The TLA$^+$ Proof System

Stephan Merz

https://members.loria.fr/Stephan.Merz/

Inria Center at University of Lorraine & LORIA

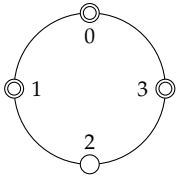
IFIP Working Group 2.3
Athens, May 2025

Context

- TLA$^+$: language for specifying (distributed) algorithms
  - mathematical set theory for describing data structures
  - linear-time temporal logic for describing executions of algorithms
  - represent state machines as formulas
- Tool support
  - TLC: explicit-state model checker (Yu et al., 1999)
  - Apalache: SMT-based symbolic model checker (Konnov et al., 2019)
  - PlusCal: front-end algorithmic language (Lamport, 2009)
  - IDEs: TLA$^+$ Toolbox, VS Code Extension
- TLAPS: interactive proof support for reasoning about TLA$^+$ specifications

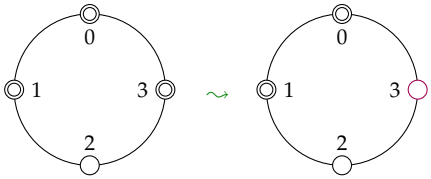
Outline

Problem Statement



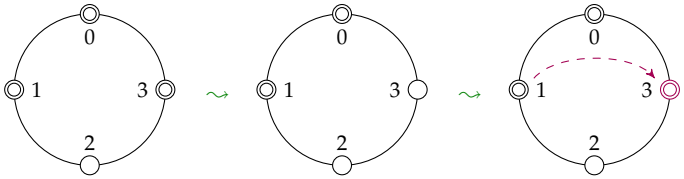
- Nodes in a distributed system perform some computation
  - nodes can be active (double circle) or inactive (simple circle)

Problem Statement



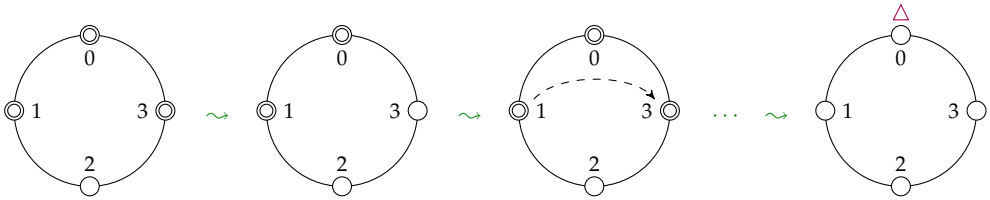
- Nodes in a distributed system perform some computation
  - nodes can be active (double circle) or inactive (simple circle)
- Possible transitions
  - an active node may locally terminate its computation

Problem Statement



- Nodes in a distributed system perform some computation
  - nodes can be active (double circle) or inactive (simple circle)
- Possible transitions
  - an active node may locally terminate its computation
  - an active node may send a message to another node, activating the receiver

Problem Statement



- Nodes in a distributed system perform some computation
  - nodes can be active (double circle) or inactive (simple circle)
- Possible transitions
  - an active node may locally terminate its computation
  - an active node may send a message to another node, activating the receiver
- Eventually, the system may signal global termination

Modeling Termination Detection in TLA$^+$: System Configurations

MODULE *TerminationDetection*

EXTENDS *Naturals*

CONSTANT N

ASSUME $NAssumption \triangleq N \in Nat \setminus \{0\}$

$Node \triangleq 0..N-1$

VARIABLES $active, termDetected$

$TypeOK \triangleq active \in [Node \to \text{BOOLEAN}] \land termDetected \in \text{BOOLEAN}$

$vars \triangleq \langle active, termDetected \rangle$

$terminated \triangleq \forall n \in Node : active[n] = \text{FALSE}$

- Declaration of constant parameters and of variables (untyped)
- Definition of some basic operators

Modeling Termination Detection in TLA$^+$: State Machine

- Initial condition

$$\begin{aligned} Init \triangleq\ & \land\ active \in [Node \to \text{BOOLEAN}] \\ & \land\ termDetected \in \{\text{FALSE}, terminated\} \end{aligned}$$

Modeling Termination Detection in TLA$^+$: State Machine

- Initial condition

$$\begin{aligned} \mathit{Init} \triangleq\ & \land\ \mathit{active} \in [\mathit{Node} \to \text{BOOLEAN}] \\ & \land\ \mathit{termDetected} \in \{\text{FALSE}, \mathit{terminated}\} \end{aligned}$$

