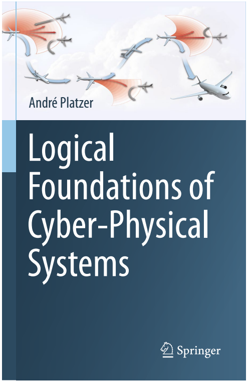
A Tale of Three Provers

Logical Foundations of Cyber-Physical Systems



Stefan Mitsch

Carnegie Mellon University
Computer Science Department

Outline

Outline

How to Put it All into Theorem Proving Practice?

Design decisions

- Axioms: what is the minimal set needed?
- Rules and derived rules
- Arithmetic procedures: use at all? how to interoperate reliably?
- Invariant generators: make non-soundness critical
- Proof management, replay, visualization

Proof repeatability challenges

- Timeouts
- External tools
- What is a robust proof?
  `auto` vs. `implyR(1)` vs. `implyR(R=="x>0->x>=0")`?

Prover Core

Term language

Should be minimal to make correctness obvious, but then makes tracking other information difficult (e.g., annotations, definitions, archives)

- Terms
- Formulas
- Programs

Proof data structure

- Conclusion and premises
- Operations on proofs?

Essential tools

- Parser: how to make not soundness-critical?
- Arithmetic tools: bridge data structure differences (numbers!)

Prover Infrastructure and User Experience

Infrastructure

- Unification
- Store proofs
- Suspend and resume proofs

Automation vs. interaction

- Fully interactive vs. fully automatic?
- “Auto-active” with annotations in the code?
- Interrupt automation + interactive?
- Tactic-based? Interpreted or hosted tactic language?
- Proof robustness?

Outline

KeYmaera 3

Build on top of an existing dynamic logic prover

Benefits

- Inherit user interface
- Inherit infrastructure (help resources, proof archives)

Implementation Reality: Non-deterministic Choice

```
box_choice_right {
  \find (==> \modality{#boxtr}#dl ++ #dl2\endmodality(post))
  \replacewith (==> \modality{#boxtr}#dl\endmodality(post));
  \replacewith (==> \modality{#boxtr}#dl2\endmodality(post))
  \heuristics (simplify_prog)
  \displayname "[++] choice"
};
```

KeYmaera 3

Successes

- Easy to add rules to proof search
- Simple proof management: single proof tree

Challenge: Limit applicability of proof rules

KeYmaera 3 proof rule mechanism adapts rules to *any* context

$$[\cup]\ \frac{\Gamma \vdash \mathscr{U}[\alpha]\phi, \Delta \quad \Gamma \vdash \mathscr{U}[\beta]\phi, \Delta}{\Gamma \vdash \mathscr{U}[\alpha \cup \beta]\phi, \Delta} \qquad ↯\ \frac{\Gamma \vdash \mathscr{U} J, \Delta \quad \Gamma, \mathscr{U} J \vdash \mathscr{U}\phi, \Delta \quad \Gamma, \mathscr{U} J \vdash \mathscr{U}[\alpha]J, \Delta}{\Gamma \vdash \mathscr{U}[\alpha^*]\phi, \Delta}$$

```
loop_inv_box_quan {
  \find (==> \modality{#boxtr}#dl*\endmodality(post))
  "Invariant Initially Valid":
    \replacewith (==> inv);
  "Use Case":
    \replacewith (==> #UnivCl(\[#dl\]true, inv->post, false));
  "Body Preserves Invariant":
    \replacewith (==> #UnivCl( ... ))
};
```

KeYmaera 3

Successes

- Easy to add rules to proof search
- Simple proof management: single proof tree

Challenge: Limit applicability of proof rules

KeYmaera 3 proof rule mechanism adapts rules to *any* context

$$[\cup]\ \frac{\Gamma \vdash \mathscr{U}[\alpha]\phi, \Delta \quad \Gamma \vdash \mathscr{U}[\beta]\phi, \Delta}{\Gamma \vdash \mathscr{U}[\alpha \cup \beta]\phi, \Delta} \qquad \lightning\ \frac{\Gamma \vdash \mathscr{U} J, \Delta \quad \Gamma, \mathscr{U} J \vdash \mathscr{U}\phi, \Delta \quad \Gamma, \mathscr{U} J \vdash \mathscr{U}[\alpha]J, \Delta}{\Gamma \vdash \mathscr{U}[\alpha^*]\phi, \Delta}$$

```
loop_inv_box_quan {
  \find (==> \modality{#boxtr}#dl*\endmodality(post))
  "Invariant Initially Valid":
    \replacewith (==> inv);
  "Use Case":
    \replacewith (==> #UnivCl(\[#dl\]true, inv->post, false));
  "Body Preserves Invariant":
    \replacewith (==> #UnivCl( ... ))
};
```

