Hybrid theorem proving as a lightweight method for verifying numerical software

Alper Altuntas [1]   John Baugh [2]

altuntas@ucar.edu, jwb@ncsu.edu

[1]National Center for Atmospheric Research, Boulder, CO

[2]North Carolina State University, Raleigh, NC

Correctness'18

November 12, 2018

Dallas, TX

NCAR | NC STATE UNIVERSITY

Numerical Software Verification

- The accuracy of physical models depend on:
  1. Convergence of numerical methods.
  2. Correctness of realization in software:
     - Complicated by intermittent discrete updates.
- This study presents:
  - A lightweight verification approach for (2).

Hybrid Theorem Proving

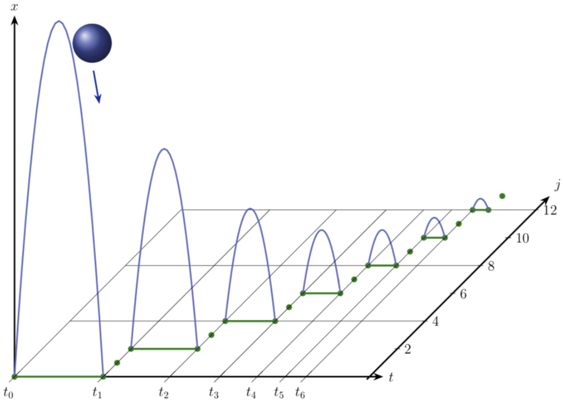
A formal verification technique for cyber-physical systems.

Hybrid Theorem Proving

- **Cyber-physical systems** are compositions of:
  1. Real world physics
     continuous evolution
  2. Computer sampling (sensors) and intervention (actuators)
     discrete changes
  - Examples: self-driving cars, ATC, robots, etc.
  - Verification tools: hybrid theorem provers, e.g., KeYmaera X
- **Hybrid programs** that model cyber-physical systems:
  1. ODEs model real world physics.
  2. Discrete programs model computer intervention.

Hybrid Theorem Proving

Continuous Evolution + Intermittent Discrete Changes



Platzer, A. "Logical Foundations of Cyber-Physical Systems." (2018)

- Continuous evolution:

$$\frac{\partial x}{\partial t} = v, \frac{\partial v}{\partial t} = -g$$

- Discrete change:

$$v := -v$$

Hybrid Theorem Proving

A hybrid model:

$(0 \leq x \leq x_{max}) \rightarrow$	*initial conditions*
$[\{$	*execution begins*
$\{x' = v, v' = -g\}$	*continuous evolution*
$v := -v;$	*discrete assignment*
$\}^*]$	*loop or terminate*
$(0 \leq x \leq x_{max})$	*postcondition*

where x is altitude, v is velocity, a is acceleration.

Hybrid Theorem Proving

A hybrid model:

	$(0 \leq x \leq x_{max}) \rightarrow$	*initial conditions*
	$[\{$	*execution begins*
over some Δt $\{$	$\{x' = v, v' = -g\}$	*continuous evolution*
instantaneous $\{$	$v := -v;$	*discrete assignment*
	$\}^*]$	*loop or terminate*
	$(0 \leq x \leq x_{max})$	*postcondition*

where x is altitude, v is velocity, a is acceleration.