- State transitions

$$\begin{aligned} \mathit{Terminate}(i) \triangleq\ & \land\ \mathit{active}[i] \land \mathit{active}' = [\mathit{active} \text{ EXCEPT } ![i] = \text{FALSE}] \\ & \land\ \mathit{termDetected}' \in \{\mathit{termDetected}, \mathit{terminated}'\} \\ \mathit{Message}(i,j) \triangleq\ & \land\ \mathit{active}[i] \land \mathit{active}' = [\mathit{active} \text{ EXCEPT } ![j] = \text{TRUE}] \\ & \land\ \text{UNCHANGED } \mathit{termDetected} \\ \mathit{SignalTerminated} \triangleq\ & \land\ \mathit{terminated} \land \mathit{termDetected}' = \text{TRUE} \\ & \land\ \text{UNCHANGED } \mathit{active} \end{aligned}$$

Modeling Termination Detection in TLA$^+$: State Machine

- Initial condition

$$\begin{aligned} Init \triangleq\ & \land\ active \in [Node \to \text{BOOLEAN}] \\ & \land\ termDetected \in \{\text{FALSE}, terminated\} \end{aligned}$$

- State transitions

$$\begin{aligned} Terminate(i) \triangleq\ & \land\ active[i] \land active' = [active \text{ EXCEPT } ![i] = \text{FALSE}] \\ & \land\ termDetected' \in \{termDetected, terminated'\} \\ Message(i,j) \triangleq\ & \land\ active[i] \land active' = [active \text{ EXCEPT } ![j] = \text{TRUE}] \\ & \land\ \text{UNCHANGED } termDetected \\ SignalTerminated \triangleq\ & \land\ terminated \land termDetected' = \text{TRUE} \\ & \land\ \text{UNCHANGED } active \end{aligned}$$

- Overall specification

$$\begin{aligned} Next \triangleq\ & \exists i, j \in Node : Terminate(i) \lor SendMsg(i,j) \lor SignalTerminated \\ Spec \triangleq\ & Init \land \Box[Next]_{vars} \land \text{WF}_{vars}(SignalTerminated) \end{aligned}$$

Expressing and Checking Correctness Properties

- Safety properties: "nothing bad ever happens"
  - type correctness $Spec \Rightarrow \Box TypeOK$
  - correct termination detection $Spec \Rightarrow \Box(termDetected \Rightarrow terminated)$
  - quiescence of the system $Spec \Rightarrow \Box(terminated \Rightarrow \Box terminated)$

Expressing and Checking Correctness Properties

- Safety properties: "nothing bad ever happens"
  - type correctness $Spec \Rightarrow \Box TypeOK$
  - correct termination detection $Spec \Rightarrow \Box(termDetected \Rightarrow terminated)$
  - quiescence of the system $Spec \Rightarrow \Box(terminated \Rightarrow \Box terminated)$
- Liveness properties: "something good happens eventually"
  - eventual termination detection $Spec \Rightarrow \Box(terminated \Rightarrow \Diamond termDetected)$

Expressing and Checking Correctness Properties

- Safety properties: "nothing bad ever happens"
  - type correctness $Spec \Rightarrow \Box TypeOK$
  - correct termination detection $Spec \Rightarrow \Box(termDetected \Rightarrow terminated)$
  - quiescence of the system $Spec \Rightarrow \Box(terminated \Rightarrow \Box terminated)$
- Liveness properties: "something good happens eventually"
  - eventual termination detection $Spec \Rightarrow \Box(terminated \Rightarrow \Diamond termDetected)$
- For finite instances, these properties can be verified using TLC
  - safety properties can also be checked using symbolic model checker Apalache

Using TLAPS to Prove Safety Properties

- TLAPS: proof assistant for verifying TLA$^+$ specifications
  - not limited to fixed instances, unlike model checkers
  - relies on user interaction to guide verification

Using TLAPS to Prove Safety Properties

- TLAPS: proof assistant for verifying TLA$^+$ specifications
  - not limited to fixed instances, unlike model checkers
  - relies on user interaction to guide verification
- Proving a simple invariant in TLAPS

THEOREM $TypeCorrect \triangleq Spec \Rightarrow \Box TypeOK$
$\langle 1 \rangle 1.\ Init \Rightarrow TypeOK$
$\langle 1 \rangle 2.\ TypeOK \wedge [Next]_{vars} \Rightarrow TypeOK'$
$\langle 1 \rangle 3.$ QED BY $\langle 1 \rangle 1, \langle 1 \rangle 2, PTL$ DEF $Spec$

