J)
\]
\[
[x'=f\&H]P \leftrightarrow H\rightarrow P
\]

...
Introduction to Differential Dynamic Logic

Trusted Core

AXIOM BASE

[x:=t]f(x) ↔ f(t)
[a;b]P ↔ [a][b]P
[a∪b]P ↔ ([a]P & [b]P)
[a*]P↔(J→P & J→[b]J)
[x'=f&H]P ↔ H→P
...

KeYmaera X Core

Q.E.D.
Introduction to Differential Dynamic Logic

Trustworthy Implementations

Automated Analyses

Control Software

Tooling

AXIOM BASE

\[
\begin{align*}
[x:=t]f(x) & \rightarrow f(t) \\
[a;b]P & \rightarrow [a][b]P \\
[a\cup b]P & \leftrightarrow ([a]P \land [b]P) \\
[a^*]P & \leftrightarrow (J\rightarrow P \land J\rightarrow[b]J) \\
[x'=f\&H]P & \leftrightarrow H\rightarrow P \\
\ldots
\end{align*}
\]

KeYmaera X Core

Q.E.D.
## Prover Core Comparison

<table>
<thead>
<tr>
<th>Tool</th>
<th>Trusted LOC (approx.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeYmaera X</td>
<td>1,682 (out of 100,000+)</td>
</tr>
<tr>
<td>KeYmaera</td>
<td>65,989</td>
</tr>
<tr>
<td>Isabelle/Pure</td>
<td>8,113</td>
</tr>
<tr>
<td>Coq</td>
<td>20,000</td>
</tr>
<tr>
<td>HSolver</td>
<td>20,000</td>
</tr>
<tr>
<td>dReal</td>
<td>50,000</td>
</tr>
<tr>
<td>SpaceEx</td>
<td>100,000</td>
</tr>
</tbody>
</table>
KeYmaera X enables interactive verification and tool development:
KeYmaera X enables interactive verification and tool development:

- A **standard library** of common proof techniques.
KeYmaera X enables interactive verification and tool development:

- **A standard library** of common proof techniques.
- **A combinator language/library** for decomposing theorems and composing proof strategies.
### Interactive Reachability Analysis in KeYmaera X

**Bellerophon**

<table>
<thead>
<tr>
<th>Tactic</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>prop</code></td>
<td>Applies propositional reasoning exhaustively.</td>
</tr>
<tr>
<td><code>unfold</code></td>
<td>Symbolically executes discrete, loop-free programs.</td>
</tr>
<tr>
<td><code>loop(J, i)</code></td>
<td>Applies loop invariance axiom to position i.</td>
</tr>
</tbody>
</table>
### Interactive Reachability Analysis in KeYmaera X

**Bellerophon**

<table>
<thead>
<tr>
<th>Tactic</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>prop</code></td>
<td>Applies propositional reasoning exhaustively.</td>
</tr>
<tr>
<td><code>unfold</code></td>
<td>Symbolically executes discrete, loop-free programs.</td>
</tr>
<tr>
<td><code>loop(J, 1)</code></td>
<td>Applies loop invariance axiom to position i.</td>
</tr>
</tbody>
</table>
### Tactic

<table>
<thead>
<tr>
<th>Tactic</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>prop</td>
<td>Applies propositional reasoning exhaustively.</td>
</tr>
<tr>
<td>unfold</td>
<td>Symbolically executes discrete, loop-free programs.</td>
</tr>
<tr>
<td>loop(J, i)</td>
<td>Applies loop invariance axiom to position i.</td>
</tr>
</tbody>
</table>

### Combinator

<table>
<thead>
<tr>
<th>Combinator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ; B</td>
<td>Execute A on current goal, then execute B on the result.</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>A*</td>
<td>Run A until it no longer applies.</td>
</tr>
<tr>
<td>A&lt;( B₀, B₁, ..., Bₙ )</td>
<td>Execute A on current goal to create N subgoals. Run Bᵢ on subgoal i.</td>
</tr>
</tbody>
</table>
Interactive Reachability Analysis in KeYmaera X

Isolating Interesting Questions

(Dive & g>0 & ...) →

[ { } * ] (x=0 → m≤ v)
Interactive Reachability Analysis in KeYmaera X

Isolating Interesting Questions

(Dive & g>0 & ...) →

\[ \{ \text{prop ; loop}(J,1) \} \]

\( x=0 \rightarrow m \leq v \)

Loop invariant holds initially

Loop invariant is preserved

Loop invariant implies safety
Interactive Reachability Analysis in KeYmaera X
Isolating Interesting Questions

\[(\text{Dive} \& g > 0 \& \ldots) \rightarrow \{ x = 0 \rightarrow m \leq v \}\]

Loop invariant holds initially

Loop invariant is preserved

Loop invariant implies safety
Interactive Reachability Analysis in KeYmaera X

Isolating Interesting Questions

\[
(Dive \& g>0 \& \ldots) \rightarrow \\
[ \{

\} \ast ] (x=0 \rightarrow m \leq v)
\]

\[
(Dive \& g>0 \& \ldots) \rightarrow \\
J
\]

\[
prop \; ; \; \text{loop}(J,1) \rightarrow \\
\]

\[
J \rightarrow [ \\
] J
\]

\[
J \rightarrow [ \\
x=0 \rightarrow m \leq v
\]

\[
J \& Dive \& r=r_a \rightarrow \\
[x'=v, v' = \ldots ] J
\]

\[
J \& r=r_p \rightarrow \\
[x'=v, v' = \ldots ] J
\]
(Dive & g>0 & ...) →
[ { prop ; loop(J,1) } ]

J →
[ ]

J →
[ ]

J & Dive & r=r_a →
[ x'=v , v' = ... ] J

J & r=r_p →
[ x'=v , v' = ... ] J
prop ; loop(J, 1) <(
  QE, /* Real arith. solver */
  QE,
  Unfold <(
    ... /* parachute open case */
    ... /* parachute closed case */
  )
)
J = v > -sqrt(g/pr) > m & ...

