The TLA$^+$ proof system

Stephan Merz

Kaustuv Chaudhuri, Damien Doligez, Leslie Lamport

INRIA Nancy & INRIA-MSR Joint Centre, France

Amir Pnueli Memorial Symposium

New York University, May 8, 2010

Amir Pnueli: Deduction is Forever (FM'99)

Just as it was unavoidable, due to the growing complexity of circuits, that circuit manufacturers started to employ formal methods for verifying their designs, it is equally inevitable that more of the practicing verifiers will turn to deductive technologies, due to their significantly better scalability.

Amir Pnueli: Deduction is Forever (contd.)

- For verifying an invariant p over a finite-state system:
  - it is usually much cheaper to check

$$\Theta \Rightarrow \varphi \qquad \varphi \wedge \rho \Rightarrow \varphi' \qquad \varphi \Rightarrow p$$

  - than computing $\overline{\rho}^*(\neg p)$ by state space exploration.

Amir Pnueli: Deduction is Forever (contd.)

- For verifying an invariant p over a finite-state system:
  - it is usually much cheaper to check

    $$\Theta \Rightarrow \varphi \qquad \varphi \wedge \rho \Rightarrow \varphi' \qquad \varphi \Rightarrow p$$

  - than computing $\overline{\rho}^*(\neg p)$ by state space exploration.

- Main differences between deduction and exploration:
  - deduction is based on induction while exploration computes the set of reachable states,
  - deduction uses a more expressive language including quantifiers, leading to succinct specification of parameterized systems,
  - deduction requires user ingenuity and interaction.

Contents

Invariance proofs in TLA$^+$

- Elementary rule for proving invariants

$$\frac{I \wedge [N]_v \Rightarrow I'}{I \wedge \Box[N]_v \Rightarrow \Box I}$$

THEOREM $Inv1 \triangleq$ ASSUME STATE I, STATE v, ACTION N,
$I \wedge [N]_v \Rightarrow I'$
PROVE $I \wedge \Box[N]_v \Rightarrow \Box I$

Invariance proofs in TLA$^+$

- Elementary rule for proving invariants

$$\frac{I \wedge [N]_v \Rightarrow I'}{I \wedge \Box[N]_v \Rightarrow \Box I}$$

THEOREM $Inv1 \triangleq$ ASSUME STATE I, STATE v, ACTION N,
$I \wedge [N]_v \Rightarrow I'$
PROVE $I \wedge \Box[N]_v \Rightarrow \Box I$

- Schema for invariant proofs

$Spec \triangleq Init \wedge \Box[Next]_{vars} \wedge L$

THEOREM $Spec \Rightarrow Inv$
$\langle 1 \rangle 1.$ $Init \Rightarrow Inv$
$\langle 1 \rangle 2.$ $Inv \wedge Next \Rightarrow Inv'$
$\langle 1 \rangle 3.$ $Inv \wedge$ UNCHANGED $vars \Rightarrow Inv'$
$\langle 1 \rangle 4.$ QED
BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $Inv1$ DEF $Spec$

Invariance proofs in TLA$^+$

- Elementary rule for proving invariants

$$\frac{I \wedge [N]_v \Rightarrow I'}{I \wedge \Box[N]_v \Rightarrow \Box I}$$

THEOREM $Inv1 \triangleq$ ASSUME STATE I, STATE v, ACTION N,
$I \wedge [N]_v \Rightarrow I'$
PROVE $I \wedge \Box[N]_v \Rightarrow \Box I$

- Schema for invariant proofs

$Spec \triangleq Init \wedge \Box[Next]_{vars} \wedge L$

THEOREM $Spec \Rightarrow Inv$
$\langle 1 \rangle 1.$ $Init \Rightarrow Inv$
$\langle 1 \rangle 2.$ $Inv \wedge Next \Rightarrow Inv'$
$\langle 1 \rangle 3.$ $Inv \wedge$ UNCHANGED $vars \Rightarrow Inv'$
$\langle 1 \rangle 4.$ QED
BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $Inv1$ DEF $Spec$

no temporal logic here!