Hybrid theorem proving for verifying numerical software

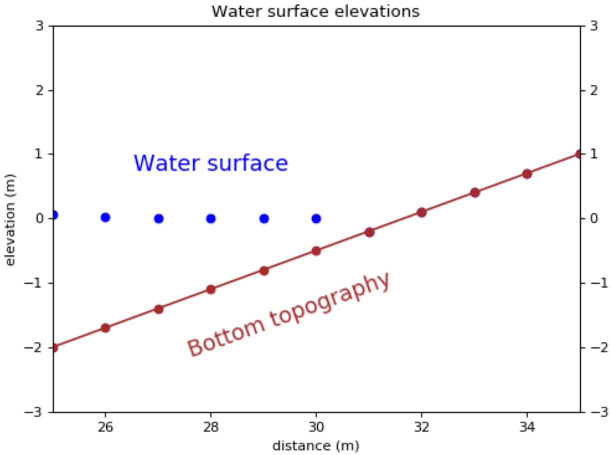
A lightweight formal methods approach

Hybrid theorem proving for verifying numerical software

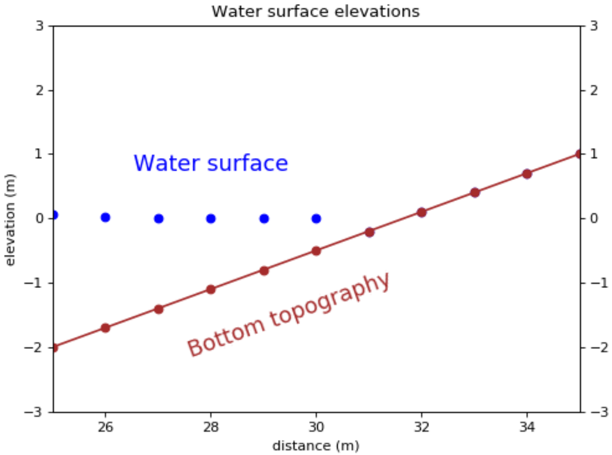
- Based on viewing numerical models as a hybrid system.
  1. **Continuous processes:** differential equations (DEs) solved by the model.
     e.g., evolution of water surface height
  2. **Discrete updates:** often arise from ad-hoc and empirical modeling.
     e.g., a location becoming wet/dry

- DEs are discretized in time and space, yet they may taken to be continuous.
  - to abstract away from numerical methods.
  - to focus on discrete decisions and updates.

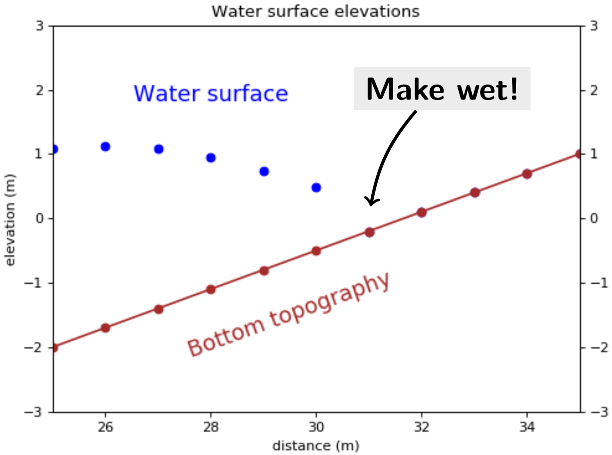
Hybrid theorem proving for verifying numerical software



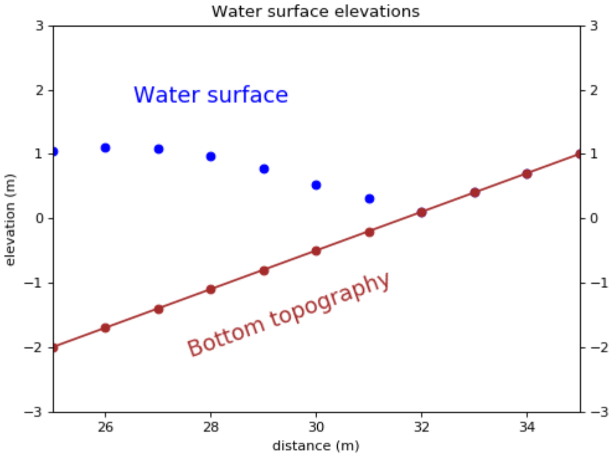
Hybrid theorem proving for verifying numerical software



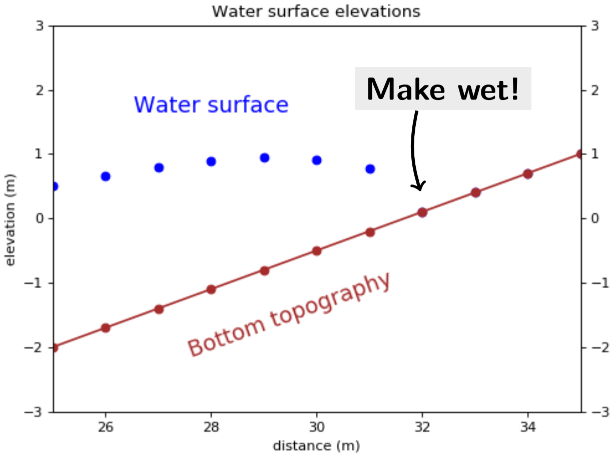
Hybrid theorem proving for verifying numerical software



