Proving Possibility Properties

Leslie Lamport

Digital Equipment Corporation, 130 Lytton Avenue, Palo Alto, California 94303, USA

Abstract

A method is described for proving "always possibly" properties of specifications in formalisms with linear-time trace semantics. It is shown to be relatively complete for TLA (Temporal Logic of Actions) specifications.

Key words: Branching time, linear time, temporal logic.

1 Introduction

Does proving possibility properties provide any useful information about a system? Why prove that it is possible for a user to press q on the keyboard and for a q subsequently to appear on the screen? We know that the user can always press the q key, and what good is knowing that a q *might* appear on the screen? Isn't it enough to prove that no q appears on the screen unless a q is typed (a safety property), and that, if a q is typed, then a q eventually does appear (a liveness property)?

Although possibility properties may tell us nothing about a system, we do not reason about a system; we reason about a mathematical model of a system. A possibility property can provide a sanity check on our model. Proving that it is always possible for a $press(q)$ action to occur tells us something useful about the model. In general, we want to prove that a model allows the occurrence of actions representing events that the system cannot prevent.

We present a method for proving that it is always possible for some state or action eventually to occur. This is the simplest class of possibility properties and seems to be the most useful. (The simpler requirement that it is always possible for an action to occur may also be useful, but it just asserts that the action is always enabled, so it is a safety property and not a possibility property.) We first describe the general approach, which applies to any formalism with a linear-time semantics. We then show how the method is used with

TLA, the Temporal Logic of Actions [8], and prove a relative completeness result.

Possibility properties pose no problem in formalisms based on branching-time semantics [4]. However, it is impossible to assert in linear-time temporal logic that something is always possible [6]. It is therefore not obvious how to prove possibility properties in the formalisms that we consider, which are based on linear-time semantics.

We are concerned with proofs, not finite-state model checking. Model checking begins by writing (or rewriting) a specification as a transition system. A finite-state linear-time specification should yield the same transition system as the corresponding branching-time specification, and hence the same model checking algorithm.

2 Possibility and Closure

2.1 Closure and Safety

We begin by reviewing some basic concepts of linear-time temporal logic [10]. A behavior is an infinite sequence of states or of events—for now, it doesn't matter which. The meaning $[\![\Pi]\!]$ of a temporal-logic formula Π is a Boolean-valued function on behaviors. We say that the behavior σ satisfies Π iff (if and only if) $[\![\Pi]\!](\sigma)$ equals $\textsc{true}$. Formula Π is valid, written $\models \Pi$, iff every behavior satisfies Π. To use temporal logic to specify (a mathematical model of) a system, we consider states to represent possible system states and events to represent possible system actions, so a behavior represents a conceivable execution of a system. A system is specified by a formula Π that is satisfied by precisely those behaviors that represent a legal system execution.

Boolean operations on formulas are defined in the obvious way; for example, $[\![\Pi \wedge \Phi]\!](\sigma) \triangleq [\![\Pi]\!](\sigma) \wedge [\![\Phi]\!](\sigma)$. We define $\Box\Pi$ to be the formula that is satisfied by a behavior σ iff every suffix of σ satisfies Π, and we define $\Diamond\Pi$ to be satisfied by σ iff some suffix of σ satisfies Π. The operators $\Box$ and $\Diamond$ are read *always* and *eventually*, respectively. We define $\leadsto$ by $\Pi \leadsto \Phi \;\triangleq\; \Box(\Pi \Rightarrow \Diamond\Phi)$.

Let $\mathbf{S}^\infty$ be the set of all behaviors, let $\mathbf{S}^*$ be the set of all finite behaviors (finite prefixes of elements of $\mathbf{S}^\infty$), let "$\cdot$" be concatenation of sequences, and let $\rho \sqsubset \sigma$ mean that ρ is a nonempty finite prefix of the behavior σ. The *closure* $\mathcal{C}(\Pi)$ of a formula Π is defined by

$$[\![\mathcal{C}(\Pi)]\!](\sigma) \;\triangleq\; \forall \rho \sqsubset \sigma \,:\, \exists \tau \in \mathbf{S}^\infty \,:\, [\![\Pi]\!](\rho \cdot \tau) \tag{1}$$

where $\forall \rho \sqsubset \sigma$ is universal quantification over all finite prefixes ρ of σ. Thus, a behavior σ satisfies $\mathcal{C}(\Pi)$ iff every finite prefix of σ can be extended to a behavior that satisfies Π. The following proposition follows easily from (1).