KeYmaera 3

Successes

- Easy to add rules to proof search
- Simple proof management: single proof tree

Challenge: Limit applicability of proof rules

KeYma…

$[\cup]\ \frac{\Gamma \vdash \mathscr{U} \ldots \alpha]J, \Delta}{}$

Challenge

Meta-operator `#UnivCl` soundness-critical (800LoC)
Similar meta-operators needed to compute Lie derivatives etc.

```
loop_inv_box_quan {
  \find (==> \modality{#boxtr}#dl*\endmodality(post))
  "Invariant Initially Valid":
    \replacewith (==> inv);
  "Use Case":
    \replacewith (==> #UnivCl(\[#dl\]true, inv->post, false));
  "Body Preserves Invariant":
    \replacewith (==> #UnivCl( ... ))
};
```

Outline

KeYmaeraD

Bare-bones prover with direct dL rendition in Scala

Benefits

- And/Or-proof tree
- Pattern matching in Scala replaces rule application mechanism
- Directly program proofs in Scala

Implementation Reality: Non-deterministic Choice

```
val choose = new ProofRule("choose") {
  def apply(p: Position) = sq => {
    lookup(p, sq) match {
      case Modality(Box, Choose(h1, h2), phi) =>
        val fm1 = Modality(Box, h1, phi)
        val fm2 = Modality(Box, h2, phi)
        Some((List(replace(p, sq, Binop(And, fm1, fm2))),Nil))
      case _ => None
    }
  }
```

KeYmaeraD

Successes

- Parallel proof search
- Low-level control over proof

Challenges

- Simple rules put burden on user, for example assignment

$$[:=]\text{eq} \quad \frac{\Gamma, y = e \vdash P_x^y, \Delta}{\Gamma \vdash [x := e]P, \Delta} \quad (y \text{ fresh})$$

$$x = y \vdash [x := x + 1][x := 2x][x := x - 2]x = 2y$$

  produces $x = y, x_0 = x + 1, x_1 = 2x_0, x_2 = x_1 - 2 \vdash x_2 = 2y$
- More clever rules require considerable admissibility-checking
- Every extension of proof rules is soundness-critical

Outline

KeYmaera X

Small-core prover

Benefits

- Proof search decoupled from proof checking
- Rule application in context

Implementation Reality: Axioms

```
Axiom "[++] choice"
  [a;++b;]p(||) <-> [a;]p(||) & [b;]p(||)
End.
Axiom "[:=] assign"
  [x:=f();]p(x) <-> p(f())
End.
Axiom "[:=] assign equality"
  [x:=f();]p(||) <-> \forall x (x=f() -> p(||))
End.
```

KeYmaera X Tactics

Assignment

```
@Tactic("[:=]", conclusion = "__[x:=e]p(x)__<->p(e)")
val assignb = anon by { w =>
    useAt("[:=] assign")(w)
  | useAt("[:=] self assign")(w)
  | asgnEq(w)
}
```

Interpreted vs. builtin tactics

- Tactic `assignb` is implemented as an interpreted tactic
- Benefit: can record inner steps
- Challenge: performance
- Challenge: debugging
- Challenge: result depends on proof state

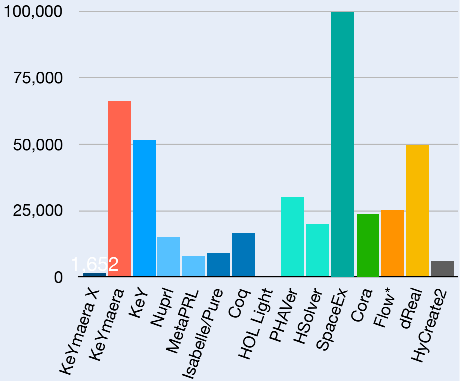
Equational assignment

```
@Tactic(
  names = "[:=]=",
  premises =   "G, x=e |- P, D",
  //     [:=]=  ------------------
  conclusion = "G |- [x:=e]P, D",
  displayLevel = "all"
)
val asgnEq = anon by ((w, seq) => seq.sub(w) match {
  case Some(Box(Assign(x, t), p)) =>
    val y = freshNamedSymbol(x, seq)
    boundRenaming(x, y)(w) &
    useAt("[:=] assign equality")(w) &
    uniformRenaming(y, x) &
    (if (w.isTopLevel&&w.isSucc) allR(w) & implyR(w) else ident)
  })
```