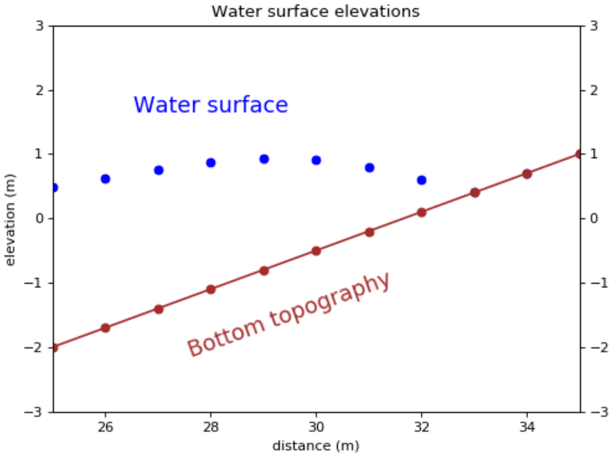
Hybrid theorem proving for verifying numerical software



Hybrid theorem proving for verifying numerical software



Hybrid theorem proving for verifying numerical software



Hybrid theorem proving for verifying numerical software

1-D shallow water equations:

$$\eta' = -\frac{\partial uh}{\partial x}, \ldots \tag{1}$$

- In an **actual numerical model**, discretize both in time and space:

$$\frac{\eta_i^{n+1} - \eta_i^n}{\Delta t} = \Big((uh)_{i+1}^n - (uh)_{i-1}^n\Big)/(\Delta x), \ldots \tag{2}$$

- In a **hybrid verification model**, discretize in space only:

$$\eta' = \Big((uh)_{i+1} - (uh)_{i-1}\Big)/(\Delta x), \ldots \tag{3}$$

where η is water elevation, h is water height, u is velocity.

Hybrid theorem proving for verifying numerical software

Abstract discrete grids:

- Small discrete grids for tractability.
- Non-determinism to represent external states.

Rationale: By the CFL condition, domain of dependence is limited.

Hybrid theorem proving for verifying numerical software

Hybrid model of the 1-D wetting and drying:

initialConditions() $\rightarrow$	*initial condition*
$[\{$	*execution begins*
$\{\ \eta' = (uh_{i+1} - uh_{i-1})/(\Delta x), ..\}$	*continuous evolution*
wettingDrying();	*discrete assignment*
$\}^*]$	*loop or terminate*
safetyCondition()	*postcondition*

where η is water elevation, h is water height, u is velocity.

Hybrid theorem proving for verifying numerical software

Key elements of the abstraction approach:

- View numerical software as a hybrid system.
- Model PDEs as continuous in time and discrete in space (as in the method of lines).
- Incorporate discrete updates.
- Work with small, discrete grids.

Test Case: The Keymaera X Model of the KPP scheme

Application of hybrid theorem proving in Earth System Modeling

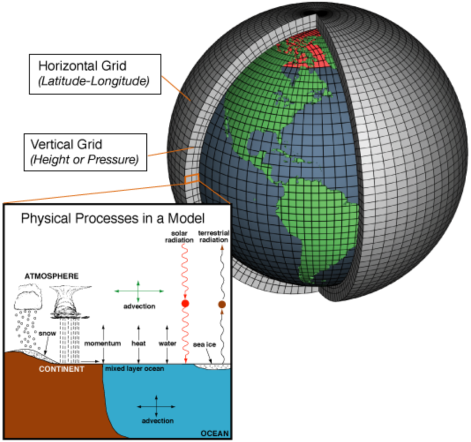
Test Case: The Keymaera X Model of the KPP scheme

- A test case involving a large-scale numerical software:
  - **CESM**: A leading climate model developed by NCAR.
  - **MOM6**: The future ocean component of CESM.
- **Main Project:** Coupling of MOM6 in CESM
  - The KPP scheme recently incorporated in MOM6.
  - An **unphysical behavior** when KPP matching is turned on!