  - hierarchical proof language represents proof tree
  - steps can be proved in any order: usually start with QED step
  - theorem follows from steps $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$ by propositional temporal logic

Using TLAPS to Prove Safety Properties

- TLAPS: proof assistant for verifying TLA$^+$ specifications
  - not limited to fixed instances, unlike model checkers
  - relies on user interaction to guide verification
- Proving a simple invariant in TLAPS

THEOREM $TypeCorrect \triangleq Spec \Rightarrow \Box TypeOK$

$\langle 1 \rangle 1$. $Init \Rightarrow TypeOK$	BY DEF $TypeOK$, $Init$, $terminated$
$\langle 1 \rangle 2$. $TypeOK \wedge [Next]_{vars} \Rightarrow TypeOK'$	BY DEF $TypeOK$, $Next$, $vars$, ...
$\langle 1 \rangle 3$. QED	BY $\langle 1 \rangle 1, \langle 1 \rangle 2, PTL$ DEF $Spec$

  - hierarchical proof language represents proof tree
  - steps can be proved in any order: usually start with QED step
  - theorem follows from steps $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$ by propositional temporal logic
  - steps $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$ are proved by expanding definitions

Proofs of Further Safety Properties

- The other safety properties are proved similarly, relying on type correctness

THEOREM $CorrectDetection \triangleq Spec \Rightarrow TDCorrect$

$\langle 1 \rangle 1.\ Init \Rightarrow TDCorrect$	BY DEF *Init*, *TDCorrect*
$\langle 1 \rangle 2.\ TypeOK \wedge TDCorrect \wedge [Next]_{vars} \Rightarrow TDCorrect'$	BY DEF *TDCorrect*, *Next*, *vars*, ...
$\langle 1 \rangle 3.$ QED	BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, *TypeCorrect*, *PTL DEF Spec*

THEOREM $Quiescent \triangleq Spec \Rightarrow Quiescence$

$\langle 1 \rangle.\ TypeOK \wedge [Next]_{vars} \Rightarrow (terminated \Rightarrow terminated')$	BY DEF *TypeOK*, *terminated*, *Next*, *vars*, ...
$\langle 1 \rangle.$ QED	BY *TypeCorrect*, *PTL* DEF *Spec*, *Quiescence*

Proofs of Further Safety Properties

- The other safety properties are proved similarly, relying on type correctness

THEOREM $CorrectDetection \triangleq Spec \Rightarrow TDCorrect$

$\langle 1 \rangle 1.\ Init \Rightarrow TDCorrect$	BY DEF *Init*, *TDCorrect*
$\langle 1 \rangle 2.\ TypeOK \wedge TDCorrect \wedge [Next]_{vars} \Rightarrow TDCorrect'$	BY DEF *TDCorrect*, *Next*, *vars*, ...
$\langle 1 \rangle 3.$ QED	BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, *TypeCorrect*, *PTL DEF Spec*

THEOREM $Quiescent \triangleq Spec \Rightarrow Quiescence$

$\langle 1 \rangle.\ TypeOK \wedge [Next]_{vars} \Rightarrow (terminated \Rightarrow terminated')$	BY DEF *TypeOK*, *terminated*, *Next*, *vars*, ...
$\langle 1 \rangle.$ QED	BY *TypeCorrect*, *PTL* DEF *Spec*, *Quiescence*

- Explicit invocation of definitions and facts
- Hierarchical proofs when brute-force expansion fails
- Minimal use of temporal logic

Fairness Conditions and Enabledness of Actions

- Liveness properties depend on fairness hypotheses
  - TLA$^+$ specifications $Init \wedge \Box[Next]_{vars}$ allow for stuttering steps
  - fairness conditions rule out infinite stuttering

    $$\mathrm{WF}_{vars}(A) \triangleq \Box\big((\Box\mathrm{ENABLED}\ \langle A\rangle_{vars}) \Rightarrow \Diamond\langle A\rangle_{vars}\big)$$

  - $\mathrm{ENABLED}\ A \triangleq \exists vars' : A$ characterizes states in which an A step is possible

Fairness Conditions and Enabledness of Actions

- Liveness properties depend on fairness hypotheses
  - TLA$^+$ specifications $Init \land \Box[Next]_{vars}$ allow for stuttering steps
  - fairness conditions rule out infinite stuttering

$$\mathrm{WF}_{vars}(A) \triangleq \Box\big((\Box\mathrm{ENABLED}\ \langle A\rangle_{vars}) \Rightarrow \Diamond\langle A\rangle_{vars}\big)$$