Reasoning about actions

- About 95% of proof steps do not involve temporal logic
  - reasoning about state predicates or state transitions
  - first-order reasoning where v and v' are distinct variables
- Aim for as much automation as possible ...
  - open proof system: harness power of different prover back-ends
  - first-order logic, rewriting, SAT and SMT solving etc.
  - ensure overall correctness by proof certification
- ... but encourage users to maintain readable proofs
  - declarative, hierarchical proof language
  - prefer an extra level of interaction over obscure automatic tactics

Proving trivial steps

- Expand definitions and discharge automatically

$$
\begin{array}{lll}
\mathit{Init} & \triangleq & \land\ \mathit{pc} = [i \in \{0,1\} \mapsto \text{“a0”}] \\
& & \land\ \mathit{turn} = 0 \\
& & \land\ \mathit{flag} = [i \in \{0,1\} \mapsto \text{FALSE}] \\
\mathit{Inv} & \triangleq & \land\ \mathit{pc} \in [\{0,1\} \to \{\text{“a0”}, \text{“a1”}, \text{“a2”}, \text{“a3a”}, \text{“a3b”}, \text{“cs”}, \text{“a4”}\}] \\
& & \land\ \mathit{turn} \in \{0,1\} \\
& & \land\ \mathit{flag} \in [\{0,1\} \to \text{BOOLEAN}] \\
& & \land\ \forall i \in \{0,1\} : \land\ \mathit{pc}[i] \in \{\text{“a2”}, \text{“a3a”}, \text{“a3b”}, \text{“cs”}, \text{“a4”}\} \Rightarrow \mathit{flag}[i] \\
& & \qquad\qquad \land\ \mathit{pc}[i] \in \{\text{“cs”}, \text{“a4”}\} \\
& & \qquad\qquad\quad \Rightarrow\ \land\ \mathit{pc}[1-i] \notin \{\text{“cs”}, \text{“a4”}\} \\
& & \qquad\qquad\qquad\ \land\ \mathit{pc}[1-i] \in \{\text{“a3a”}, \text{“a3b”}\} \Rightarrow \mathit{turn} = i
\end{array}
$$

$\langle 1 \rangle 1.\ \mathit{Init} \Rightarrow \mathit{Inv}$

BY DEFS $\mathit{Init}, \mathit{Inv}$

When automatic proof fails . . .

- Decompose proof into a sequence of "simpler" steps

$$Next \triangleq \exists i \in \{0,1\} : Proc(i)$$
$$Proc(i) \triangleq a0(i) \vee a1(i) \vee \ldots \vee a4(i)$$

$\langle 1 \rangle 2.\ Inv \wedge Next \Rightarrow Inv'$

When automatic proof fails ...

- Decompose proof into a sequence of "simpler" steps

$Next \triangleq \exists i \in \{0,1\} : Proc(i)$
$Proc(i) \triangleq a0(i) \vee a1(i) \vee \ldots \vee a4(i)$

$\langle 1 \rangle 2.\ Inv \wedge Next \Rightarrow Inv'$
  $\langle 2 \rangle 1.$ SUFFICES ASSUME Inv, NEW $i \in \{0,1\}$, $Proc(i)$
    PROVE Inv'
    BY DEF $Next$
  $\langle 2 \rangle 2.$ CASE $a0(i)$
  $\langle 2 \rangle 3.$ CASE $a1(i)$
  ...
  $\langle 2 \rangle 8.$ CASE $a4(i)$
  $\langle 2 \rangle 9.$ QED
    BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, ..., $\langle 2 \rangle 8$ DEF $Proc$

Contents

TLA$^+$ proof language

- Hierarchical, declarative proof
  - linear representation of proof tree
  - step labels $\langle d \rangle lbl$ (where d is the depth of the step)
  - steps assert sequents ASSUME ... PROVE ...
  - top-down development: refine assertions until they are "obvious"
  - leaf: invoke proof method, citing necessary assumptions and facts
- Controlling the use of assumptions, facts, and definitions
  - limit search space for automatic provers
  - require explicit citation of assumptions, facts, and definitions ...
  - ... or make them usable throughout the current scope

Example: proof of Cantor's theorem

THEOREM ASSUME NEW S, NEW $f \in [S \to \text{SUBSET } S]$
PROVE $\exists A \in \text{SUBSET } S : \forall x \in S : f[x] \neq A$

$\langle 1 \rangle$. DEFINE $T \triangleq \{z \in S : z \notin f[z]\}$

$\langle 1 \rangle 1$. $\forall x \in S : f[x] \neq T$

$\langle 1 \rangle 2$. QED BY $\langle 1 \rangle 1$

Example: proof of Cantor's theorem

THEOREM ASSUME NEW S, NEW $f \in [S \to \text{SUBSET } S]$
PROVE $\exists A \in \text{SUBSET } S : \forall x \in S : f[x] \neq A$

$\langle 1 \rangle$. DEFINE $T \triangleq \{z \in S : z \notin f[z]\}$
$\langle 1 \rangle 1$. $\forall x \in S : f[x] \neq T$
$\langle 2 \rangle 1$. ASSUME NEW $x \in S$ PROVE $f[x] \neq T$