Test Case: The Keymaera X Model of the KPP scheme

Earth System Models:

- HPC software that simulate Earth's climate.
- Components: atmosphere, ocean, ice, land, etc.
- Differential equations that model physical, chemical, and biological processes.
- Millions of core-hours!



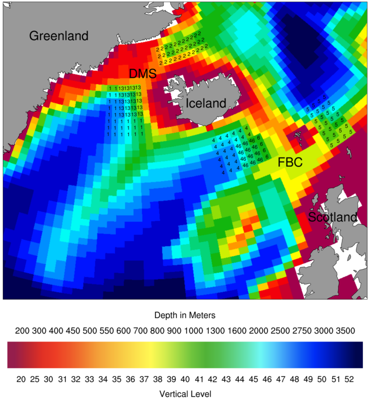
https://celebrating200years.noaa.gov/breakthroughs/climate_model/

Test Case: The Keymaera X Model of the KPP scheme

Global Ocean Models:

- The 3D primitive equations.
- Finite difference approximations.
- Subgrid-scale processes included as parameterizations. Example:
  - **KPP** scheme parameterizes ocean mixing due to vertical turbulent fluxes in the OBL.

Horizontal resolution of workhorse ocean grids are $\sim 1^\circ \times 1^\circ$



Briegleb et al. (2010, NCAR Tech. Note)

Test Case: The Keymaera X Model of the KPP scheme

The KPP scheme

- The continuous evolution of a scalar quantity λ over a vertical water column:

$$\frac{\partial\overline{\lambda}}{\partial t} = \frac{\partial}{\partial z}(\overline{w'\lambda'} + \overline{w}\,\overline{\lambda}) \tag{4}$$

- The unresolved turbulent flux parameterized as a diffusive process (Griffies et al., 2015):

$$\overline{w'\lambda'} = -K_\lambda(\frac{\partial\overline{\lambda}}{\partial z} + \gamma_\lambda) \tag{5}$$

- The diffusivity K_λ at depth d within the OBL (Large et al., 1994):

$$K_\lambda = h \cdot w_\lambda(\sigma) \cdot G_\lambda(\sigma) \tag{6}$$

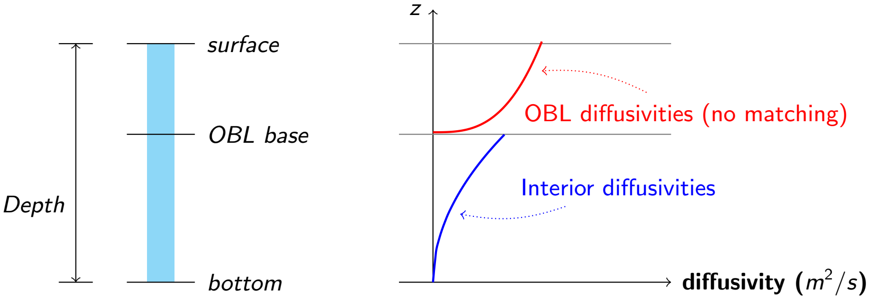
- Shape function for the OBL diffusivities:

$$G_\lambda(\sigma) = a_0 + a_1\sigma + a_2\sigma^2 + a_3\sigma^3 \tag{7}$$

Test Case: The Keymaera X Model of the KPP scheme

The KPP scheme

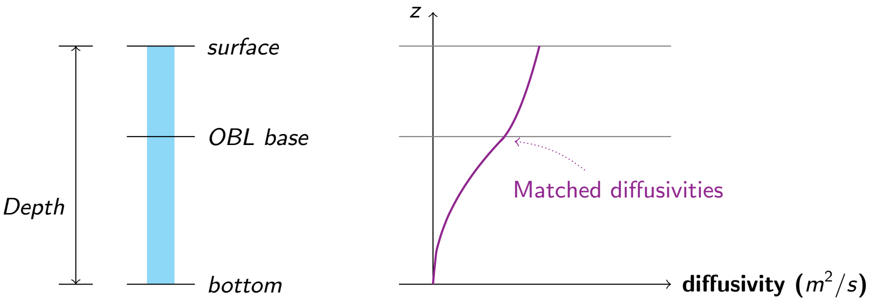
- Compute the ocean boundary layer (OBL) depth.
- Compute the diffusivities within ocean interior.
- Compute the OBL diffusivities (no matching).