  - $\mathrm{ENABLED}\ A \triangleq \exists vars' : A$ characterizes states in which an A step is possible

- Reduce enabledness conditions to simple state predicates

LEMMA *EnabledST* $\triangleq$ ASSUME *TypeOK*
PROVE $(\mathrm{ENABLED}\ \langle SignalTerminated\rangle_{vars}) \equiv terminated \land \neg termDetected$
BY *ExpandENABLED* DEF *TypeOK*, *SignalTerminated*, *vars*, *terminated*

Proving Liveness

- Establishing liveness is now easy

THEOREM $Liveness \triangleq Spec \Rightarrow \Box(terminated \Rightarrow \Diamond termDetected)$
$\langle 1 \rangle 1$. DEFINE $P \triangleq terminated \wedge \neg termDetected \quad Q \triangleq termDetected$
$\langle 1 \rangle 2$. $TypeOK \wedge P \wedge [Next]_{vars} \Rightarrow P' \vee Q'$ BY DEF $TypeOK$, $Next$, $vars$, ...
$\langle 1 \rangle 3$. $TypeOK \wedge P \wedge \langle SignalTerminated \rangle_{vars} \Rightarrow Q'$ BY DEF $TypeOK$, $SignalTerminated$
$\langle 1 \rangle 4$. $TypeOK \wedge P \Rightarrow$ ENABLED $\langle SignalTerminated \rangle_{vars}$ BY $EnabledST$
$\langle 1 \rangle 5$. QED BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, PTL DEF $Spec$

  - steps $\langle 1 \rangle 2$ and $\langle 1 \rangle 3$ require standard action-level reasoning
  - step $\langle 1 \rangle 4$ is an immediate consequence of lemma $EnabledST$

Proving Liveness

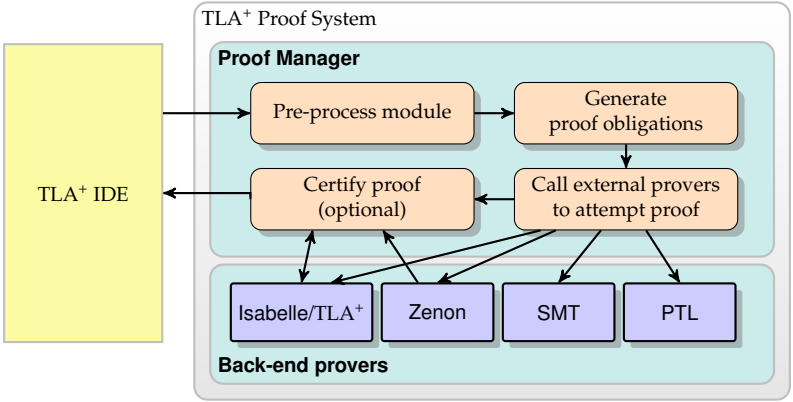
- Establishing liveness is now easy

THEOREM $Liveness \triangleq Spec \Rightarrow \Box(terminated \Rightarrow \Diamond termDetected)$
$\langle 1 \rangle 1$. DEFINE $P \triangleq terminated \wedge \neg termDetected \quad Q \triangleq termDetected$
$\langle 1 \rangle 2$. $TypeOK \wedge P \wedge [Next]_{vars} \Rightarrow P' \vee Q'$ BY DEF $TypeOK$, $Next$, $vars$, ...
$\langle 1 \rangle 3$. $TypeOK \wedge P \wedge \langle SignalTerminated \rangle_{vars} \Rightarrow Q'$ BY DEF $TypeOK$, $SignalTerminated$
$\langle 1 \rangle 4$. $TypeOK \wedge P \Rightarrow$ ENABLED $\langle SignalTerminated \rangle_{vars}$ BY $EnabledST$
$\langle 1 \rangle 5$. QED BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, PTL DEF $Spec$

  - steps $\langle 1 \rangle 2$ and $\langle 1 \rangle 3$ require standard action-level reasoning
  - step $\langle 1 \rangle 4$ is an immediate consequence of lemma $EnabledST$

- Propositional temporal reasoning is again enough here
  - first-order temporal logic is required when reasoning about well-founded orders

TLAPS Architecture



- Isabelle/TLA$^+$: faithful encoding of TLA$^+$ in Isabelle's meta-logic
- PTL: decision procedure for propositional temporal logic