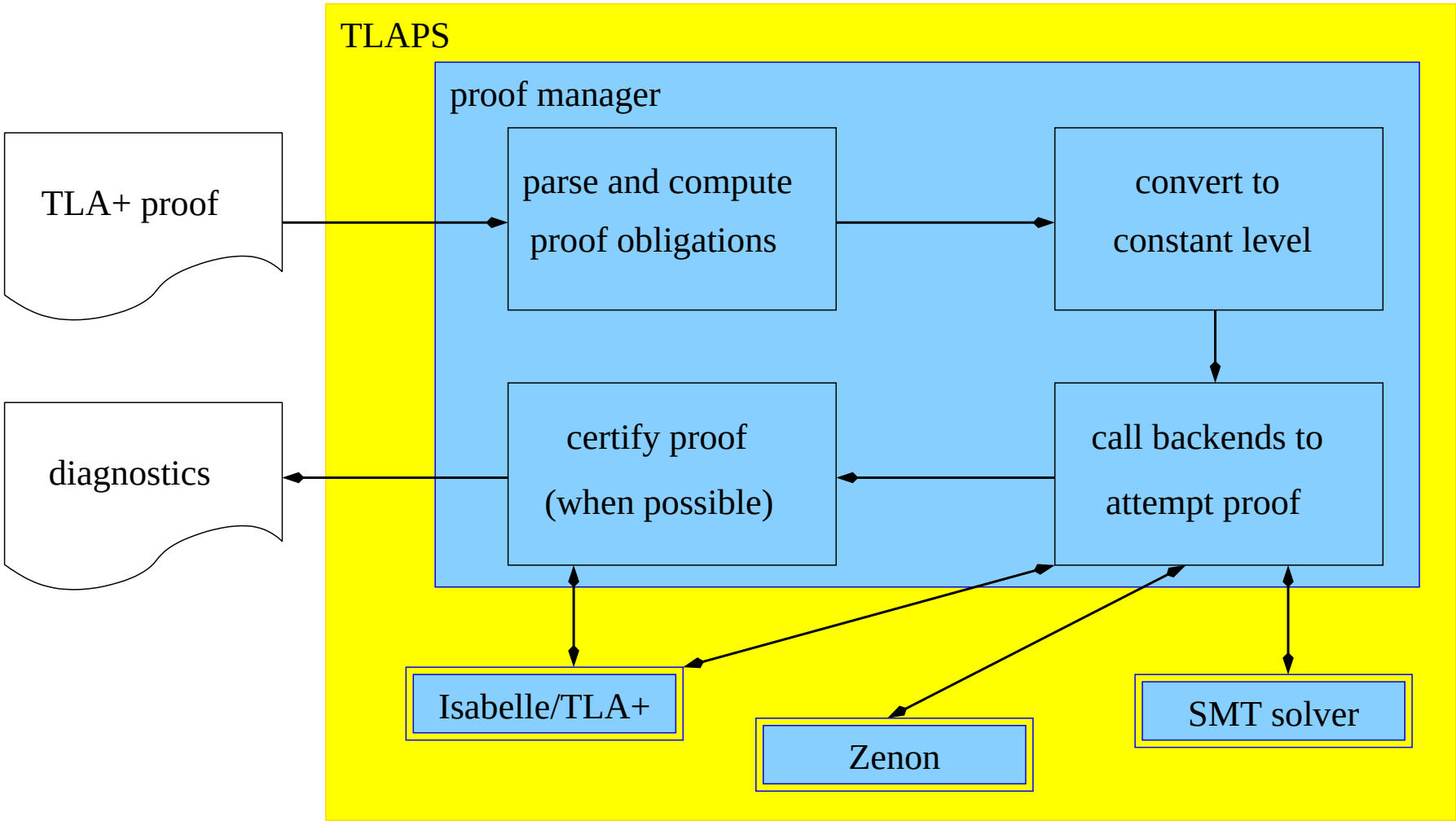
$\langle 2 \rangle 2$. QED BY $\langle 2 \rangle 1$
$\langle 1 \rangle 2$. QED BY $\langle 1 \rangle 1$

Example: proof of Cantor's theorem

THEOREM ASSUME NEW S, NEW $f \in [S \to \text{SUBSET } S]$
PROVE $\exists A \in \text{SUBSET } S : \forall x \in S : f[x] \neq A$

$\langle 1 \rangle$. DEFINE $T \triangleq \{z \in S : z \notin f[z]\}$
$\langle 1 \rangle 1$. $\forall x \in S : f[x] \neq T$
$\langle 2 \rangle 1$. ASSUME NEW $x \in S$ PROVE $f[x] \neq T$
$\langle 3 \rangle 1$. CASE $x \in T$ OBVIOUS
$\langle 3 \rangle 2$. CASE $x \notin T$ OBVIOUS
$\langle 3 \rangle 3$. QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$
$\langle 2 \rangle 2$. QED BY $\langle 2 \rangle 1$
$\langle 1 \rangle 2$. QED BY $\langle 1 \rangle 1$

