A framework for proof certificates in finite state exploration

PxTP 2015

Quentin Heath Dale Miller

Inria Saclay–Île-de-France

LIX, École polytechnique

August 7, 2015

Proof Certificates

Computational logic systems should output *formal proofs* for independent checking.

Proof structures vary greatly. These choices are not just a matter of taste. There are important trade-offs between

- simplicity and complexity of checkers
- implicit and explicit proofs
- proof size and checking time

We do not focus on proof search: one machine (client) generates a proof, and another machine (kernel) checks it.

ProofCert

A multi-year project where we are developing the **Foundational Proof Certificate** (FPC) framework.

We aim to formally define the semantics of a wide range of *proof evidences*.

- such formal semantics can be executed to yield checkers
- specific checkers can be build by anyone from these definitions

Analogous frameworks exist:

- context-free grammars (CFG) define programming language structures
- structural operational semantics (SOS) define programming language execution

ProofCert, stage 1

We have been successful at developing the FPC framework for first-order classical and intuitionistic logics.

Classical logic	resolution, expansion trees, decision procedures such as CNF, truth tables
Intuitionistic logic	dependently typed λ-terms, G3ip
others	Also: rewriting, Frege-style proofs

Focused proof systems for classical and intuitionistic logics provide the theoretical justification for the design of FPCs.

The λProlog programming language (implemented via the Teyjus compiler) serves as a natural *prototyping system* for executing formal semantic definitions.

ProofCert, stage 2

Continue the work for proof evidence that may contain induction and co-induction.

We will then be able to treat model checking. In particular, we discuss here:

- reachability (existence of a path)
- non-reachability
- simulation, bisimulation, winning strategy
- non-simulation, non-bisimulation

We need a focused proof system for a logic with induction and co-induction.

Outline

1. The μF logic
   - Formulae
   - Focused system
   - Restricted formulae
   - Augmented system
2. Examples
   - Clerks & Experts
   - Certificates
3. Implementation

Outline

1. The μF logic
   - Formulae
   - Focused system
   - Restricted formulae
   - Augmented system
2. Examples
3. Implementation

Fixed points in linear logic

Surprisingly, neither

- intuitionistic nor classical logics nor
- full linear logic (with Girard's ! and ?)

are the right starting point for us here.

We rely on μMALL[Baelde, PhD, ToCL 2012] instead: this is MALL (*multiplicative additive linear logic*) plus

- the least (μ) and greatest (ν) fixed points operators
- first-order quantifiers $\forall$, $\exists$
- term equality

All three of these are treated as *logical connectives*.

μMALL formulae

To be more "user friendly", we

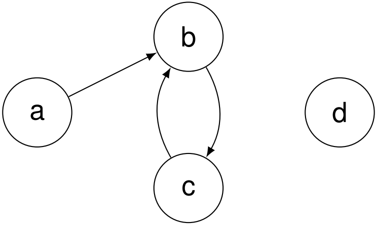
- drop the linear logic connectives for more conventional looking symbols
- use two sided sequents

There are two sets of connectives (following focusing polarity).

Negative connectives: f^-, $\supset$, t^-, $\wedge^-$, $\forall$, $\neq$ and ν,
Positive connectives: t^+, $\wedge^+$, f^+, $\vee$, $\exists$, $=$ and μ.

The negation of B is written as $B \supset f^-$.

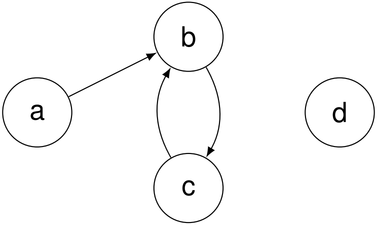
Example: graph




```
step a b. step b c. step c b.

path X Y :- step X Y.
path X Z :- exists Y (step X Y, path Y Z).
```

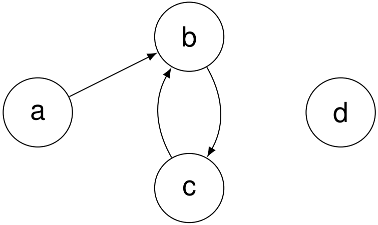
Example: graph



The `step` relation becomes a binary predicate $\cdot \longrightarrow \cdot$ defined by

$$\mu(\lambda A \lambda x \lambda y. ((x = a) \wedge^{+} (y = b)) \vee ((x = b) \wedge^{+} (y = c)) \\ \vee((x = c) \wedge^{+} (y = b)))$$

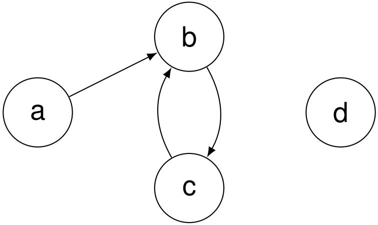
Example: graph



Similarly, the `path` relation becomes the binary predicate

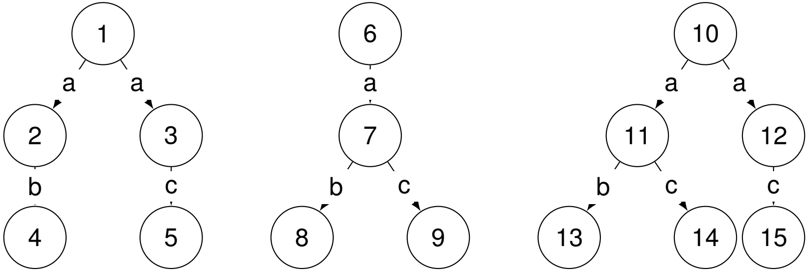
$$\mu\Big(\lambda P \lambda x \lambda z.\, x \longrightarrow z \vee (\exists y.\, x \longrightarrow y \wedge^{+} P\, y\, z)\Big)$$

Example: graph

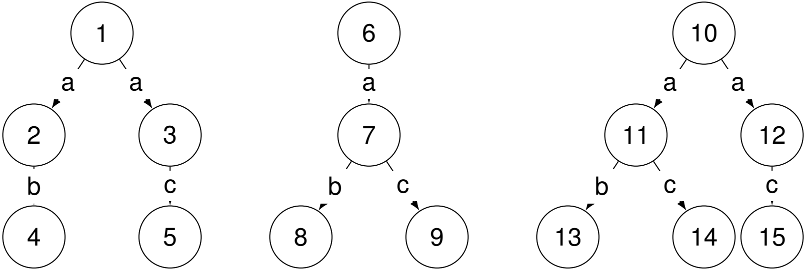


- we need only positive connectives to translate **Horn clauses**!