KeYmaera X

Successes

- Core size
- Extensibility with tactics
- Modularization
- Extension with non-trusted generators



Challenges

- Performance
- Handover between automation and interaction

Outline

Reasoning Styles

Assignments

$$x=1 \vdash \{x:=x+1\}[?x\geq 1]x\geq 0 \quad \textbf{K3 (update)}$$

$$x=1, x_1=x+1 \vdash [?x_1\geq 1]x_1\geq 0 \quad \textbf{KD (equation)}$$

$$\cfrac{x=1 \vdash [?x+1\geq 1]2\geq 0 \quad \textbf{KX (substitution)}}{x=1 \vdash [x:=x+1][?x\geq 1]x\geq 0}\,[:=]$$

The surprising subtlety of assignments

$$[:=]\cfrac{\vdash [x:=3]x\geq 3}{\vdash [x:=2][x:=3]x\geq 3}$$

$$[:=]\cfrac{\vdash [x:=2+1]x\geq 3}{\vdash [x:=2][x:=x+1]x\geq 3}$$

$$[:=]\cfrac{\vdash [\{x'=-x^2\}](x+1)^2\geq 2}{\vdash [\{x'=-x^2\}][x:=x+1]x^2\geq 2}$$

$$[:=]\cfrac{x_0=1, x=2 \vdash [\{x'=x\}]x\geq 2}{x=1 \vdash [x:=2][\{x'=x\}]x\geq 2}$$

$$[:=]\cfrac{\vdash [x:=2+1 \cup x:=3][(x:=x+1)^*]x\geq 3}{\vdash [x:=2][x:=x+1 \cup x:=3][(x:=x+1)^*]x\geq 3}$$

$$[:=]\cfrac{\vdash [(x:=x+1)^*]\forall x\,(x=2 \rightarrow [\{x'=3\}]x\geq 2)}{\vdash [(x:=x+1)^*][x:=2][\{x'=3\}]x\geq 2}$$

Reasoning Styles

Loops

K3	$x = 1, b > 0 \vdash x \geq 1$	**K3**	$x = 1, b > 0 \vdash \forall x\,(x \geq 1 \rightarrow x \geq 0)$
KD	$x = 1, b > 0 \vdash x \geq 1 \land b > 0$	**KD**	$x \geq 1 \land b > 0 \vdash x \geq 0$
KX	$x = 1, b > 0 \vdash x \geq 1$	**KX**	$b > 0, x \geq 1 \vdash x \geq 0$

(a) Base case (b) Use case

K3 $x = 1, b > 0 \vdash \forall x\,(x \geq 1 \rightarrow [x := x + 1/b]x \geq 1)$

KD $x \geq 1 \land b > 0 \vdash [x := x + 1/b](x \geq 1 \land b > 0)$

KX $b > 0, x \geq 1 \vdash [x := x + 1/b]x \geq 1$

(c) Induction step

$$\text{loop}\ \frac{\text{Base case (1a)} \qquad \text{Use case (1b)} \qquad \text{Induction step (1c)}}{x = 1, b > 0 \vdash [(x := x + 1/b)^*]x \geq 0}$$

Differential Induction

$$\dfrac{\begin{array}{llr} & \text{(init)} & \text{(step)} \\ \textbf{K3} & x^2+y^2=1 \vdash x^2+y^2=1 & x^2+y^2=1 \vdash \forall x \forall y\,(2xy+2y(-x)=0) \\ \textbf{KD} & x^2+y^2=1 \vdash x^2+y^2=1 & \vdash 2xy+2y(-x)=0 \\ \textbf{KX} & x^2+y^2=1 \vdash x^2+y^2=1 \checkmark & x_0^2+y_0^2=1 \vdash [x':=y][y':=-x]2xx'+2yy'=0 \checkmark \end{array}}{x^2+y^2=1 \vdash [x'=y, y'=-x]x^2+y^2=1}\,\text{DI}$$

- KeYmaera X provides several versions of differential induction (fully automatic to fully manual) to accommodate examples of varying complexity
- Fully manual differential induction is challenging in axiomatic prover, easy in rule-based prover