Proposition 1 *For any formulas* Π *and* Φ*:*

(1) $\models \Pi \Rightarrow \mathcal{C}(\Pi)$
(2) $\models \Pi \Rightarrow \Phi$ *implies* $\models \mathcal{C}(\Pi) \Rightarrow \mathcal{C}(\Phi)$

A *safety* formula is one that equals its closure. Thus, a safety formula Π is satisfied by a behavior σ iff every prefix of σ can be extended to a behavior satisfying Π. Intuitively, a safety property Π constrains only the finite behavior of a system—any behavior that fails to satisfy Π fails at some specific instant. More precisely, Π is a safety property (equals its closure) iff

$$\forall \sigma \in \mathbf{S}^{\infty} \; : \; [\![\neg\Pi]\!](\sigma) \;\equiv\; \exists \rho \sqsubset \sigma \; : \; \forall \tau \in \mathbf{S}^{\infty} \; : \; [\![\neg\Pi]\!](\rho \cdot \tau) \qquad (2)$$

2.2 Possibility

We now define a class of possibility properties and relate them to closure. The properties are of the form *always possibly* P, meaning that at all times during an execution of the system, it is possible for P eventually to become true. In linear-time temporal logic, it is impossible to write a formula whose meaning is always possibly P [6]. However, for any particular system, we can write a formula asserting that always possibly P holds for behaviors of that system. More precisely, we can define a formula $\mathbf{P}_{\Pi}(P)$ such that always possibly P holds for the system specified by Π iff $\mathbf{P}_{\Pi}(P)$ is valid.

Intuitively, always possibly P holds for a system iff, at any point during any execution of the system, it is possible to choose some particular way of continuing the execution that makes P eventually hold. In other words, if ρ is the prefix of a behavior satisfying the system's specification Π, then there exists a behavior τ such that $\rho \cdot \tau$ satisfies Π, and P holds at some point in τ. We can therefore define $\mathbf{P}_{\Pi}(P)$ by

$$[\![\mathbf{P}_{\Pi}(P)]\!](\sigma) \;\triangleq\; [\![\Pi]\!](\sigma) \Rightarrow \forall \rho \sqsubset \sigma \; : \; \exists \tau \; : \; [\![\Pi]\!](\rho \cdot \tau) \wedge [\![\Diamond P]\!](\tau) \qquad (3)$$

Our method of proving possibility properties is based on the following result. It and all subsequent propositions are proved in the appendix.

Proposition 2 *If* $\neg P$ *is a safety property, then*

$$\models \; (\mathcal{C}(\Pi) \Rightarrow \mathcal{C}(\mathcal{C}(\Pi) \wedge \Box\Diamond P)) \;\Rightarrow\; \mathbf{P}_{\Pi}(P)$$

We will use this result when $[\![P]\!](\sigma)$ depends only on the first one or two elements of σ. By (2), $\neg P$ is a safety property for such a P.

3 Proving Possibility Properties in TLA

3.1 TLA

To apply Proposition 2, we need to compute closures. One can write TLA specifications in a way that makes computing the closure easy. We now give a thumbnail review of TLA; see [8] for a real explanation of the logic.

In TLA, behaviors are infinite sequences of states, where a *state* is an assignment of variables to values. We let $\mathbf{S}$ be the set of all states. Formulas are built from actions, Boolean operators, and the temporal operator $\Box$. An *action* is a Boolean expression containing primed and unprimed variables. For states s and t, we define $[\![A]\!](s,t)$ to equal TRUE iff A holds with values from s substituted for unprimed variables and with values from t substituted for primed variables. We consider action A to be a temporal formula by letting $[\![A]\!](s_0, s_1, s_2, \ldots)$ equal $[\![A]\!](s_0, s_1)$.

A *state predicate* P is an action with no primed variables; we write $[\![P]\!](s)$ instead of $[\![P]\!](s,t)$, which is independent of t. For an action A, we define the predicate $\textsc{Enabled}\, A$ by $[\![\textsc{Enabled}\, A]\!](s) \triangleq \exists t \in \mathbf{S} : [\![A]\!](s,t)$. A *state function* is a nonBoolean expression containing no primed variables. For any state function v, we let $[A]_v \triangleq A \vee (v' = v)$ and $\langle A \rangle_v \triangleq A \wedge (v' \neq v)$, where v' is the expression obtained by priming the free variables in v.

The canonical form of a TLA formula is $\mathit{Init} \wedge \Box[N]_v \wedge F$, where Init is a state predicate, N an action, v a state function, and F the conjunction of formulas of the form $\mathrm{WF}_v(A)$ (weak fairness) or $\mathrm{SF}_v(A)$ (strong fairness), with

$$
\begin{aligned}
\mathrm{WF}_v(A) &\triangleq \Box\Diamond\neg\textsc{Enabled}\,\langle A \rangle_v \vee \Box\Diamond\langle A \rangle_v \\
\mathrm{SF}_v(A) &\triangleq \Diamond\Box\neg\textsc{Enabled}\,\langle A \rangle_v \vee \Box\Diamond\langle A \rangle_v
\end{aligned}
$$