Outline

Untyped Logic: Boolean Expressions

- TLA$^+$ is untyped: no distinction between terms and formulas

  $(42 = \text{TRUE}) \land$ "abc" denotes some (undetermined) value

Untyped Logic: Boolean Expressions

- TLA$^+$ is untyped: no distinction between terms and formulas

  $(42 = \text{TRUE}) \land \text{“abc”}$ denotes some (undetermined) value

- Two-valued semantics with underspecification
  - operators such as $=, \in, \land, \forall$ always evaluate to TRUE or FALSE
  - $(p = q) \Rightarrow (p \Leftrightarrow q)$ holds, but $(p \Leftrightarrow q) \Rightarrow (p = q)$ does not

Untyped Logic: Boolean Expressions

- TLA$^+$ is untyped: no distinction between terms and formulas

  $(42 = \text{TRUE}) \land \text{“abc”}$ denotes some (undetermined) value

- Two-valued semantics with underspecification
  - operators such as $=, \in, \land, \forall$ always evaluate to TRUE or FALSE
  - $(p = q) \Rightarrow (p \Leftrightarrow q)$ holds, but $(p \Leftrightarrow q) \Rightarrow (p = q)$ does not

- Most standard laws of logic remain true

$$
\begin{array}{lll}
(\neg P) = (P \Rightarrow \text{FALSE}) & \neg(P \land Q) = (\neg P \lor \neg Q) & \mathit{boolify}(P) \triangleq P = \text{TRUE} \\
(P \land \text{TRUE}) = \mathit{boolify}(P) & \mathit{boolify}(x = y) = (x = y) & \mathit{boolify}(Q \land R) = (Q \land R) \\
P(t) \Rightarrow (\exists x : P(x)) & (\forall x : P(x)) \Rightarrow P(x) &
\end{array}
$$

Encoding Set Theory

- Reduce set-theoretic constructions to first-order logic
  - define set-theoretic operators in terms of uninterpreted binary predicate $\in$

$$
\begin{aligned}
& e \in (S \cup T) \;\equiv\; (e \in S) \vee (e \in T) \\
& S \subseteq T \;\equiv\; \forall x \in S : x \in T \\
& e \in \{\, x \in S : P(x) \,\} \;\equiv\; (e \in S) \wedge P(e) \\
& e \in \{\, f(x) : x \in S \,\} \;\equiv\; \exists x \in S : e = f(x)
\end{aligned}
$$

Encoding Set Theory

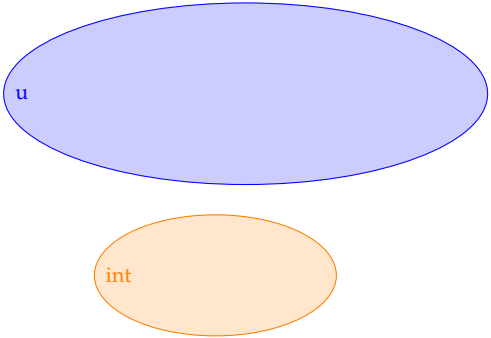
- Reduce set-theoretic constructions to first-order logic
  - define set-theoretic operators in terms of uninterpreted binary predicate $\in$

$$
\begin{aligned}
& e \in (S \cup T) \;\equiv\; (e \in S) \vee (e \in T) \\
& S \subseteq T \;\equiv\; \forall x \in S : x \in T \\
& e \in \{\, x \in S : P(x) \,\} \;\equiv\; (e \in S) \wedge P(e) \\
& e \in \{\, f(x) : x \in S \,\} \;\equiv\; \exists x \in S : e = f(x)
\end{aligned}
$$

- Two implementations of this encoding
  - first implementation relies on extensive rewriting
  - second implementation uses axioms with well-chosen triggers
  - no significant performance difference, but rewriting is brittle
- TLA$^+$ functions encoded in a similar way

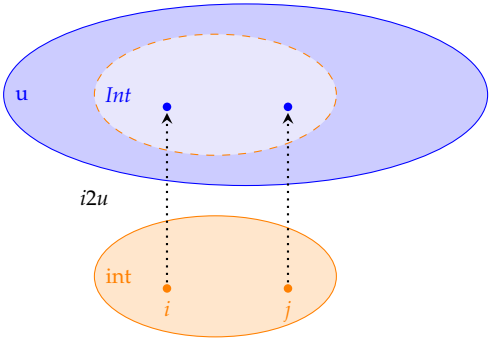
Untyped Logic: Theory Reasoning

- SMT solvers provide automation for interpreted theories
- Untyped embedding: inject interpreted sorts into TLA$^+$ universe



Untyped Logic: Theory Reasoning

- SMT solvers provide automation for interpreted theories
- Untyped embedding: inject interpreted sorts into TLA$^+$ universe