Test Case: The Keymaera X Model of the KPP scheme

The KPP scheme

- Compute the ocean boundary layer (OBL) depth.
- Compute the diffusivities within ocean interior.
- Compute the OBL diffusivities **(matching)**.



Test Case: The Keymaera X Model of the KPP scheme

- **Undesired behavior:**
  - Matching algorithm leads to negative diffusivities in OBL.
- **Debugging:**
  - Via an interactive debugger.
  - Took several days and thousands of CPU hours.
- **Fix:**
  - Modify the matching algorithm for cases where the interior diffusivity gradient is negative.
- **Verification:**
  - A KeYmaera X model of the KPP scheme.

Test Case: The Keymaera X Model of the KPP scheme

The hybrid model of the KPP scheme:

$$
\begin{array}{lll}
 & \textit{initialConditions}() \rightarrow & \\
 & \quad [\{ & \\
\text{timestep} \left\{ \begin{array}{l} \\ \\ \\ \\ \end{array} \right. & \qquad \textit{compute\_OBL\_depth}; & // \textit{ discrete updates} \\
 & \qquad \textit{compute\_interior\_K}; & \\
 & \qquad \textit{compute\_OBL\_K}; & \\
 & \qquad \{z'_{cr} = -z_{cr}\} & // \textit{ continuous system} \\
 & \quad \}^*] & \\
 & K > 0 &
\end{array}
$$

where K is diffusivity and z_{cr} is the depth at which Ri_B is equal to Ri_{cr}.

Test Case: The Keymaera X Model of the KPP scheme

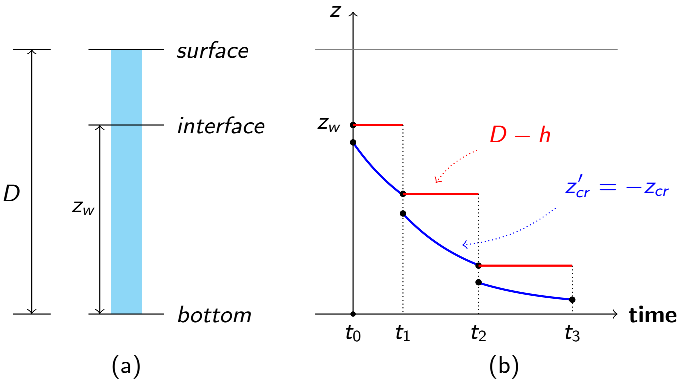
The hybrid model of the KPP scheme:

$$
\begin{array}{lll}
 & \mathit{initialConditions}() \rightarrow & \\
 & \quad [\{ & \\
\text{instantaneous} \left\{ \begin{array}{l} \\ \\ \\ \end{array} \right. & \quad\quad \mathit{compute\_OBL\_depth}; & // \text{ discrete updates} \\
 & \quad\quad \mathit{compute\_interior\_K}; & \\
 & \quad\quad \mathit{compute\_OBL\_K}; & \\
\text{over some } \Delta t \ \{ & \quad\quad \{z_{cr}' = -z_{cr}\} & // \text{ continuous system} \\
 & \quad \}^*] & \\
 & K > 0 &
\end{array}
$$

where K is diffusivity and z_{cr} is the depth at which Ri_B is equal to Ri_{cr}.