For example, a system that starts with x and y both 0, and repeatedly either increments x by ± 1 or, if x equals 0, increments y by ± 1, is specified by the following formula Πxy.[1]

$$\begin{array}{rcl} Nxy & \triangleq & \lor \land x' \in \{x+1, x-1\} \\ & & \quad \land y' = y \\ & & \lor \land x = x' = 0 \\ & & \quad \land y' \in \{y+1, y-1\} \\ \Pi xy & \triangleq & (x = y = 0) \land \Box[Nxy]_{\langle x,y \rangle} \land \mathrm{WF}_{\langle x,y \rangle}(Nxy) \end{array}$$

The fairness condition $\mathrm{WF}_{\langle x,y \rangle}(Nxy)$ asserts that the system never stops.

TLA also has an operator $\exists$, where $\exists x : \Pi$ is essentially Π with variable x hidden. The system specified by $\exists x : \Pi$ satisfies a possibility property iff Π does—assuming x does not occur free in the property—so we ignore the $\exists$ operator here. Using $\exists$, we can express $\mathbf{P}_{\Pi}(P)$ and $\mathcal{C}(\Pi)$ as TLA formulas, for any formulas Π and P. Propositions 1 and 2 can then be proved by temporal-logic reasoning.

Closures of TLA formulas are computed using the following result.

Proposition 3 *If Init is a state predicate, M and N are actions such that M implies N, and F is the conjunction of countably many formulas of the form* $\mathrm{WF}_v(A)$ *and/or* $\mathrm{SF}_v(A)$, *where each $\langle A \rangle_v$ implies M, then*

$$\mathcal{C}(\mathit{Init} \land \Box[N]_v \land \Diamond\Box[M]_v \land F) \equiv \mathit{Init} \land \Box[N]_v$$

Since $\Box\Pi$ implies $\Diamond\Box\Pi$, for any Π, substituting Nxy for both N and M in the proposition proves that $\mathcal{C}(\Pi xy) \equiv (x = y = 0) \land \Box[Nxy]_{\langle x,y \rangle}$. For $M = N$, Proposition 3 is a special case of Proposition 2 of [1].

A formula of the form $\mathit{Init} \land \Box[N]_v \land F$ is called *machine closed* [1] if its closure equals $\mathit{Init} \land \Box[N]_v$. Proposition 3 implies that such a formula is machine closed if F is the conjunction of fairness conditions for actions that imply N. Machine closure means that F does not rule out any finite prefixes of behaviors. It can be argued that any specification that models a real implementation should be machine closed, and that possibility properties need be proved only for a model of an implementation, not for a high-level specification.

[1] A list of formulas bulleted with $\land$ or $\lor$ denotes the conjunction or disjunction of the formulas; indentation is used to eliminate parentheses. Angle brackets enclose tuples.

3.2 The Proof Method

We now show how to use Propositions 1, 2, and 3 to prove possibility properties of the form $\mathbf{P}_{\Pi}(P)$ for a state predicate P, where Π equals $Init \wedge \Box[N]_v \wedge F$, and $\mathcal{C}(\Pi)$ equals $Init \wedge \Box[N]_v$. For any action A, formula $\mathbf{P}_{\Pi}(A)$ is equivalent to $\mathbf{P}_{\Pi}(\textsc{Enabled}\,([N]_v \wedge A))$. Hence, our method can be used to prove properties $\mathbf{P}_{\Pi}(A)$ for arbitrary actions A.

To prove $\mathbf{P}_{\Pi}(P)$, we find an action M and a conjunction G of fairness properties such that

$$Init \wedge \Box[N]_v \wedge \Diamond\Box[M]_v \wedge G \Rightarrow \Box\Diamond P \tag{4}$$

and for which we can use Proposition 3 to prove

$$\mathcal{C}(Init \wedge \Box[N]_v \wedge \Diamond\Box[M]_v \wedge G) \equiv Init \wedge \Box[N]_v \tag{5}$$

We then deduce $\mathbf{P}_{\Pi}(P)$ as follows.

1. $Init \wedge \Box[N]_v \wedge \Diamond\Box[M]_v \wedge G \Rightarrow Init \wedge \Box[N]_v \wedge \Box\Diamond P$
   Proof: (4).
2. $Init \wedge \Box[N]_v \Rightarrow \mathcal{C}(Init \wedge \Box[N]_v \wedge \Box\Diamond P)$
   Proof: (5) and part 2 of Proposition 1.
3. Q.E.D.
   Proof: By Proposition 2, since $Init \wedge \Box[N]_v \equiv \mathcal{C}(\Pi)$.