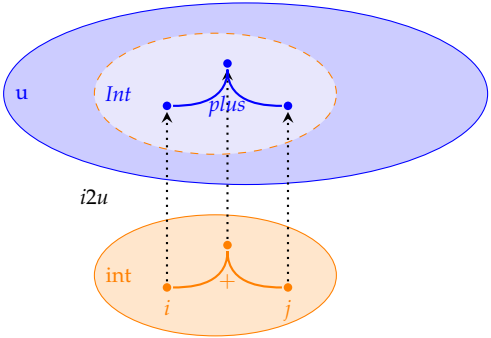
Characteristic axioms

$\forall i, j : i2u(i) = i2u(j) \Rightarrow i = j$

$\forall u : u \in Int \equiv \exists i : u = i2u(i)$

Untyped Logic: Theory Reasoning

- SMT solvers provide automation for interpreted theories
- Untyped embedding: inject interpreted sorts into TLA$^+$ universe



Characteristic axioms

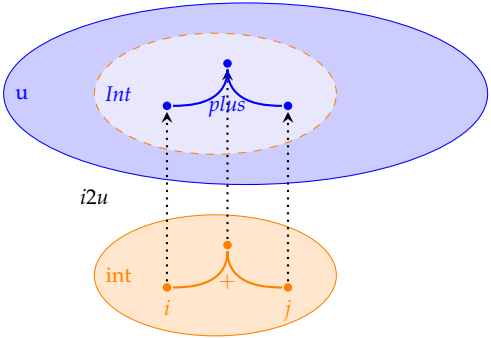
$\forall i, j : i2u(i) = i2u(j) \Rightarrow i = j$

$\forall u : u \in Int \equiv \exists i : u = i2u(i)$

$\forall i, j : plus(i2u(i), i2u(j)) = i2u(i + j)$

Untyped Logic: Theory Reasoning

- SMT solvers provide automation for interpreted theories
- Untyped embedding: inject interpreted sorts into TLA$^+$ universe



Characteristic axioms

$\forall i, j : i2u(i) = i2u(j) \Rightarrow i = j$

$\forall u : u \in Int \equiv \exists i : u = i2u(i)$

$\forall i, j : plus(i2u(i), i2u(j)) = i2u(i + j)$

- Use of triggers again makes this work in practice

Support for Temporal Logic Reasoning

- Temporal logic breaks natural deduction
  - $F \vdash G$ cannot be identified with $\vdash F \Rightarrow G$
  - for example, have $F \vdash \Box F$ but not $\vdash F \Rightarrow \Box F$

Support for Temporal Logic Reasoning

- Temporal logic breaks natural deduction
  - $F \vdash G$ cannot be identified with $\vdash F \Rightarrow G$
  - for example, have $F \vdash \Box F$ but not $\vdash F \Rightarrow \Box F$
  - But: $\Box F \vdash G$ can be identified with $\vdash \Box F \Rightarrow G$

Support for Temporal Logic Reasoning

- Temporal logic breaks natural deduction
  - $F \vdash G$ cannot be identified with $\vdash F \Rightarrow G$
  - for example, have $F \vdash \Box F$ but not $\vdash F \Rightarrow \Box F$
  - But: $\Box F \vdash G$ can be identified with $\vdash \Box F \Rightarrow G$

- Arrange temporal reasoning so that all hypotheses are boxed
  - formula F is boxed if $F \Leftrightarrow \Box F$
  - syntactic approximation: constant formulas, $\Box F$, $\Diamond \Box F$, $\mathrm{WF}_v(A)$, conjunction, ...
  - implicit generalization of formulas derived in boxed context
  - requires disciplined, but quite natural use of temporal reasoning

Temporal Logic Reasoning in Practice

- Reduce temporal conclusions to action-level hypotheses

$$\frac{P \wedge [N]_v \Rightarrow P' \vee Q' \quad P \wedge \langle A \rangle_v \Rightarrow Q' \quad P \Rightarrow \text{ENABLED } \langle A \rangle_v}{\Box[N]_v \wedge \text{WF}_v(A) \;\Rightarrow\; \Box(P \Rightarrow \Diamond Q)}$$

- Temporal reasoning is mostly propositional
  - use PTL decision procedure rather than hard-wired rules