Test Case: The Keymaera X Model of the KPP scheme

Continuous evolution of z_{cr} + discrete changes:

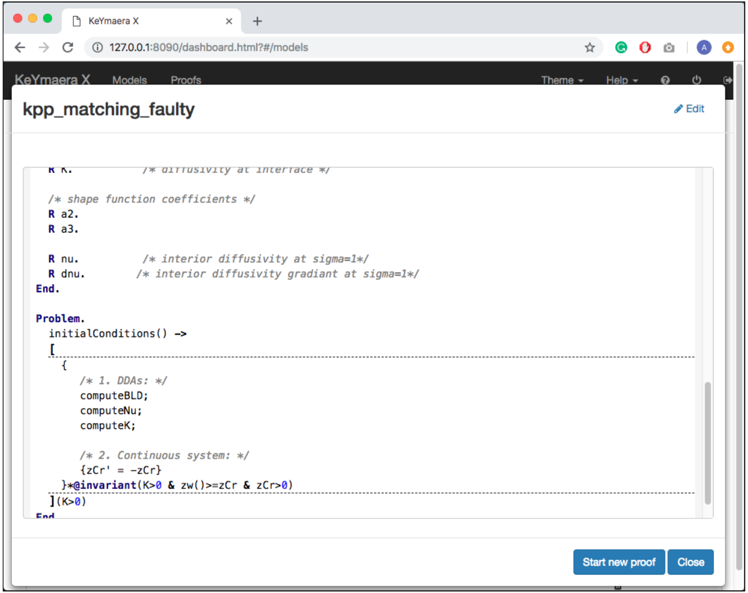


Test Case: The Keymaera X Model of the KPP scheme

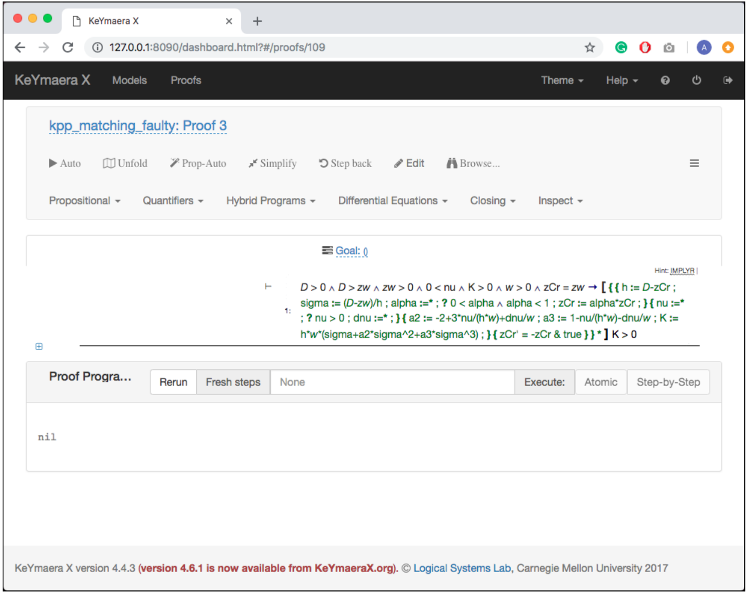
The KeYmaera X proof process:

1. Develop the hybrid model of the KPP scheme
2. Reproduce the undesired behavior via a custom proof tactic:
   - Conventional logical assertions and rules, e.g.,
     - loop invariants
   - as well as *differential dynamic logic* rules, e.g.,
     - differential invariants
3. Apply the fix in the matching algorithm: match gradients only if interiorgradients are non-negative.
4. Re-run the proof tactic and confirm that undesired behavior is eliminated.

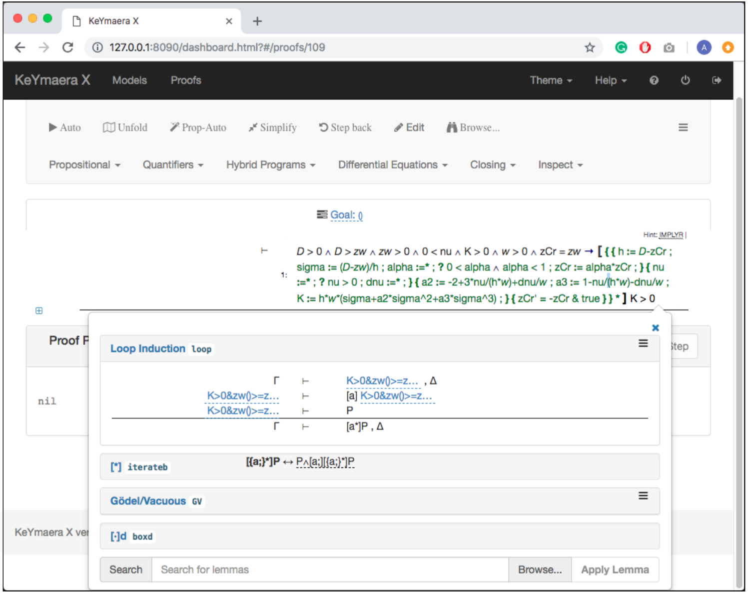
KeYmaera X UI:
Develop the hybrid model.