For example, to prove $\mathbf{P}_{\Pi xy}(y = 17)$, we take

$$\begin{aligned} M \;\triangleq\;\; & \vee \wedge ((x > 0) \wedge (x' = x - 1)) \vee ((x < 0) \wedge (x' = x + 1)) \\ & \quad \wedge\, y' = y \\ & \vee \wedge\, x = x' = 0 \\ & \quad \wedge ((y > 17) \wedge (y' = y - 1)) \vee ((y < 17) \wedge (y' = y + 1)) \end{aligned}$$

and let G be $\mathrm{WF}_{\langle x,y\rangle}(M)$ To prove (4), we use the TLA rules from Figure 5 (page 888) of [8].

We now show that this proof method is complete relative to non-temporal reasoning about actions. This means that if all the necessary valid action formulas can be proved, then every valid formula $\mathbf{P}_{\Pi}(P)$ is provable. We write $\vdash \Psi$ to mean that formula Ψ is provable from Propositions 1, 2, and 3 and the rules in [8].

Our results assume that valid actions in some class of *expressible* formulas are provable. We assume that expressible terms and formulas are closed under the operations of first-order logic (conjunction, quantification, etc.), priming, forming tuples, and primitive recursive definitions. Relative completeness results for programming logics are generally based on some form of predicate transformer analogous to the *sin* operator of [7]. For any action A and state predicate P, the state predicate $sin(A, P)$ can be defined by

$$\begin{aligned} &[\![sin(A,P)]\!](s) \;\triangleq \\ &\quad \exists s_0, \ldots, s_n \in \mathbf{S} \,:\, (s = s_n) \wedge [\![P]\!](s_0) \wedge (\forall i < n \,:\, [\![A]\!](s_i, s_{i+1})) \end{aligned} \tag{6}$$

for all states s. We first show completeness of the TLA rules for proving invariance properties.

Proposition 4 *For any predicates I and $Init$, state function v, and action N, if*

(1) Every valid expressible action formula is provable.
(2) I, $Init$, v, N, and $sin([N]_v, Init)$ are expressible.
(3) $\models Init \wedge \Box[N]_v \Rightarrow \Box I$

then $\vdash Init \wedge \Box[N]_v \Rightarrow \Box I$.

Proposition 4 is essentially the TLA version of the classical completeness results for Hoare logics [3]. We use it to show completeness of our method for proving possibility properties:

Proposition 5 *If*

(1) Every valid expressible action formula is provable.
(2) P, $Init$, v, N, and $sin([N]_v, Init)$ are expressible.
(3) $\vdash \mathcal{C}(\Pi) \equiv Init \wedge \Box[N]_v$
(4) $\models \mathbf{P}_{\Pi}(P)$

then $\vdash \mathbf{P}_{\Pi}(P)$.

4 Conclusion

Proving possibility properties provides a way of checking that the mathematical models we make of our systems are sensible. For real time specifications, an important possibility property is nonZenoness, which asserts that it is always possible for time to advance. The relation between possibility and closure was first observed for nonZenoness in [1]. Our method generalizes a method described there for proving nonZenoness.

Propositions 1 and 2 are independent of TLA. They can be used for proving possibility properties in any trace-based specification method for which closures can be computed. It is easy to compute closures when specifications are written as certain kinds of transition systems. For example, the closure of (the temporal-logic formula corresponding to) a Büchi automaton [2] with a strongly connected state graph is the automaton obtained by making every state an accepting state. The closure of a specification written as a state transition system [5,9] is obtained by removing the fairness properties, if those properties are expressed as fairness conditions on transitions. We do not know of any practical method for computing the closure of arbitrary temporal-logic formulas, or of transition systems with arbitrary temporal formulas as fairness requirements. We do not know how to prove possibility properties for traditional temporal-logic specifications [10].

Acknowledgement

Martín Abadi and Stephan Merz pointed out mistakes in the text of an earlier version. Fred Schneider suggested some improvements to the presentation.

References

[1] Martín Abadi and Leslie Lamport. An old-fashioned recipe for real time. *ACM Transactions on Programming Languages and Systems*, 16(5):1543–1571, September 1994.

[2] Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.

[3] Krzysztof R. Apt. Ten years of Hoare's logic: A survey—part one. *ACM Transactions on Programming Languages and Systems*, 3(4):431–483, October 1981.

[4] E. Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 16, pages 995–1072. Elsevier and MIT Press, Amsterdam and Cambridge, Massachusetts, 1990.

[5] Simon S. Lam and A. Udaya Shankar. Specifying modules to satisfy interfaces: A state transition system approach. *Distributed Computing*, 6(1):39–63, 1992.

[6] Leslie Lamport. 'Sometime' is sometimes 'not never': A tutorial on the temporal logic of programs. In *Proceedings of the Seventh Annual Symposium on Principles of Programming Languages*, pages 174–185. ACM SIGACT-SIGPLAN, January 1980.