Proof Management: Central Proof Tree vs. Separate Steps

$$
\begin{array}{ll}
 & * \\
\mathbb{R} & \hline \text{\faEyeSlash}\, x < 0 \lor x = 0, y = x \vdash xy \le y^2 \\
\end{array}
\qquad
\begin{array}{ll}
 & * \\
\mathbb{R} & \hline x < 0 \lor x = 0, y > 0 \vdash xy \le y^2 \\
\text{WLi} & \hline \text{\faEyeSlash}\, x < 0 \lor x = 0, y > 0 \vdash xy \le y^2 \\
\end{array}
$$

$$
\begin{array}{ll}
\lor\text{L} & \hline \text{\faEyeSlash}\, x < 0 \lor x = 0, y = x \lor y > 0 \vdash xy \le y^2 \\
\text{WL} & \hline x < 0 \lor x = 0, y = x \lor y > 0 \vdash xy \le y^2
\end{array}
$$

(a) Wanted effect of temporarily hiding a formula: ignore for some proof steps, re-introduce when needed

$$
\begin{array}{ll}
 & * \\
\text{id} & \hline x = 0, x = 1 \vdash x = 0 \\
\text{WLi} & \hline \text{\faEyeSlash}\, x = 0, x = 1 \vdash x = 0 \\
\lightning & \hline \text{\faEyeSlash}\, x = 0 \vdash [x := 1]x = 0 \\
\text{WL} & \hline x = 0 \vdash [x := 1]x = 0
\end{array}
$$

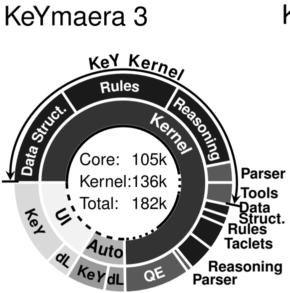
(b) Unsoundly ignoring temporarily hidden formula

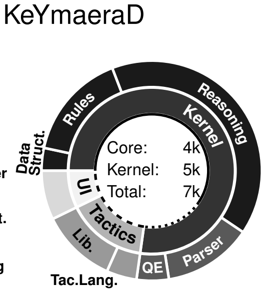
$$
\begin{array}{ll}
 & x_0 = 0, x = 1 \vdash x = 0 \\
\circlearrowleft\text{US} & \hline P_{\text{\faEyeSlash}}(x_0), x = 1 \vdash x = 0 \\
[:=] & \hline P_{\text{\faEyeSlash}}(x) \vdash [x := 1]x = 0 \\
\circlearrowleft\text{US} & \hline x = 0 \vdash [x := 1]x = 0
\end{array}
$$

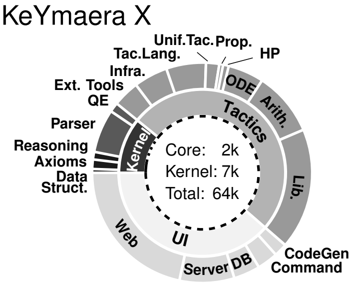
(c) Sub-proof with substitution $\sigma = \{P_{\text{\faEyeSlash}}(\cdot) \mapsto \cdot = 0\}$

Code Statistics

- Soundness-critical core
- Non-critical prover code
- Correctness-critical tools
- Non-critical user-facing infrastructure