Example: labeled transition systems



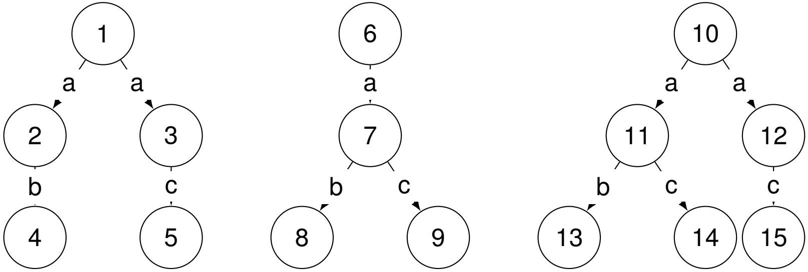
Example: labeled transition systems



For step, an LTS needs the ternary predicate $\cdot \xrightarrow{\cdot} \cdot$ defined by

$$\mu\left(\lambda A \lambda p \lambda a \lambda q. \bigvee_i ((p = u_i) \wedge^+ (a = v_i) \wedge^+ (q = w_i))\right)$$

Example: labeled transition systems

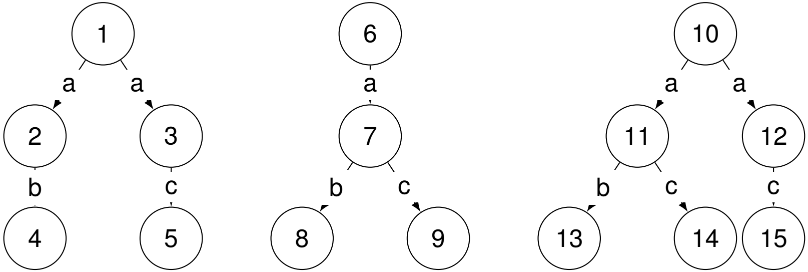


Finally, **Simulation** and **bisimulation** can be defined by

$$\nu\Big(\lambda S \lambda p \lambda q.\, \forall a \forall p'.\, p \xrightarrow{a} p' \supset \exists q'.\, q \xrightarrow{a} q' \wedge^{+} S\, p'\, q'\Big) \qquad (sim)$$

$$\begin{aligned}\nu\Big(\lambda B \lambda p \lambda q.\, &(\forall a \forall p'.\, p \xrightarrow{a} p' \supset \exists q'.\, q \xrightarrow{a} q' \wedge^{+} B\, p'\, q') \\ &\wedge^{-} (\forall a \forall q'.\, q \xrightarrow{a} q' \supset \exists p'.\, p \xrightarrow{a} p' \wedge^{+} B\, q'\, p')\Big)\end{aligned} \qquad (bisim)$$

Example: labeled transition systems



- these are not purely positive, but they are bipoles
- *bisim* contains both $\wedge^-$ and $\wedge^+$

A two-sided version of μMALLF

Let $\mathcal{N}$ and $\mathcal{P}$ denote, respectively, lists of negative and positive formulae.

Let Γ and Δ denote multisets of formulae.

Introduction rules are applied to formulae in the zones between occurrences of $\Uparrow$ / $\Downarrow$ and $\vdash$.

$\mathcal{N} \Uparrow \Gamma \vdash \Delta \Uparrow \mathcal{P}$	similar to the one-sided sequent $\vdash \mathcal{N}^\perp, \mathcal{P} \Uparrow \Gamma^\perp, \Delta$
$\Downarrow A \vdash$	*left-focused*, similar to $\vdash \Downarrow A^\perp$
$\vdash A \Downarrow$	*right-focused*, similar to $\vdash \Downarrow A$

Introduction of negative connectives

$$\frac{\mathcal{N}\theta \Uparrow \Gamma\theta \vdash \Delta\theta \Uparrow}{\mathcal{N} \Uparrow s = t, \Gamma \vdash \Delta \Uparrow}\,\dagger \qquad \frac{\mathcal{N}\theta \Uparrow \;\vdash\; \Uparrow}{\mathcal{N} \Uparrow \;\vdash s \neq t \Uparrow}\,\dagger \qquad \dagger\theta = mgu(s,t)$$

$$\frac{\mathcal{N} \Uparrow \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow t^{+}, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow \;\vdash\; \Uparrow}{\mathcal{N} \Uparrow \;\vdash f^{-} \Uparrow}$$

$$\frac{\mathcal{N} \Uparrow A_1, A_2, \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow A_1 \wedge^{+} A_2, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow A_1 \vdash A_2 \Uparrow}{\mathcal{N} \Uparrow \;\vdash A_1 \supset A_2 \Uparrow}$$

$$\frac{}{\mathcal{N} \Uparrow s = t, \Gamma \vdash \Delta \Uparrow}\,\ddagger \qquad \frac{}{\mathcal{N} \Uparrow \;\vdash s \neq t \Uparrow}\,\ddagger \qquad \ddagger s \text{ and } t \text{ not unifiable}$$

$$\frac{}{\mathcal{N} \Uparrow f^{+}, \Gamma \vdash \Delta \Uparrow} \qquad \frac{}{\mathcal{N} \Uparrow \;\vdash t^{-} \Uparrow}$$

$$\frac{\mathcal{N} \Uparrow A_1, \Gamma \vdash \Delta \Uparrow \quad \mathcal{N} \Uparrow A_2, \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow A_1 \vee A_2, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow \;\vdash A_1 \Uparrow \quad \mathcal{N} \Uparrow \;\vdash A_2 \Uparrow}{\mathcal{N} \Uparrow \;\vdash A_1 \wedge^{-} A_2 \Uparrow}$$

$$\frac{\mathcal{N} \Uparrow C\,y, \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow \exists x.\, C\,x, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow \;\vdash C\,y \Uparrow}{\mathcal{N} \Uparrow \;\vdash \forall x.\, C\,x \Uparrow}$$