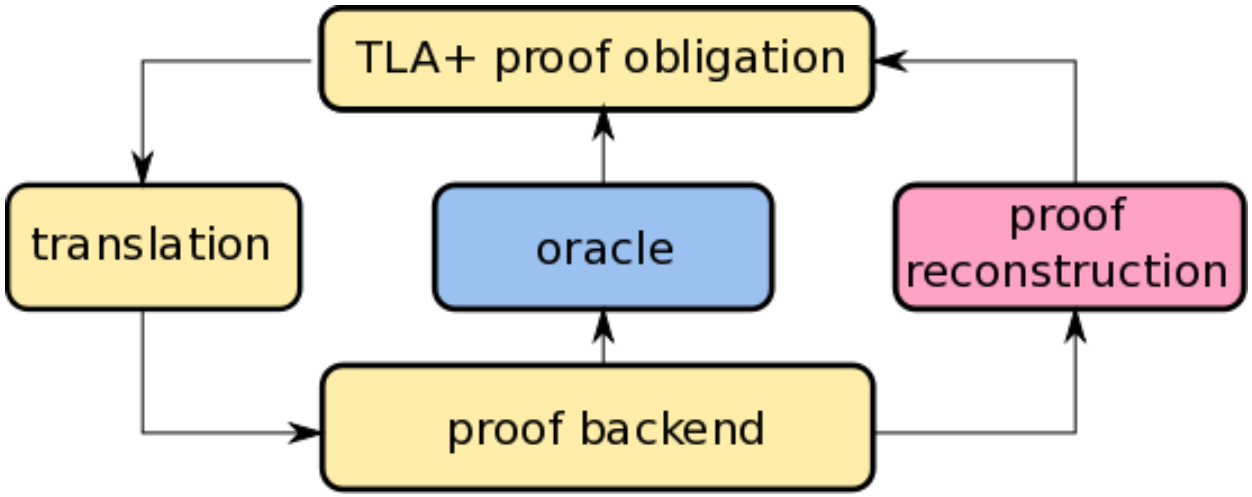
System architecture



Proof Manager

- Interpret hierarchical TLA$^+$ proof
  - manage assumptions and current goal
  - expand operator definitions if they are USEd
- Rewrite proof obligations to constant level
  - handle primed expressions such as Inv'
  - distribute prime over (constant-level) operators
  - introduce distinct variables e and e' for atomic state expression e
- Invoke back-end provers
  - user chooses which prover to use (default: Zenon, then Isabelle)

Proof reconstruction



- Oracle: trusted external reasoner
  - simple, but error-prone
  - translation and backend part of trusted code base
- Proof reconstruction: skeptical integration
  - replay proofs in trusted proof assistant (Isabelle/TLA$^+$)
  - reconstruction should be cheap

Isabelle/TLA$^+$

- Encoding of TLA$^+$ as Isabelle object logic
  - TLA$^+$ is untyped: incompatible with existing object logics
  - Isabelle/TLA$^+$ provides a library of standard data structures
- Instantiate automated proof methods
  - exploit genericity of Isabelle as a logical framework
  - tableau prover, rewriting engine, and combinations
- Trusted backend for proof reconstruction
  - formal definition of (constant-level) TLA$^+$ semantics

Contents

Current state of the TLAPS

- First public release: april 2010
  - `http://msr-inria.inria.fr/~doligez/tlaps/`
  - binary release includes prover backends
  - batch processing of proofs, Emacs interface
- Restricted to proving safety properties
  - invariant and step simulation (refinement) proofs
  - several examples provided in the distribution
- Looking forward to user feedback

Current work

- Support for temporal logic proofs (liveness properties)
  - extend proof manager to track temporal contexts
    $\leadsto$ hierarchical proof language helps separating contexts
  - encode semantics of temporal logic in Isabelle/TLA$^+$
  - TLA-specific proof rules
  - decision procedure for propositional temporal logic
- Improve user interface
  - integration with TLA$^+$ toolbox (Eclipse-based)
  - focus on specific subproofs, selectively rerun proof fragments
  - display and select usable assumptions

Amir Pnueli, again

The previous examples indicated the great raw potential possessed by the deductive technology which can often be impressively utilized by experts. We also pointed out that user ingenuity is an essential ingredient in its application.

The main challenge is to propose a working deductive tool and associated methodology which can be used by the "masses".

- Automatic invariant generation
- Counter-examples from failed proof attempts
- Generic proof methods developed by experts