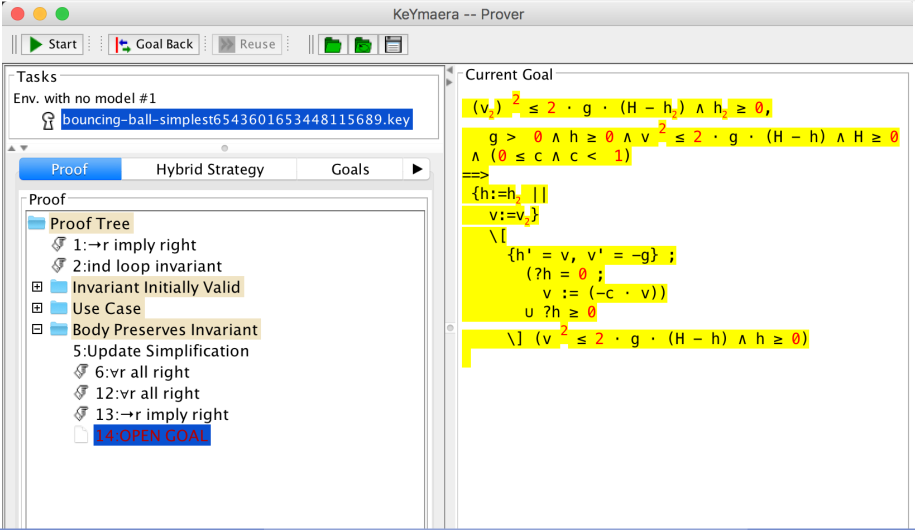




Outline

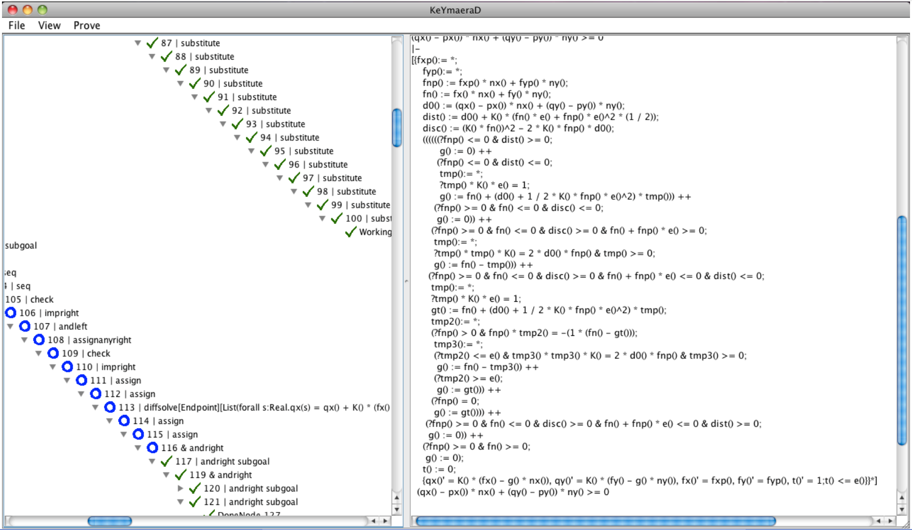
Challenges

- Easy to lose track of remaining proof effort
- Visualize large trees



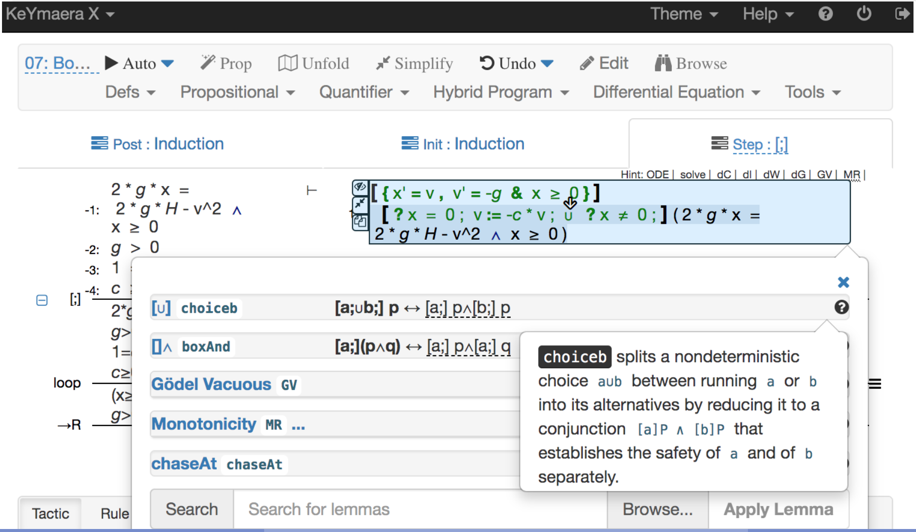
Challenges

- Easy to lose track of remaining proof effort
- Visualize large trees



Challenges

- Easy to lose track of remaining proof effort
- Visualize large trees



Outline

Summary

Lessons Learned

- Small core makes it significantly easier to advance development
- Uniform substitution calculus increases flexibility in proofs
- Common proof state data structure helps interleaving automation and interaction

Advice for Future Development

- Invest in common infrastructure early
- Invest in the ability of visualizing/debugging tactics
- Strive for deterministic tactic behavior (timeouts in automation are challenging)
- Invest in good error messages

Stefan Mitsch and André Platzer.
A retrospective on developing hybrid system provers in the KeYmaera family - A tale of three provers.
In *Deductive Software Verification: Future Perspectives - Reflections on the Occasion of 20 Years of KeY*, pages 21–64. 2020.
doi:10.1007/978-3-030-64354-6_2.

Stefan Mitsch.
Implicit and explicit proof management in KeYmaera X.
In *Proceedings of the 6th Workshop on Formal Integrated Development Environment, F-IDE@NFM 2021*, pages 53–67, 2021.
doi:10.4204/EPTCS.338.8.

Stefan Mitsch and André Platzer.
The KeYmaera X proof IDE - concepts on usability in hybrid systems theorem proving.
In *Proceedings of the Third Workshop on Formal Integrated Development Environment, F-IDE@FM 2016, Limassol, Cyprus, November 8, 2016*, pages 67–81, 2016.
doi:10.4204/EPTCS.240.5.