[7] Leslie Lamport. *win* and *sin*: Predicate transformers for concurrency. *ACM Transactions on Programming Languages and Systems*, 12(3):396–428, July 1990.

[8] Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.

[9] Nancy Lynch and Mark Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the Sixth Symposium on the Principles of Distributed Computing*, pages 137–151. ACM, August 1987.

[10] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, New York, 1991.

Appendix

We now prove Propositions 2–5. The proofs use a hierarchical style in which the proof of statement $\langle i \rangle j$ is either an ordinary paragraph-style proof or the sequence of statements $\langle i+1 \rangle 1$, $\langle i+1 \rangle 2$, ... and their proofs. We recommend reading proofs top-down—reading the proof of a level-k step by first reading the level-$(k+1)$ statements that form the proof, together with the proof of the final Q.E.D. step, and then reading the proofs of the level-$(k+1)$ steps in any order.

A.1 Proof of Proposition 2

To prove the proposition, we must prove that if a behavior σ satisfies $\mathcal{C}(\Pi) \Rightarrow \mathcal{C}(\mathcal{C}(\Pi) \wedge \Box\Diamond P)$, then it satisfies $\mathbf{P}_{\Pi}(P)$. By the definition (3) of $\mathbf{P}_{\Pi}(P)$, the proposition is proved as follows.

ASSUME: 1. $[\![\Pi]\!](\sigma)$
2. $[\![\mathcal{C}(\Pi) \Rightarrow \mathcal{C}(\mathcal{C}(\Pi) \wedge \Box\Diamond P)]\!](\sigma)$

PROVE: $\forall \rho \sqsubset \sigma \, : \, \exists \tau \, : \, [\![\Pi]\!](\rho \cdot \tau) \wedge [\![\Diamond P]\!](\tau)$

$\langle 1 \rangle 1.$ $\forall \rho \sqsubset \sigma \, : \, \exists \eta \in \mathbf{S}^{\infty} \, : \, [\![\mathcal{C}(\Pi)]\!](\rho \cdot \eta) \wedge [\![\Box\Diamond P]\!](\rho \cdot \eta)$

$\langle 2 \rangle 1.$ $[\![\mathcal{C}(\Pi)(\sigma)]\!]$

PROOF: Assumption 1 and part 1 of Proposition 1.

$\langle 2 \rangle 2.$ $\mathcal{C}(\mathcal{C}(\Pi) \wedge \Box\Diamond P)(\sigma)$

PROOF: $\langle 2\rangle 1$, assumption 2, and the definition of $\Rightarrow$ for temporal formulas.

$\langle 2\rangle 3$. Q.E.D.

PROOF: $\langle 2\rangle 2$, (1), and the definition of $\wedge$ for temporal formulas.

$\langle 1\rangle 2$. $\forall \rho \sqsubset \sigma : \exists \xi \in \mathbf{S}^* : \wedge\ \exists \phi \in \mathbf{S}^\infty : [\![\Pi]\!](\rho \cdot \xi \cdot \phi)$

$\qquad\qquad\qquad\qquad \wedge\ \forall \chi \in \mathbf{S}^\infty : [\![\Diamond P]\!](\xi \cdot \chi)$

$\langle 2\rangle 1$. $\forall \rho \in \mathbf{S}^*, \eta \in \mathbf{S}^\infty : [\![\Box\Diamond P]\!](\rho \cdot \eta) \Rightarrow \exists \eta_1, \eta_2 : \eta = \eta_1 \cdot \eta_2 \wedge [\![P]\!](\eta_2)$

PROOF: By definition of $\Box$ and $\Diamond$.

$\langle 2\rangle 2$. $\forall \eta_2 \in \mathbf{S}^\infty : [\![P]\!](\eta_2) \Rightarrow \exists \eta_3 \sqsubset \eta_2 : \forall \chi \in \mathbf{S}^\infty : [\![P]\!](\eta_3 \cdot \chi)$

PROOF: By the hypothesis that $\neg P$ is a safety property and (2) (substituting $\neg P$ for Π).

$\langle 2\rangle 3$. $\forall \rho \in \mathbf{S}^*, \eta \in \mathbf{S}^\infty : [\![\Box\Diamond P]\!](\rho \cdot \eta) \Rightarrow \exists \xi \sqsubset \eta : \forall \chi \in \mathbf{S}^\infty : [\![\Diamond P]\!](\xi \cdot \chi)$

PROOF: By $\langle 2\rangle 1$, $\langle 2\rangle 2$, and the definition of $\Diamond$, taking $\eta_1 \cdot \eta_3$ for ξ.

$\langle 2\rangle 4$. $\forall \rho \in \mathbf{S}^*, \eta \in \mathbf{S}^\infty, \xi \sqsubset \eta : [\![\mathcal{C}(\Pi)]\!](\rho \cdot \eta) \Rightarrow \exists \phi \in \mathbf{S}^\infty : [\![\Pi]\!](\rho \cdot \xi \cdot \phi)$

PROOF: By the definition (1) of $\mathcal{C}$.