Introduction of positive connectives & structural rules

$$\frac{}{\Downarrow t \neq t \vdash} \qquad \frac{}{\vdash t = t \Downarrow} \qquad \frac{}{\Downarrow f^- \vdash} \qquad \frac{}{\vdash t^+ \Downarrow}$$

$$\frac{\vdash A_1 \Downarrow \qquad \Downarrow A_2 \vdash}{\Downarrow A_1 \supset A_2 \vdash} \qquad \frac{\vdash A_1 \Downarrow \qquad \vdash A_2 \Downarrow}{\vdash A_1 \wedge^+ A_2 \Downarrow}$$

$$\frac{\Downarrow A_i \vdash}{\Downarrow A_1 \wedge^- A_2 \vdash} \qquad \frac{\vdash A_i \Downarrow}{\vdash A_1 \vee A_2 \Downarrow}$$

$$\frac{\Downarrow C\, t \vdash}{\Downarrow \forall x.\, C\, x \vdash} \qquad \frac{\vdash C\, t \Downarrow}{\vdash \exists x.\, C\, x \Downarrow}$$

$$\frac{N \Uparrow \Gamma \vdash \Delta \Uparrow}{\Uparrow N, \Gamma \vdash \Delta \Uparrow}\ Store_L \qquad \frac{\Downarrow N \vdash}{N \Uparrow\ \vdash\ \Uparrow}\ Decide_L \qquad \frac{\Uparrow P \vdash\ \Uparrow}{\Downarrow P \vdash}\ Release_L$$

$$\frac{\Uparrow\ \vdash\ \Uparrow P}{\Uparrow\ \vdash P \Uparrow}\ Store_R \qquad \frac{\vdash P \Downarrow}{\Uparrow\ \vdash\ \Uparrow P}\ Decide_R \qquad \frac{\Uparrow\ \vdash N \Uparrow}{\vdash N \Downarrow}\ Release_R$$

Fixed-point rules: induction, coinduction, unfolding

$$\frac{\Uparrow B\, S\, \bar{y} \vdash S\, \bar{y} \Uparrow \qquad \mathcal{N} \Uparrow S\, \bar{t}, \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow \mu B\, \bar{t}, \Gamma \vdash \Delta \Uparrow}$$

$$\frac{\mathcal{N} \Uparrow \; \vdash S\, \bar{t} \Uparrow \qquad \Uparrow S\, \bar{y} \vdash B\, S\, \bar{y} \Uparrow}{\mathcal{N} \Uparrow \; \vdash \nu B\, \bar{t} \Uparrow}$$

$$\frac{\mathcal{N} \Uparrow B(\mu B)\bar{t}, \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow \mu B\, \bar{t}, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow \; \vdash B(\nu B)\bar{t} \Uparrow}{\mathcal{N} \Uparrow \; \vdash \nu B\, \bar{t} \Uparrow}$$

$$\frac{\Downarrow B(\nu B)\bar{t} \vdash}{\Downarrow \nu B\, \bar{t} \vdash} \qquad \frac{\vdash B(\mu B)\bar{t} \Downarrow}{\vdash \mu B\, \bar{t} \Downarrow}$$

The resulting proof system has no initial and no cut-rules.

Cut and initial are needed for richer aspects of model checking, but not immediately in this talk.

Branching negative connectives

$$\frac{\mathcal{N}\theta \Uparrow \Gamma\theta \vdash \Delta\theta \Uparrow}{\mathcal{N} \Uparrow s = t, \Gamma \vdash \Delta \Uparrow}\,\dagger \qquad \frac{\mathcal{N}\theta \Uparrow\ \vdash\ \Uparrow}{\mathcal{N} \Uparrow\ \vdash s \neq t \Uparrow}\,\dagger \qquad \dagger\theta = mgu(s,t)$$

$$\frac{\mathcal{N} \Uparrow \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow t^+, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow\ \vdash\ \Uparrow}{\mathcal{N} \Uparrow\ \vdash f^- \Uparrow}$$

$$\frac{\mathcal{N} \Uparrow A_1, A_2, \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow A_1 \wedge^+ A_2, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow A_1 \vdash A_2 \Uparrow}{\mathcal{N} \Uparrow\ \vdash A_1 \supset A_2 \Uparrow}$$

$$\frac{}{\mathcal{N} \Uparrow s = t, \Gamma \vdash \Delta \Uparrow}\,\ddagger \qquad \frac{}{\mathcal{N} \Uparrow\ \vdash s \neq t \Uparrow}\,\ddagger \qquad \ddagger s \text{ and } t \text{ not unifiable}$$

$$\frac{}{\mathcal{N} \Uparrow f^+, \Gamma \vdash \Delta \Uparrow} \qquad \frac{}{\mathcal{N} \Uparrow\ \vdash t^- \Uparrow}$$

$$\frac{\mathcal{N} \Uparrow A_1, \Gamma \vdash \Delta \Uparrow \quad \mathcal{N} \Uparrow A_2, \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow A_1 \vee A_2, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow\ \vdash A_1 \Uparrow \quad \mathcal{N} \Uparrow\ \vdash A_2 \Uparrow}{\mathcal{N} \Uparrow\ \vdash A_1 \wedge^- A_2 \Uparrow}$$

$$\frac{\mathcal{N} \Uparrow C\,y, \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow \exists x.\, C\,x, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow\ \vdash C\,y \Uparrow}{\mathcal{N} \Uparrow\ \vdash \forall x.\, C\,x \Uparrow}$$