Parachute Open Case:

\[
\begin{align*}
v &\geq v_0 - gt \\
&\geq v_0 - gT \\
&> -sqrt(g/pr)
\end{align*}
\]

Inductive invariants
Interactive Reachability Analysis in KeYmaera X
Differential Induction

DI Axiom:
\([x'=f \& H]P \leftrightarrow (P \& (H \rightarrow [x':=f]P'))\)
DI Axiom:
$[x' = f \& H]P \leftrightarrow (P \& (H \rightarrow [x' := f]P'))$

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

$[x'=f \& H]P \leftrightarrow (P \& (H \rightarrow [x':=f]P'))$

Example:

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

$\ldots$

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

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

$r_p \geq 0$
**DI Axiom:**

\([x'=f&H]P \leftrightarrow (P \land (H \rightarrow [x' := f]P'))\)

**Example:**

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

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

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

**Side derivation:**

\(H = r_p \geq 0 \land r_a \geq 0 \land g > 0 \land ...\)

\(H \rightarrow r_p \geq 0\)
**DI Axiom:**

\[
[x'=f \& H]P \leftrightarrow (P \& (H \rightarrow [x' := f]P'))
\]

**Example:**

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

\[
\ldots
\]

\[
[v' := r_p v^2 - g] [t' := 1] v' \geq -g \times t' 
\]

\[
r_p v^2 - g \geq -g
\]

\[
H \rightarrow r_p \geq 0
\]

**Tactics recover a useful level of abstraction.**
Pedantry is the price of trust.
Pedantry is the price of trust.

Bellerophon automates pedantic deductions.
Hybrid Systems Analyses can be built on top of KeYmaera X.

Examples:
- ODE Solver
- Runtime Monitoring
1. Use untrusted code to find a conjecture.

2. Prove the conjecture systematically.

AXIOM BASE

Untrusted ODE Solver

Axiomatic Solver (Bellerophon Program)

KeYmaera X Core

Q.E.D.
Toward Automated Deduction
ModelPlex Tactic
Toward Automated Deduction

Learning how to be Safe

Dual Strategy controller with Stochastic Perturbations
\( (A=1, B=2) \)
Automated Analysis for nonlinear systems:
- Pretty decent automation for systems with univariate nonlinearities.
- Heuristics for multi-variate systems.
Automated Analysis for nonlinear systems:
  ○ Pretty decent automation for systems with univariate nonlinearities.
  ○ Heuristics for multi-variate systems.
Heuristic loop invariant generation for control loops
● Automated Analysis for nonlinear systems:
  ○ Pretty decent automation for systems with univariate nonlinearities.
  ○ Heuristics for multi-variate systems.
● Heuristic loop invariant generation for control loops
● Taylor Approximations
● ...
● Automated Analysis for nonlinear systems:
  ○ Pretty decent automation for systems with univariate nonlinearities.
  ○ Heuristics for multi-variate systems.
● Heuristic loop invariant generation for control loops
● Taylor Approximations
● ...
● Component-based Verification Tooling
  Mueller et al., *Change and Delay Contracts for Hybrid System Component Verification*, FASE’17 -- Thursday 10:30-12:30
KeYmaera X is a hybrid systems theorem prover with:

- A small and trustworthy prover core and
- Excellent infrastructure for interactively verifying complex systems and implementing automated analyses.
KeYmaera X is a hybrid systems theorem prover with:

- A small and trustworthy prover core and
- Excellent infrastructure for **interactively verifying complex systems** and implementing automated analyses.

**Project Website (start here)**  keymaeraX.org

**Online Demo**  web.keymaeraX.org

**GPL’d Source Code**  github.com/ls-lab/KeYmaeraX-release

**Course Materials**  symbolaris.com/course/fcps17.html
Developers:

- Stefan Mitsch
- Nathan Fulton
- Andre Platzer
- Jan-David Quesel
- Brandon Bohrer
- Yong Kiam Tan
- Markus Voelp

Special Thanks:

- 15-424 students, Jean-Baptiste Jeanin, Khalil Ghorbal, Daniel Ricketts
Parachute Closed:
\[ J \land t=0 \land r=r_p \rightarrow \]
\[ [x'=v, v'=rv^2-g \land 0\leq x \land t\leq T] v> -\sqrt{g/ pr} > m \]

Proof requires a **differential ghost** because the property is **not inductive**.
Interactive Reachability Analysis in KeYmaera X

**Differential Ghosts**

An example differential ghost.

\[ x > 0 \rightarrow [x' = -x] x > 0 \]
An example differential ghost.

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

Ghost: \[ y' = y/2 \]
Conserved: \[ 1 = xy^2 \]
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:

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