$\langle 2\rangle 5$. Q.E.D.

PROOF: $\langle 1\rangle 1$, $\langle 2\rangle 3$, and $\langle 2\rangle 4$.

$\langle 1\rangle 3$. Q.E.D.

PROOF: By $\langle 1\rangle 2$, letting τ be $\xi \cdot \phi$ and instantiating χ with ϕ.

A.2 Proof of Proposition 3

We prove the proposition for the special case that F consists of a single WF or SF formula, which is the only case used here. The general case is handled much as in the proof of Proposition 2 of [1]. In the following proof, W/SF denotes either WF or SF.

ASSUME: 1. $\models M \Rightarrow N$
2. $\models \langle A\rangle_v \Rightarrow M$
3. $\sigma \in \mathbf{S}^\infty$

PROVE: $[\![\mathcal{C}(\mathit{Init} \wedge \Box[N]_v \wedge \Diamond\Box[M]_v \wedge \mathrm{W/SF}_v(A))]\!](\sigma) \equiv [\![\mathit{Init} \wedge \Box[N]_v]\!](\sigma)$

$\langle 1\rangle 1$. ASSUME: $\forall \rho \sqsubset \sigma : \exists \tau : [\![\mathit{Init} \wedge \Box[N]_v \wedge \Diamond\Box[M]_v \wedge \mathrm{W/SF}_v(A)]\!](\rho \cdot \tau)$
PROVE: $[\![\mathit{Init} \wedge \Box[N]_v]\!](\sigma)$

PROOF: Assumption $\langle 1\rangle$ (from this step) implies that *Init* holds in the first state of σ and $[N]_v$ holds in every pair of successive states of σ, which implies $[\![\mathit{Init} \wedge \Box[N]_v]\!](\sigma)$ by definition of $\Box$ and of $[\![B]\!]$ for an action B.

$\langle 1\rangle 2$. ASSUME: 1. $[\![\mathit{Init} \wedge \Box[N]_v]\!](\sigma)$
2. $\rho \sqsubset \sigma$
PROVE: $\exists \tau : [\![\mathit{Init} \wedge \Box[N]_v \wedge \Diamond\Box[M]_v \wedge \mathrm{W/SF}_v(A)]\!](\rho \cdot \tau)$

$\langle 2\rangle 1$. Choose states $s_0, s_1, \ldots$ such that $\rho = s_0, \ldots, s_n$ and, for all $i \geq n$,

$$\begin{aligned} &\wedge [\![\text{ENABLED } \langle A\rangle_v]\!](s_i) \Rightarrow [\![\langle A\rangle_v]\!](s_i, s_{i+1}) \\ &\wedge \neg[\![\text{ENABLED } \langle A\rangle_v]\!](s_i) \Rightarrow (s_{i+1} = s_i) \end{aligned}$$

PROOF: The existence of the s_i follows from the definition of ENABLED .

$\langle 2\rangle 2$. $[\![\Box[M]_v]\!](s_n, s_{n+1}, \ldots)$

$\langle 3\rangle 1$. $\forall i \geq n : [\![[M]_v]\!](s_i, s_{i+1})$

PROOF: If $[\![\text{ENABLED } \langle A\rangle_v]\!](s_i)$, this follows from $\langle 2\rangle 1$ and assumption 2. If $\neg[\![\text{ENABLED } \langle A\rangle_v]\!](s_i)$, this also follows from $\langle 2\rangle 1$ because $[\![[M]_v]\!](s, s)$ holds for any state s.

$\langle 3\rangle 2$. Q.E.D.

PROOF: $\langle 3\rangle 1$ and the definitions of $\Box$ and of $[\![B]\!]$ for an action B.

$\langle 2\rangle 3$. $[\![\text{W/SF}_v(A)]\!](s_0, s_1, \ldots)$

PROOF: $[\![\Box\Diamond\text{ENABLED } \langle A\rangle_v]\!](s_0, s_1, \ldots)$ implies $[\![\text{ENABLED } \langle A\rangle_v]\!](s_i)$ for infinitely many i, which by $\langle 2\rangle 1$ implies $[\![\langle A\rangle_v]\!](s_i, s_{i+1})$ for infinitely many i, which implies $[\![\Box\Diamond\langle A\rangle_v]\!](s_0, s_1, \ldots)$. The result then follows from the definition of WF and SF, since $\neg\Box\Diamond\text{ENABLED } \langle A\rangle_v$ is equivalent to $\Diamond\Box\neg\text{ENABLED } \langle A\rangle_v$, which implies $\Box\Diamond\neg\text{ENABLED } \langle A\rangle_v$.