Branching negative connectives

$$\frac{\mathcal{N}\theta \Uparrow \Gamma\theta \vdash \Delta\theta \Uparrow}{\mathcal{N} \Uparrow s = t, \Gamma \vdash \Delta \Uparrow} \dagger \qquad \frac{\mathcal{N}\theta \Uparrow \ \vdash \ \Uparrow}{\mathcal{N} \Uparrow \ \vdash s \neq t \Uparrow} \dagger \qquad \dagger\theta = mgu(s,t)$$

$$\frac{\mathcal{N} \Uparrow \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow t^+, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow \ \vdash \ \Uparrow}{\mathcal{N} \Uparrow \ \vdash f^- \Uparrow}$$

$$\frac{\mathcal{N} \Uparrow A_1, A_2, \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow A_1 \wedge^+ A_2, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow A_1 \vdash A_2 \Uparrow}{\mathcal{N} \Uparrow \ \vdash A_1 \supset A_2 \Uparrow}$$

$$\frac{}{\mathcal{N} \Uparrow s = t, \Gamma \vdash \Delta \Uparrow} \ddagger \qquad \frac{}{\mathcal{N} \Uparrow \ \vdash s \neq t \Uparrow} \ddagger \qquad \ddagger s \text{ and } t \text{ not unifiable}$$

$$\frac{}{\mathcal{N} \Uparrow f^+, \Gamma \vdash \Delta \Uparrow} \qquad \frac{}{\mathcal{N} \Uparrow \ \vdash t^- \Uparrow}$$

$$\frac{\mathcal{N} \Uparrow A_1, \Gamma \vdash \Delta \Uparrow \quad \mathcal{N} \Uparrow A_2, \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow A_1 \vee A_2, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow \ \vdash A_1 \Uparrow \quad \mathcal{N} \Uparrow \ \vdash A_2 \Uparrow}{\mathcal{N} \Uparrow \ \vdash A_1 \wedge^- A_2 \Uparrow}$$

$$\frac{\mathcal{N} \Uparrow C\,y, \Gamma \vdash \Delta \Uparrow}{\mathcal{N} \Uparrow \exists x.\, C\,x, \Gamma \vdash \Delta \Uparrow} \qquad \frac{\mathcal{N} \Uparrow \ \vdash C\,y \Uparrow}{\mathcal{N} \Uparrow \ \vdash \forall x.\, C\,x \Uparrow}$$

Branching structural rules

$$\frac{}{\Downarrow t \neq t \vdash} \qquad \frac{}{\vdash t = t \Downarrow} \qquad \frac{}{\Downarrow f^- \vdash} \qquad \frac{}{\vdash t^+ \Downarrow}$$

$$\frac{\vdash A_1 \Downarrow \quad \Downarrow A_2 \vdash}{\Downarrow A_1 \supset A_2 \vdash} \qquad \frac{\vdash A_1 \Downarrow \quad \vdash A_2 \Downarrow}{\vdash A_1 \wedge^+ A_2 \Downarrow}$$

$$\frac{\Downarrow A_i \vdash}{\Downarrow A_1 \wedge^- A_2 \vdash} \qquad \frac{\vdash A_i \Downarrow}{\vdash A_1 \vee A_2 \Downarrow}$$

$$\frac{\Downarrow C\, t \vdash}{\Downarrow \forall x.\, C\, x \vdash} \qquad \frac{\vdash C\, t \Downarrow}{\vdash \exists x.\, C\, x \Downarrow}$$

$$\frac{N \Uparrow \Gamma \vdash \Delta \Uparrow}{\Uparrow N, \Gamma \vdash \Delta \Uparrow}\ Store_L \qquad \frac{\Downarrow N \vdash}{N \Uparrow\ \vdash\ \Uparrow}\ Decide_L \qquad \frac{\Uparrow P \vdash\ \Uparrow}{\Downarrow P \vdash}\ Release_L$$

$$\frac{\Uparrow\ \vdash\ \Uparrow P}{\Uparrow\ \vdash P \Uparrow}\ Store_R \qquad \frac{\vdash P \Downarrow}{\Uparrow\ \vdash\ \Uparrow P}\ Decide_R \qquad \frac{\Uparrow\ \vdash N \Uparrow}{\vdash N \Downarrow}\ Release_R$$

Branching structural rules

$$\frac{}{\Downarrow t \neq t \vdash} \qquad \frac{}{\vdash t = t \Downarrow} \qquad \frac{}{\Downarrow f^- \vdash} \qquad \frac{}{\vdash t^+ \Downarrow}$$

$$\frac{\vdash A_1 \Downarrow \qquad \Downarrow A_2 \vdash}{\Downarrow A_1 \supset A_2 \vdash} \qquad \frac{\vdash A_1 \Downarrow \qquad \vdash A_2 \Downarrow}{\vdash A_1 \wedge^+ A_2 \Downarrow}$$

$$\frac{\Downarrow A_i \vdash}{\Downarrow A_1 \wedge^- A_2 \vdash} \qquad \frac{\vdash A_i \Downarrow}{\vdash A_1 \vee A_2 \Downarrow}$$

$$\frac{\Downarrow C\,t \vdash}{\Downarrow \forall x.\, C\,x \vdash} \qquad \frac{\vdash C\,t \Downarrow}{\vdash \exists x.\, C\,x \Downarrow}$$

$$\frac{N \Uparrow \Gamma \vdash \Delta \Uparrow}{\Uparrow N, \Gamma \vdash \Delta \Uparrow}\ Store_L \qquad \frac{\Downarrow N \vdash}{N \Uparrow\ \vdash\ \Uparrow}\ Decide_L \qquad \frac{\Uparrow P \vdash \Uparrow}{\Downarrow P \vdash}\ Release_L$$

$$\frac{\Uparrow\ \vdash\ \Uparrow P}{\Uparrow\ \vdash P \Uparrow}\ Store_R \qquad \frac{\vdash P \Downarrow}{\Uparrow\ \vdash\ \Uparrow P}\ Decide_R \qquad \frac{\Uparrow\ \vdash N \Uparrow}{\vdash N \Downarrow}\ Release_R$$

Switchable formulae

Multiple formulae can only exist inside $\Uparrow$-sequents.

A restriction on formulae is needed to ensure that there is *exactly one formula* in a sequent when there is a change of phase.

