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
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.
For verifying an invariant $p$ over a finite-state system:

- it is usually much cheaper to check
  \[ \Theta \Rightarrow \varphi \quad \varphi \land \rho \Rightarrow \varphi' \quad \varphi \Rightarrow p \]
- than computing $\overline{\rho^*}(\neg p)$ by state space exploration.
For verifying an invariant $p$ over a finite-state system:

- it is usually much cheaper to check
  
  $\Theta \Rightarrow \varphi \quad \varphi \land \rho \Rightarrow \varphi' \quad \varphi \Rightarrow p$

- 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.
1 Using the TLA$^+$ proof system for proving invariants

2 The TLA$^+$ proof language and system

3 Conclusion and outlook
Invariance proofs in TLA$^+$

- Elementary rule for proving invariants

\[
\begin{align*}
& I \land [N]_v \Rightarrow I' \\
& I \land \Box [N]_v \Rightarrow \Box I
\end{align*}
\]

\[
\text{THEOREM Inv1} \triangleq \text{ASSUME STATE } I, \text{ STATE } v, \text{ ACTION } N, \]

\[
I \land [N]_v \Rightarrow I'
\]

\[
\text{PROVE } I \land \Box [N]_v \Rightarrow \Box I
\]
Invariance proofs in TLA⁺

- Elementary rule for proving invariants

\[
\frac{I \land [N]v \Rightarrow I'}{I \land \Box [N]v \Rightarrow \Box I}
\]

THEOREM Inv1 \(\triangleq\) ASSUME STATE \(I\), STATE \(v\), ACTION \(N\),
\[I \land [N]v \Rightarrow I'\]
PROVE \(I \land \Box [N]v \Rightarrow \Box I\)

- Schema for invariant proofs

\[
Spec \triangleq Init \land \Box [Next]vars \land L
\]
THEOREM Spec \(\Rightarrow\) Inv
\[\langle 1 \rangle 1. \ Initi \Rightarrow Inv\]
\[\langle 1 \rangle 2. \ Inv \land Next \Rightarrow Inv'\]
\[\langle 1 \rangle 3. \ Inv \land UNCHANGED \ vars \Rightarrow Inv'\]
\[\langle 1 \rangle 4. \ \text{QED}\]
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**

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

**THEOREM Inv1** \(\triangleq\) ASSUME STATE \(I\), STATE \(v\), ACTION \(N\),  
\[I \land [N]_v \Rightarrow I'\]  
PROVE \[I \land \Box [N]_v \Rightarrow \Box I\]

- **Schema for invariant proofs**

\[Spec \triangleq Init \land \Box [Next]_vars \land L\]

**THEOREM** Spec \(\Rightarrow\) Inv  
\(\langle 1 \rangle 1.\) Init \(\Rightarrow\) Inv  
\(\langle 1 \rangle 2.\) Inv \(\land\) Next \(\Rightarrow\) Inv'  
\(\langle 1 \rangle 3.\) Inv \(\land\) 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

\[\begin{align*}
\text{Init} & \triangleq \land pc = [i \in \{0, 1\} \mapsto \text{“a0”}] \\
& \land turn = 0 \\
& \land flag = [i \in \{0, 1\} \mapsto \text{FALSE}]
\end{align*}\]

\[\begin{align*}
\text{Inv} & \triangleq \land pc \in [\{0, 1\} \rightarrow \{\text{“a0”, “a1”, “a2”, “a3a”, “a3b”, “cs”, “a4”}\}] \\
& \land turn \in \{0, 1\} \\
& \land flag \in [\{0, 1\} \rightarrow \text{BOOLEAN}] \\
& \land \forall i \in \{0, 1\} : \land pc[i] \in \{\text{“a2”, “a3a”, “a3b”, “cs”, “a4”}\} \Rightarrow flag[i] \\
& \land pc[i] \in \{\text{“cs”, “a4”}\} \\
& \quad \Rightarrow \land pc[1 - i] \notin \{\text{“cs”, “a4”}\} \\
& \quad \land pc[1 - i] \in \{\text{“a3a”, “a3b”}\} \Rightarrow turn = i
\end{align*}\]

\[(1)\text{1. Init} \Rightarrow \text{Inv}\]

BY DEFS Init, Inv
When automatic proof fails . . .