$\langle 2\rangle 4$. $[\![\Box[N]_v]\!](s_0, s_1, \ldots)$

$\langle 3\rangle 1$. $\forall i : [\![[N]_v]\!](s_i, s_{i+1})$

$\langle 4\rangle 1$. ASSUME: $i < n$
PROVE: $[\![[N]_v]\!](s_i, s_{i+1})$

PROOF: $\langle 2\rangle 1$ and assumptions $\langle 1\rangle$:1 and $\langle 1\rangle$:2 (from step $\langle 1\rangle 2$).

$\langle 4\rangle 2$. ASSUME: $i \geq n$
PROVE: $[\![[N]_v]\!](s_i, s_{i+1})$

PROOF: By $\langle 2\rangle 2$, the definition of $\Box$, and assumption 1.

$\langle 4\rangle 3$. Q.E.D.

PROOF: $\langle 4\rangle 1$ and $\langle 4\rangle 2$.

$\langle 3\rangle 2$. Q.E.D.

PROOF: $\langle 3\rangle 1$ and the definitions of $\Box$ and of $[\![B]\!]$ for an action B.

$\langle 2\rangle 5$. Q.E.D.

PROOF: $\langle 2\rangle 2$, $\langle 2\rangle 3$, $\langle 2\rangle 4$, the definition of $[\![\mathit{Init}]\!]$, and the definition of $\Diamond$, taking $s_n, s_{n+1}, \ldots$ for τ.

$\langle 1\rangle 3$. Q.E.D.

PROOF: $\langle 1\rangle 1$, $\langle 1\rangle 2$, and the definition (1) of $\mathcal{C}$.

A.3 Proof of Proposition 4

$\langle 1\rangle 1$. $\vdash \mathit{Init} \wedge \Box[N]_v \Rightarrow \Box \mathit{sin}([N]_v, \mathit{Init})$

$\langle 2\rangle 1$. $\models \mathit{Init} \Rightarrow \mathit{sin}([N]_v, \mathit{Init})$

PROOF: Definition (6) of *sin*.

⟨2⟩2. $\models [N]_v \wedge \mathit{sin}([N]_v, \mathit{Init}) \Rightarrow \mathit{sin}([N]_v, \mathit{Init})'$

PROOF: Definition (6) of *sin*.

⟨2⟩3. $\vdash \mathit{sin}([N]_v, \mathit{Init}) \wedge \Box[N]_v \Rightarrow \Box \mathit{sin}([N]_v, \mathit{Init})$

PROOF: ⟨2⟩2, assumptions 1 and 2, and proof rule INV1.

⟨2⟩4. Q.E.D.

PROOF: ⟨2⟩1, ⟨2⟩3, and assumptions 1 and 2.

⟨1⟩2. $\vdash \mathit{sin}([N]_v, \mathit{Init}) \Rightarrow I$

⟨2⟩1. $\forall s \in \mathbf{S} : [\![\mathit{sin}([N]_v, \mathit{Init})]\!](s) \Rightarrow$

$$\exists s_0, \ldots, s_n \in \mathbf{S} : [\![\mathit{Init} \wedge \Box[N]_v]\!](s_0, \ldots, s_n, s, s, s, \ldots)$$

PROOF: Definition (6) of *sin*, and the definitions of $\Box$ and $[N]_v$.

⟨2⟩2. $\forall s, s_0, \ldots, s_n \in \mathbf{S} : [\![\mathit{Init} \wedge \Box[N]_v]\!](s_0, \ldots, s_n, s, s, s, \ldots) \Rightarrow [\![I]\!](s)$

PROOF: Assumption 3 and definition of $\Box I$.

⟨2⟩3. $\models \mathit{sin}([N]_v, \mathit{Init}) \Rightarrow I$

PROOF: ⟨2⟩1 and ⟨2⟩2.

⟨2⟩4. Q.E.D.

PROOF: ⟨2⟩3 and assumptions 1 and 2.

⟨1⟩3. Q.E.D.

PROOF: ⟨1⟩1, ⟨1⟩2, and proof rule STL4 of [8].

A.4 Proof of Proposition 5

Let $\mathbf{N}$ be the set of natural numbers and let $x_1, \ldots, x_n$ be the free variables of P and N. Since $[N]_v \equiv [[N]_v]_{\langle v,w \rangle}$, by replacing N with $[N]_v$ and v with $\langle v, x_1, \ldots, x_n \rangle$, we can assume:

(5) v is a tuple whose components include all free variables of P and N.

In the following proof, P_n is the predicate that is true iff P can be made true by taking n N-steps, but with no fewer than n such steps.