A μMALL formula is *switchable* if

- whenever a subformula $C \wedge^{+} D$ occurs negatively (under an odd number of implications), either C or D is purely positive
- whenever a subformula $C \supset D$ occurs positively (under an even number of implications), either C is purely positive or D is purely negative

Example switchable formulae

- purely positive formulae
- purely negative formulae

Example proof evidence

The following are typical kinds of proof evidence in model checking.

reachability	can be witnessed by a *path* through a graph
non-reachability	can be witnessed by a *reachable set* for one node that does not contain the other
(bi)similarity	in a given LTS can be witnessed by a set of pairs called, resp, *simulation* and *bisimulation*
non-bisimilarity	in a given LTS can be witnessed by a *Hennessy-Milner logic (HML) formula* that is satisfied by one but not by the other

Our challenge: How can we formally define such proof evidence in terms of μMALL proof theory?

Augmenting focused sequents

We augment all sequents in the focused proof system with certificates, by giving them an extra argument Ξ (encoding a certificate):

$$\begin{array}{r@{\;}l}\mathcal{N} \Uparrow \Gamma & \vdash \Delta \Uparrow \mathcal{P} \\ \Downarrow A & \vdash \\ & \vdash A \Downarrow\end{array}$$

Augmenting focused sequents

We augment all sequents in the focused proof system with certificates, by giving them an extra argument Ξ (encoding a certificate):

$$\Xi : \mathcal{N} \Uparrow \Gamma \vdash \Delta \Uparrow \mathcal{P}$$

$$\Xi : \Downarrow A \vdash$$

$$\Xi : \vdash A \Downarrow$$

Augmenting focused inference rules

Also, every inference rule gets an additional atomic premise.

In the $\Uparrow$ phase, a *clerk* performs some simple computations on the input certificate (Ξ_0) to produce continuation certificates (Ξ_1, Ξ_2):

$$\frac{\Xi_1 : \mathcal{N} \Uparrow A_1, \Gamma \vdash \Delta \Uparrow \qquad \Xi_2 : \mathcal{N} \Uparrow A_2, \Gamma \vdash \Delta \Uparrow \qquad \vee_c(\Xi_0, \Xi_1, \Xi_2)}{\Xi_0 : \mathcal{N} \Uparrow A_1 \vee A_2, \Gamma \vdash \Delta \Uparrow}$$

In the $\Downarrow$-sequent, an *expert* digs out information from the input certificate not only to compute continuation certificates (Ξ_1) but also additional guiding information (the term t):

$$\frac{\Xi_1 : \ \vdash C\, t \Downarrow \qquad \exists_e(\Xi_0, \Xi_1, t)}{\Xi_0 : \ \vdash \exists x.\, C\, x \Downarrow}$$

Augmented fixed point rules

$$\frac{\Xi_1\,\bar{y} : \Uparrow B\,S\,\bar{y} \vdash S\,\bar{y} \Uparrow \quad \Xi_2 : \mathcal{N} \Uparrow S\,\bar{t}, \Gamma \vdash \Delta \Uparrow \quad \mathrm{ind}(\Xi_0, \Xi_1, \Xi_2, S)}{\Xi_0 : \mathcal{N} \Uparrow \mu B\,\bar{t}, \Gamma \vdash \Delta \Uparrow}$$

$$\frac{\Xi_1 : \mathcal{N} \Uparrow B(\mu B)\bar{t}, \Gamma \vdash \Delta \Uparrow \quad \mu\text{-unfold}_L(\Xi_0, \Xi_1)}{\Xi_0 : \mathcal{N} \Uparrow \mu B\,\bar{t}, \Gamma \vdash \Delta \Uparrow}$$

$$\frac{\Xi_1 : \ \vdash B(\mu B)\bar{t} \Downarrow \quad \mu\text{-unfold}_R(\Xi_0, \Xi_1)}{\Xi_0 : \ \vdash \mu B\,\bar{t} \Downarrow}$$

Outline

1. The μF logic
2. Examples
   - Clerks & Experts
   - Certificates
3. Implementation

Common proof certificates: **sync**:`cert->cert`

Certificate constructor for a synchronous phase ($\forall\,\Xi$ is implied).
The right rules are:

$=^s_e(\mathbf{sync}(\Xi)).$

$\wedge^+{}_e(\mathbf{sync}(\Xi), \mathbf{sync}(\Xi), \mathbf{sync}(\Xi)).$

$\mu\text{-unfold}_R(\mathbf{sync}(\Xi), \mathbf{sync}(\Xi)).$

$\text{release}_R(\mathbf{sync}(\Xi), \Xi).$

$\vee_e(\mathbf{sync}(\Xi), \mathbf{sync}(\Xi), 1).$

$\vee_e(\mathbf{sync}(\Xi), \mathbf{sync}(\Xi), 2).$

$\forall T.\, \exists_e(\mathbf{sync}(\Xi), \mathbf{sync}(\Xi), T).$

Common proof certificates: **sync**:`cert->cert`

Certificate constructor for a synchronous phase ($\forall\,\Xi$ is implied).
The right rules are:

$=^s_e(\mathbf{sync}(\Xi)).$

$\wedge^+{}_e(\mathbf{sync}(\Xi), \mathbf{sync}(\Xi), \mathbf{sync}(\Xi)).$

$\mu\text{-unfold}_R(\mathbf{sync}(\Xi), \mathbf{sync}(\Xi)).$

$\text{release}_R(\mathbf{sync}(\Xi), \Xi).$

$\vee_e(\mathbf{sync}(\Xi), \mathbf{sync}(\Xi), 1).$

$\vee_e(\mathbf{sync}(\Xi), \mathbf{sync}(\Xi), 2).$

$\forall T.\, \exists_e(\mathbf{sync}(\Xi), \mathbf{sync}(\Xi), T).$

- unbounded synchronous search

Common proof certificates: **sync**: `cert->cert`

Certificate constructor for a synchronous phase ($\forall\, \Xi$ is implied).
The right rules are:

$=^{s}_{e}(\textbf{sync}(\Xi)).$

$\wedge^{+}{}_{e}(\textbf{sync}(\Xi), \textbf{sync}(\Xi), \textbf{sync}(\Xi)).$

$\mu\text{-unfold}_R(\textbf{sync}(\Xi), \textbf{sync}(\Xi)).$

$\text{release}_R(\textbf{sync}(\Xi), \Xi).$

$\vee_e(\textbf{sync}(\Xi), \textbf{sync}(\Xi), 1).$

$\vee_e(\textbf{sync}(\Xi), \textbf{sync}(\Xi), 2).$

$\forall T.\, \exists_e(\textbf{sync}(\Xi), \textbf{sync}(\Xi), T).$

- unbounded synchronous search
- no clerks, but a continuation certificate

Common proof certificates: **sync**:`cert->cert`

Certificate constructor for a synchronous phase ($\forall\,\Xi$ is implied).
The right rules are:

$=_e^s(\textbf{sync}(\Xi)).$

$\wedge^+{}_e(\textbf{sync}(\Xi), \textbf{sync}(\Xi), \textbf{sync}(\Xi)).$

$\mu\text{-unfold}_R(\textbf{sync}(\Xi), \textbf{sync}(\Xi)).$

$\text{release}_R(\textbf{sync}(\Xi), \Xi).$

$\vee_e(\textbf{sync}(\Xi), \textbf{sync}(\Xi), 1).$

$\vee_e(\textbf{sync}(\Xi), \textbf{sync}(\Xi), 2).$

$\forall T.\, \exists_e(\textbf{sync}(\Xi), \textbf{sync}(\Xi), T).$

- unbounded synchronous search
- no clerks, but a continuation certificate
- exhaustive non-deterministic search for $\vee$ and $\exists$

Common proof certificates: **async**: `cert->cert`

Certificate constructor for an asynchronous phase (dual of **sync**).
The left rules are:

$=^s_c(\textbf{async}(\Xi), \textbf{async}(\Xi)).$

$\wedge^+{}_c(\textbf{async}(\Xi), \textbf{async}(\Xi)).$

$\exists_c(\textbf{async}(\Xi), \lambda x.\, \textbf{async}(\Xi)).$

$\mu\text{-unfold}_L(\textbf{async}(\Xi), \textbf{async}(\Xi)).$

$\vee_c(\textbf{async}(\Xi), \textbf{async}(\Xi), \textbf{async}(\Xi)).$

$\text{store}_L(\textbf{async}(\Xi), \textbf{async}(\Xi)).$

$\text{decide}_L(\textbf{async}(\Xi), \Xi).$

Common proof certificates: **async**: `cert->cert`

Certificate constructor for an asynchronous phase (dual of **sync**).
The left rules are:

$=_c^s(\mathbf{async}(\Xi), \mathbf{async}(\Xi)).$ $\vee_c(\mathbf{async}(\Xi), \mathbf{async}(\Xi), \mathbf{async}(\Xi)).$

$\wedge^+{}_c(\mathbf{async}(\Xi), \mathbf{async}(\Xi)).$

$\exists_c(\mathbf{async}(\Xi), \lambda x.\, \mathbf{async}(\Xi)).$ $\text{store}_L(\mathbf{async}(\Xi), \mathbf{async}(\Xi)).$

$\mu\text{-unfold}_L(\mathbf{async}(\Xi), \mathbf{async}(\Xi)).$ $\text{decide}_L(\mathbf{async}(\Xi), \Xi).$

- unbounded asynchronous search

Common proof certificates: **async**: `cert->cert`

Certificate constructor for an asynchronous phase (dual of **sync**). The left rules are:

$=^s_c(\mathbf{async}(\Xi), \mathbf{async}(\Xi)).$

$\vee_c(\mathbf{async}(\Xi), \mathbf{async}(\Xi), \mathbf{async}(\Xi)).$

$\wedge^+{}_c(\mathbf{async}(\Xi), \mathbf{async}(\Xi)).$

$\exists_c(\mathbf{async}(\Xi), \lambda x.\, \mathbf{async}(\Xi)).$

$\mathrm{store}_L(\mathbf{async}(\Xi), \mathbf{async}(\Xi)).$

$\mu\text{-unfold}_L(\mathbf{async}(\Xi), \mathbf{async}(\Xi)).$

$\mathrm{decide}_L(\mathbf{async}(\Xi), \Xi).$

- unbounded asynchronous search
- no experts, apart from the decide rules, but a continuation certificate

Some other certificates

stop: `cert` authorizes no search (no clerk or expert is defined for this constant)

Some other certificates

stop : `cert` authorizes no search (no clerk or expert is defined for this constant)

$\textbf{bipole}_n$: `cert` is defined as a sequence of n composition of **async**(**sync**($\cdot$)) before a final **stop**

Some other certificates

stop: `cert` authorizes no search (no clerk or expert is defined for this constant)

bipole$_n$: `cert` is defined as a sequence of n composition of **async**(**sync**($\cdot$)) before a final **stop**

decproc: `cert` is short-hand for **bipole**$_\infty$, the unbounded version of **bipole**$_n$

Some other certificates

stop:`cert` authorizes no search (no clerk or expert is defined for this constant)

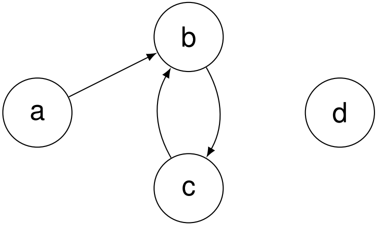
$\textbf{bipole}_n$:`cert` is defined as a sequence of n composition of **async**(**sync**($\cdot$)) before a final **stop**

decproc:`cert` is short-hand for $\textbf{bipole}_\infty$, the unbounded version of $\textbf{bipole}_n$

inv,**co-inv**:`(i->i->bool)->cert->cert` take an explicit invariant S and attempt to complete the invariant subproof as a single bipole:

$$\forall S.\, \mathrm{ind}(\textbf{inv}(S, \Xi), (\lambda\bar{x}.\, \textbf{bipole}), \Xi, S)$$
$$\forall S.\, \mathrm{co\text{-}ind}(\textbf{co-inv}(S, \Xi), \Xi, (\lambda\bar{x}.\, \textbf{bipole}), S)$$

Lists as reachability certificates



Certificate for $\vdash path(x, y)$: list of nodes between x and y:

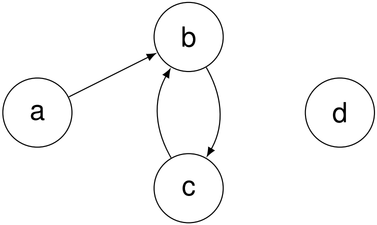
$$\Xi_{a,c} : \vdash path(a, c) \quad \text{for} \quad \Xi_{a,c} \in \{[b], [b; c; b], \ldots\}$$

$$\forall L.\, \text{decide}_R(L, L). \qquad \forall L.\, \mu\text{-unfold}_R(L, L).$$

$$\forall X \forall L.\, \vee_e(X :: L, X :: L, 2). \qquad \forall L.\, \vee_e(nil, \mathbf{sync}(\mathbf{stop}), 1).$$

$$\forall X \forall L.\, \exists_e(X :: L, L, X). \qquad \forall L.\, \wedge^+{}_e(L, \mathbf{sync}(\mathbf{stop}), L).$$

Lists as reachability certificates



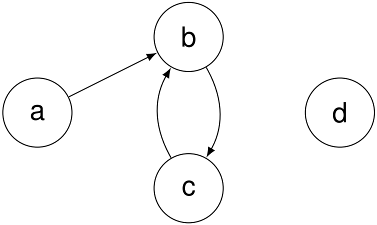
$$path = \mu\Big(\lambda P \lambda x \lambda z.\, x \longrightarrow z \vee (\exists y.\, x \longrightarrow y \wedge^{+} P\, y\, z)\Big)$$

$\forall L.\, \mathrm{decide}_R(L, L).$ $\quad\quad$ $\forall L.\, \mu\text{-}\mathrm{unfold}_R(L, L).$

$\forall X \forall L.\, \vee_e(X :: L, X :: L, 2).$ $\quad\quad$ $\forall L.\, \vee_e(nil, \mathbf{sync}(\mathbf{stop}), 1).$

$\forall X \forall L.\, \exists_e(X :: L, L, X).$ $\quad\quad$ $\forall L.\, \wedge^{+}{}_e(L, \mathbf{sync}(\mathbf{stop}), L).$

Lists as reachability certificates



$$path = \mu\Big(\lambda P \lambda x \lambda z.\, x \longrightarrow z \vee (\exists y.\, x \longrightarrow y \wedge^{+} P\, y\, z)\Big)$$

$\forall L.\, \mathrm{decide}_R(L, L).$

$\forall X \forall L.\, \vee_e(X :: L, X :: L, 2).$

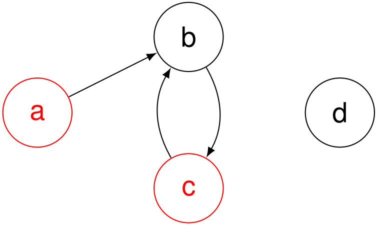
$\forall X \forall L.\, \exists_e(X :: L, L, X).$

$\forall L.\, \mu\text{-unfold}_R(L, L).$

$\forall L.\, \vee_e(nil, \textbf{sync}(\textbf{stop}), 1).$

$\forall L.\, \wedge^{+}{}_e(L, \textbf{sync}(\textbf{stop}), L).$

Lists as reachability certificates



$[b] \; : \vdash path(a, c)$

$$path = \mu\Big(\lambda P \lambda x \lambda z.\, x \longrightarrow z \vee (\exists y.\, x \longrightarrow y \wedge^{+} P\, y\, z)\Big)$$

$\forall L.\, \mathrm{decide}_R(L, L).$

$\forall X \forall L.\, \vee_e(X :: L, X :: L, 2).$

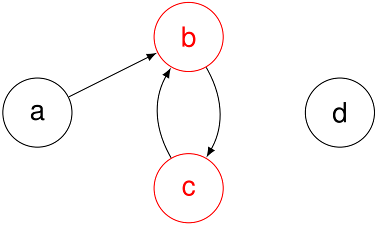
$\forall X \forall L.\, \exists_e(X :: L, L, X).$

$\forall L.\, \mu\text{-unfold}_R(L, L).$

$\forall L.\, \vee_e(nil, \mathbf{sync}(\mathbf{stop}), 1).$

$\forall L.\, \wedge^{+}{}_e(L, \mathbf{sync}(\mathbf{stop}), L).$

Lists as reachability certificates



$[\,] \ : \vdash path(b, c)$

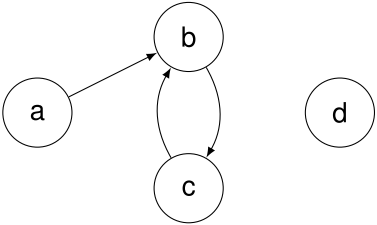
$$path = \mu\Big(\lambda P \lambda x \lambda z.\, x \longrightarrow z \vee (\exists y.\, x \longrightarrow y \wedge^{+} P\, y\, z)\Big)$$

$$\forall L.\, \text{decide}_R(L, L). \qquad \forall L.\, \mu\text{-unfold}_R(L, L).$$

$$\forall X \forall L.\, \vee_e(X :: L, X :: L, 2). \qquad \forall L.\, \vee_e(nil, \mathbf{sync}(\mathbf{stop}), 1).$$

$$\forall X \forall L.\, \exists_e(X :: L, L, X). \qquad \forall L.\, \wedge^{+}{}_e(L, \mathbf{sync}(\mathbf{stop}), L).$$

Lists as reachability certificates



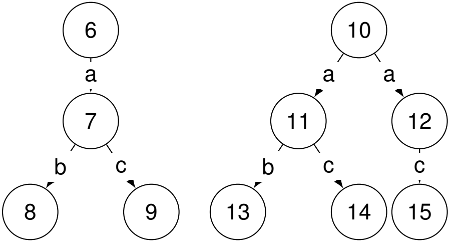
$$path = \mu\Big(\lambda P \lambda x \lambda z.\, x \longrightarrow z \vee (\exists y.\, x \longrightarrow y \wedge^{+} P\, y\, z)\Big)$$

$$\forall L.\, \mathrm{decide}_R(L, L). \qquad \forall L.\, \mu\text{-unfold}_R(L, L).$$

$$\forall X \forall L.\, \vee_e(X :: L, X :: L, 2). \qquad \forall L.\, \vee_e(nil, \mathbf{sync(stop)}, 1).$$

$$\forall X \forall L.\, \exists_e(X :: L, L, X). \qquad \forall L.\, \wedge^{+}{}_e(L, \mathbf{sync(stop)}, L).$$