Temporal Logic Reasoning in Practice

- Reduce temporal conclusions to action-level hypotheses

$$\frac{P \wedge [N]_v \Rightarrow P' \vee Q' \quad P \wedge \langle A \rangle_v \Rightarrow Q' \quad P \Rightarrow \text{ENABLED } \langle A \rangle_v}{\Box[N]_v \wedge \text{WF}_v(A) \;\Rightarrow\; \Box(P \Rightarrow \Diamond Q)}$$

- Temporal reasoning is mostly propositional
  - use PTL decision procedure rather than hard-wired rules
- Support for first-order temporal reasoning by on-the-fly abstraction
  - hide operators that are not part of PTL, and vice versa
  - during pre-processing, commute $\forall$ and $\Box$ as well as $\exists$ and $\Diamond$

First-Order Temporal Reasoning

THEOREM $Init \wedge \Box[Next]_v \Rightarrow \forall p \in Proc : \Box Safe(p)$
$\langle 1 \rangle$. SUFFICES ASSUME NEW $p \in Proc$
PROVE $Init \wedge \Box[Next]_v \Rightarrow \Box Safe(p)$
OBVIOUS
$\langle 1 \rangle 1$. $Init \Rightarrow Safe(p)$ BY DEF $Init$, $Safe$, ...
$\langle 1 \rangle 2$. $Safe(p) \wedge [Next]_v \Rightarrow Safe(p)'$ BY DEF $Safe$, $Next$, v, ...
$\langle 1 \rangle 3$. QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, PTL

- Mix of first-order and temporal reasoning
  - first-order provers vs. PTL decision procedure
  - prime "modality" handled by pre-processing at action level
- What is really going on here?

Coalescing: Basic Idea

- Abstract subformulas from different sublogic
  - in the SUFFICES step, the FOL prover sees the proof obligation

$$\frac{p \in Proc \qquad Init \land \boxed{\Box[Next]_v} \Rightarrow \boxed{\Box Safe}(p)}{Init \land \boxed{\Box[Next]_v} \Rightarrow \forall p \in Proc : \boxed{\Box Safe}(p)}$$

Coalescing: Basic Idea

- Abstract subformulas from different sublogic
  - in the SUFFICES step, the FOL prover sees the proof obligation

$$\frac{p \in Proc \qquad Init \wedge \boxed{\Box[Next]_v} \Rightarrow \boxed{\Box Safe}(p)}{Init \wedge \boxed{\Box[Next]_v} \Rightarrow \forall p \in Proc : \boxed{\Box Safe}(p)}$$

  - in the QED step, the PTL decision procedure sees

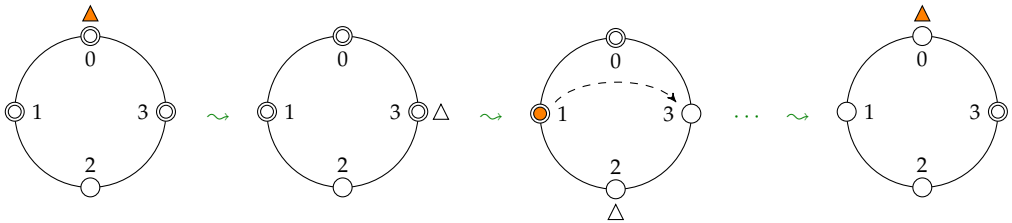
$$\frac{\boxed{Init} \Rightarrow \boxed{Safe(p)} \qquad \boxed{Safe(p)} \wedge \boxed{[Next]_v} \Rightarrow \circ \boxed{Safe(p)}}{\boxed{Init} \wedge \Box \boxed{[Next]_v} \Rightarrow \Box \boxed{Safe(p)}}$$

  - the formulas in boxes are introduced as ad-hoc operators
- Sound combination of temporal and first-order reasoning

Outline

Overall Idea of Dijkstra's Algorithm (EWD 840, 1983)

- Token circulates on the ring of nodes



  - a node that has locally terminated passes the token to its neighbor
  - when a node sends a message to a higher-numbered node, it becomes "stained"
  - passing token cleans the node but collects the "stain"

- Condition for detecting termination
  - master node is inactive and clean and holds clean token

Specifying and Verifying Dijkstra's Algorithm

- Modeling the algorithm in TLA$^+$ is a straightforward exercise

Specifying and Verifying Dijkstra's Algorithm

- Modeling the algorithm in TLA$^+$ is a straightforward exercise
- Expected properties: type correctness, safety, quiescence, liveness
  - we can prove these in the same way as for the abstract version