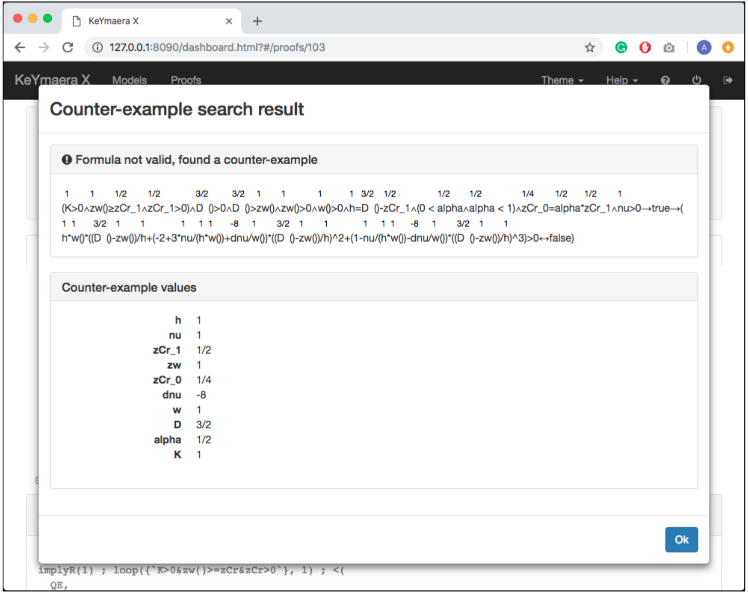
KeYmaera X UI:
Construct a proof.



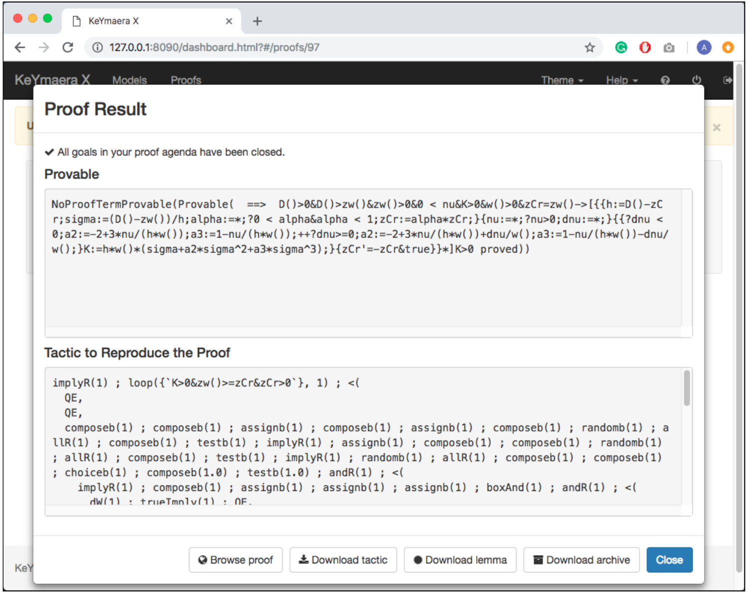
KeYmaera X UI:
Steer the proof.



KeYmaera X UI:
Generate counter-example if formula is invalid.



KeYmaera X UI:
Close the proof.



Conclusions

- A lightweight formal methods application.
- Highly efficient compared to testing.
- Provides more confidence.
  - Generality (due to nondeterminism)
  - The coverage in the temporal dimension is much greater.
- Limitations:
  - floating point arithmetic
  - numerical issues

Thanks

altuntas@ucar.edu, jwb@ncsu.edu