HML assertions as non-bisimulation certificates



$$A := \bigwedge_{i \in I} B_i$$

$$B := \langle a_i \rangle A_i \mid \neg(\langle a_i \rangle A_i)$$

Certificate for $bisim(x, y) \vdash$: Hennessy-Milner Language formula A such that $x \models A$ but $y \not\models A$:

$$\Xi_{6,10} : bisim(6, 10) \vdash \quad \text{for} \quad \Xi_{6,10} \in \{\langle a \rangle \neg \langle b \rangle true, \ldots\}$$

as $10 \models \Xi$ but $6 \not\models \Xi$.

HML assertions as non-bisimulation certificates (continued)

$$\forall A.\,\mathrm{store}_L(A, A). \qquad \forall (B_i)_i \forall j.\,\mathrm{decide}_L(\textstyle\bigwedge_i B_i, B_j).$$

$$\forall B.\,\nu\text{-unfold}_L(B, B). \qquad \forall a \forall A.\, \wedge^-{}_e(\ \langle a\rangle A, \langle a\rangle A, \mathit{left}).$$

$$\forall a \forall A.\, \forall_e(\langle a\rangle A, A, a). \qquad \forall a \forall A.\, \wedge^-{}_e(\neg\langle a\rangle A, \langle a\rangle A, \mathit{right}).$$

$$\forall T \forall A.\, \forall_e(A, A, T). \qquad \forall A.\, \supset_e(A, \mathbf{sync}(\mathbf{stop}), A).$$

$$\forall A.\,\mathrm{release}_L(A, A).$$

$$\forall A.\, \exists_c(A, \lambda x.\, A). \qquad \forall A.\, \wedge^+{}_c(A, A).$$

$$\forall A.\, \mu\text{-unfold}_L(A, A). \qquad \forall A.\, \vee_c(A, A, A).$$

$$\forall A.\, =^s_c(A, A).$$

HML assertions as non-bisimulation certificates (continued)

$$\forall A.\, \mathrm{store}_L(A, A). \qquad \forall (B_i)_i \forall j.\, \mathrm{decide}_L(\bigwedge_i B_i, B_j).$$

$$\forall B.\, \nu\text{-unfold}_L(B, B). \qquad \forall a \forall A.\, \wedge^-_e(\ \langle a \rangle A, \langle a \rangle A, \mathit{left}).$$

$$\forall a \forall A.\, \forall_e(\langle a \rangle A, A, a). \qquad \forall a \forall A.\, \wedge^-_e(\neg\langle a \rangle A, \langle a \rangle A, \mathit{right}).$$

$$\forall T \forall A.\, \forall_e(A, A, T). \qquad \forall A.\, \supset_e(A, \mathbf{sync}(\mathbf{stop}), A).$$

$$\forall A.\, \mathrm{release}_L(A, A).$$

$$\forall A.\, \exists_c(A, \lambda x.\, A). \qquad \forall A.\, \wedge^+_c(A, A).$$

$$\forall A.\, \mu\text{-unfold}_L(A, A). \qquad \forall A.\, \vee_c(A, A, A).$$

$$\forall A.\, =^s_c(A, A).$$

HML assertions as non-bisimulation certificates (continued)

$$\forall A.\,\mathrm{store}_L(A, A). \qquad \forall (B_i)_i \forall j.\,\mathrm{decide}_L(\textstyle\bigwedge_i B_i, B_j).$$

$$\forall B.\,\nu\text{-}\mathrm{unfold}_L(B, B). \qquad \forall a \forall A.\,\wedge^-{}_e(\langle a\rangle A, \langle a\rangle A, \mathit{left}).$$

$$\forall a \forall A.\,\forall_e(\langle a\rangle A, A, a). \qquad \forall a \forall A.\,\wedge^-{}_e(\neg\langle a\rangle A, \langle a\rangle A, \mathit{right}).$$

$$\forall T \forall A.\,\forall_e(A, A, T). \qquad \forall A.\,\supset_e(A, \mathbf{sync}(\mathbf{stop}), A).$$

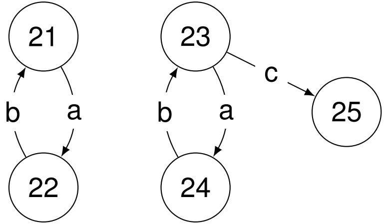
$$\forall A.\,\mathrm{release}_L(A, A).$$

$$\forall A.\,\exists_c(A, \lambda x.\,A). \qquad \forall A.\,\wedge^+{}_c(A, A).$$

$$\forall A.\,\mu\text{-}\mathrm{unfold}_L(A, A). \qquad \forall A.\,\vee_c(A, A, A).$$

$$\forall A.\,=^s_c(A, A).$$

Invariants as simulation certificates



The set $\{(21, 23), (22, 24)\}$ is a simulation and, therefore, the process (21) is simulated by the process (23).
From this set we build

$$S = \lambda x \lambda y. (x = 21 \wedge^{+} y = 23) \vee (x = 22 \wedge^{+} y = 24)$$

which is such that

$$\textbf{co-inv}\,(S, \textbf{bipole}) \; : \; \vdash sim(21, 23)$$

Outline

A reference proof checker

We have built a reference proof checker within the Bedwyr computational logic system:
`http://slimmer.gforge.inria.fr/bedwyr/pcmc/`

- implemented by Tiu, Baelde, Gacek, & Heath
- λProlog is not strong enough for checking these certificates

Future Plans

How not to put invariants into proof certificates?

- obvious induction invariant
- bisimulation up-to, etc.

Combine proof checking for both stage 1 and stage 2.

Embrace much more of model checking.

- predicate abstractions
- tables and lemmas
- partial order reductions

Build proof certificates for the Abella prover, thereby merging model checking and inductive theorem proving into one platform.