Specifying and Verifying Dijkstra's Algorithm

- Modeling the algorithm in TLA$^+$ is a straightforward exercise
- Expected properties: type correctness, safety, quiescence, liveness
  - we can prove these in the same way as for the abstract version
  - alternatively: prove that EWD840 refines TerminationDetection

$$TD \triangleq \text{INSTANCE } \mathit{TerminationDetection} \text{ WITH } \mathit{termDetected} \leftarrow \mathit{terminationDetected}$$
$$\text{THEOREM } \mathit{Refinement} \triangleq \mathit{Spec} \Rightarrow \mathit{TD}!\mathit{Spec}$$

  - stuttering invariance of TLA formulas is essential for this to work

Specifying and Verifying Dijkstra's Algorithm

- Modeling the algorithm in TLA$^+$ is a straightforward exercise
- Expected properties: type correctness, safety, quiescence, liveness
  - we can prove these in the same way as for the abstract version
  - alternatively: prove that EWD840 refines TerminationDetection

$$TD \triangleq \text{INSTANCE } \mathit{TerminationDetection} \text{ WITH } \mathit{termDetected} \leftarrow \mathit{terminationDetected}$$
$$\text{THEOREM } \mathit{Refinement} \triangleq \mathit{Spec} \Rightarrow \mathit{TD}!\mathit{Spec}$$

  - stuttering invariance of TLA formulas is essential for this to work
- First step: establish Dijkstra's invariant for EWD840

$$\begin{aligned} \mathit{Inv} \triangleq\ & \lor \forall i \in \mathit{Node} : \mathit{tpos} < i \Rightarrow \neg \mathit{active}[i] \\ & \lor \exists j \in 0 \,..\, \mathit{tpos} : \mathit{color}[j] = \text{“orange”} \\ & \lor \mathit{tcolor} = \text{“orange”} \end{aligned}$$

Proving Refinement

- Safety part: we need to prove two facts

  $Init \Rightarrow TD!Init$
  $[Next]_{vars} \Rightarrow [TD!Next]_{TD!vars}$

  - initialization is straightforward

Proving Refinement

- Safety part: we need to prove two facts

  $Init \Rightarrow TD!Init$
  $TypeOK \wedge Inv \wedge [Next]_{vars} \Rightarrow [TD!Next]_{TD!vars}$

  - initialization is straightforward
  - step simulation relies on the already established invariant

Proving Refinement

- Safety part: we need to prove two facts

  $Init \Rightarrow TD!Init$
  $TypeOK \wedge Inv \wedge [Next]_{vars} \Rightarrow [TD!Next]_{TD!vars}$

  - initialization is straightforward
  - step simulation relies on the already established invariant

- Liveness part: $Spec \Rightarrow WF_{TD!vars}(TD!SignalTerminated)$
  - show that $TD!SignalTerminated$ cannot stay enabled forever

Proving Refinement

- Safety part: we need to prove two facts

$$Init \Rightarrow TD!Init$$
$$TypeOK \wedge Inv \wedge [Next]_{vars} \Rightarrow [TD!Next]_{TD!vars}$$

  - initialization is straightforward
  - step simulation relies on the already established invariant

- Liveness part: $Spec \Rightarrow WF_{TD!vars}(TD!SignalTerminated)$
  - show that $TD!SignalTerminated$ cannot stay enabled forever
  - algorithm may require 3 rounds of the token after global termination
  - (i) bring token back to node 0, (ii) clean all nodes, (iii) establish *terminationDetected*

Proving Refinement

- Safety part: we need to prove two facts

  $Init \Rightarrow TD!Init$
  $TypeOK \wedge Inv \wedge [Next]_{vars} \Rightarrow [TD!Next]_{TD!vars}$

  - initialization is straightforward
  - step simulation relies on the already established invariant

- Liveness part: $Spec \Rightarrow WF_{TD!vars}(TD!SignalTerminated)$
  - show that $TD!SignalTerminated$ cannot stay enabled forever
  - algorithm may require 3 rounds of the token after global termination
  - (i) bring token back to node 0, (ii) clean all nodes, (iii) establish *terminationDetected*
  - standard liveness proof for EWD840 specification

Outline

Summing Up

- TLAPS: Proof support for TLA$^+$
  - focus on automation and usability, not foundational purity
  - declarative proof language allows engineers to decompose the overall proof
  - extensible set of automated back-end reasoners
  - adapt backend reasoners to the language, not the other way round
  - sound integration of predicate logic and modal/temporal reasoning
- Ongoing work
  - better automation for data structures (sequences, bags, ...)
  - help with discovering useful facts and expanding definitions
  - improved IDE support (proof decomposition, refactoring)
  - certification of SMT proofs in foundational proof assistants