- Decompose proof into a sequence of “simpler” steps

\[
\begin{align*}
  \text{Next} & \triangleq \exists i \in \{0, 1\} : \text{Proc}(i) \\
  \text{Proc}(i) & \triangleq a_0(i) \lor a_1(i) \lor \ldots \lor a_4(i)
\end{align*}
\]

\(\langle 1\rangle 2. \ Inv \land \text{Next} \Rightarrow Inv'\)
When automatic proof fails …

- Decompose proof into a sequence of “simpler” steps

\[
\begin{align*}
Next & \triangleq \exists i \in \{0, 1\} : Proc(i) \\
Proc(i) & \triangleq a0(i) \lor a1(i) \lor \ldots \lor a4(i)
\end{align*}
\]

\[
\langle 1 \rangle 2. \ Inv \land Next \Rightarrow Inv'
\]

\[
\langle 2 \rangle 1. \text{ SUFFICES ASSUME } Inv, \text{ NEW } i \in \{0, 1\}, \text{ Proc}(i)
\]

\[
\text{PROVE } \ Inv'
\]

\[
\text{BY DEF } Next
\]

\[
\langle 2 \rangle 2. \text{ CASE } a0(i)
\]

\[
\langle 2 \rangle 3. \text{ CASE } a1(i)
\]

\[
\ldots
\]

\[
\langle 2 \rangle 8. \text{ CASE } a4(i)
\]

\[
\langle 2 \rangle 9. \text{ QED}
\]

\[
\text{BY}\langle 2 \rangle 2, \langle 2 \rangle 3, \ldots, \langle 2 \rangle 8 \text{ DEF Proc}
\]
Contents

1. Using the TLA\(^+\) proof system for proving invariants

2. The TLA\(^+\) proof language and system

3. Conclusion and outlook
**TLA⁺ proof language**

- Hierarchical, declarative proof
  - linear representation of proof tree
  - step labels $⟨d⟩lbr$ (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

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

\langle 1 \rangle. \quad \text{DEFINE } T \overset{\Delta}{=} \{z \in S : z \notin f[z]\}\]
\langle 1 \rangle 1. \quad \forall x \in S : f[x] \neq T

\langle 1 \rangle 2. \quad \text{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 \( \exists A \in \text{SUBSET } S : \forall x \in S : f[x] \neq A \)

\( \langle 1 \rangle . \) \text{DEFINE } T \overset{\Delta}{=} \{ z \in S : z \notin f[z] \} \\
\( \langle 1 \rangle 1. \) \( \forall x \in S : f[x] \neq T \)
\( \langle 2 \rangle 1. \) \text{ASSUME NEW } x \in S \text{ 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 $\exists A \in \text{SUBSET } S : \forall x \in S : f[x] \neq A$

⟨1⟩.
DEFINE $T \triangleq \{ z \in S : z \notin f[z] \}$
⟨1⟩1. $\forall x \in S : f[x] \neq T$

⟨2⟩1. ASSUME NEW $x \in S$ PROVE $f[x] \neq T$
   ⟨3⟩1. CASE $x \in T$ OBVIOUS
   ⟨3⟩2. CASE $x \notin T$ OBVIOUS
   ⟨3⟩3. QED BY ⟨3⟩1, ⟨3⟩2

⟨2⟩2. QED BY ⟨2⟩1
⟨1⟩2. QED BY ⟨1⟩1
Stephan Merz (INRIA Nancy)

The TLA\textsuperscript{+} proof system

Amir Pnueli 2010

12 / 19
Proof Manager

- Interpret hierarchical TLA\(^+\) proof
  - manage assumptions and current goal
  - expand operator definitions if they are \texttt{USE}\n
- Rewrite proof obligations to constant level
  - handle primed expressions such as \texttt{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
1. Using the TLA\(^+\) proof system for proving invariants

2. The TLA\(^+\) proof language and system

3. Conclusion and outlook
Current state of the TLAPS

- First public release: April 2010
  - [http://msr-inria.inria.fr/~doligez/tlaps/](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
  - 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