LET: $P_n \triangleq$ **if** $n = 0$ **then** P

$$\textbf{else} \ \wedge \forall i < n : \neg P_i \quad \wedge \text{ENABLED}\,(N \wedge (v' \neq v) \wedge P'_{n-1})$$

$M \triangleq N \wedge (\forall n : P_{n+1} \Rightarrow P'_n)$

⟨1⟩1. $\vdash \mathit{Init} \wedge \Box[N]_v \Rightarrow \Box(\exists n : P_n)$

LET: $\pi(s, n) \triangleq \exists s_0, \ldots, s_n : \wedge (s = s_0) \wedge [\![P]\!](s_n)$

$$\wedge \forall i < n : [\![N \wedge (v' \neq v)]\!](s_i, s_{i+1})$$

⟨2⟩1. $\forall (s_0, s_1, \ldots) \in \mathbf{S}^\infty :$

$$[\![\mathit{Init} \wedge \Box[N]_v]\!](s_0, s_1, \ldots) \Rightarrow \forall i \in \mathbf{N} : \exists n \in \mathbf{N} : \pi(s_i, n)$$

PROOF: Assumptions 3 and 4, (3) (the definition of $\mathbf{P}_{\Pi}(P)$), and the definitions of $\mathcal{C}$ and $\Diamond$.

$\langle 2 \rangle 2$. $\forall s \in \mathbf{S}, n \in \mathbf{N} : [\![P_n]\!](s) \equiv \pi(s, n) \wedge (\forall i < n : \neg \pi(s, i))$

PROOF: By induction on n from the definitions of P_n, π, and ENABLED.

$\langle 2 \rangle 3$. $\forall s \in \mathbf{S} : [\![\exists n : P_n]\!](s) \equiv (\exists n \in \mathbf{N} : \pi(s, n))$

PROOF: $\langle 2 \rangle 2$.

$\langle 2 \rangle 4$. $\models \mathit{Init} \wedge \Box[N]_v \Rightarrow \Box(\exists n : P_n)$

PROOF: $\langle 2 \rangle 1$, $\langle 2 \rangle 3$, and the definitions of $\Box$ and $[\![[N]_v]\!]$.

$\langle 2 \rangle 5$. Q.E.D.

PROOF: $\langle 2 \rangle 4$, assumptions 2 and 1, and Proposition 4, since ENABLED A is obtained by existential quantification over the primed variables of A, so it is expressible if A is, for any action A.

$\langle 1 \rangle 2$. ASSUME: $k \in \mathbf{N}$

PROVE: $\vdash \Box[M]_v \wedge \mathrm{WF}_v(M) \Rightarrow (P_{k+1} \leadsto P_k)$

$\langle 2 \rangle 1$. $\vdash P_{k+1} \wedge [M]_v \Rightarrow P'_{k+1} \vee P'_k$

PROOF: Definition of M and assumption 5 (which, by induction on k, implies $P_{k+1} \wedge (v' = v) \Rightarrow P'_{k+1}$).

$\langle 2 \rangle 2$. $\vdash P_{k+1} \wedge \langle M \rangle_v \Rightarrow P'_k$

PROOF: Definition of M.

$\langle 2 \rangle 3$. $\vdash P_{k+1} \Rightarrow$ ENABLED $\langle M \rangle_v$

$\langle 3 \rangle 1$. $\models P_{k+1} \Rightarrow \forall n \neq (k+1) : \neg P_n$

PROOF: Definition of P_n.

$\langle 3 \rangle 2$. $\models P_{k+1} \Rightarrow (M \equiv N \wedge P'_k)$

PROOF: $\langle 3 \rangle 1$ and definition of M.

$\langle 3 \rangle 3$. $\models P_{k+1} \Rightarrow$ ENABLED $\langle M \rangle_v$

PROOF: $\langle 3 \rangle 2$ and the definition of P_{k+1}.

$\langle 3 \rangle 4$. Q.E.D.

PROOF: $\langle 3 \rangle 3$ and assumption 1.

$\langle 2 \rangle 4$. Q.E.D.

PROOF: $\langle 2 \rangle 1$–$\langle 2 \rangle 3$ and rule WF1 of [8].

$\langle 1 \rangle 3$. $\vdash \Diamond\Box[M]_v \wedge \mathrm{WF}_v(M) \Rightarrow \Box\Diamond P$

$\langle 2 \rangle 1$. $\vdash \Box(\exists n : P_n) \wedge \Diamond\Box[M]_v \wedge \mathrm{WF}_v(M) \Rightarrow ((\exists n : P_n) \leadsto P)$

PROOF: $\langle 1 \rangle 2$ and the Lattice Rule of [8].

$\langle 2 \rangle 2$. $\vdash \Box F \wedge (F \leadsto G) \Rightarrow \Box\Diamond G$, for any temporal formulas F and G.

PROOF:
$$\begin{array}{rcll} \Box F \wedge (F \leadsto G) & \equiv & \Box F \wedge \Box(F \Rightarrow \Diamond G) & \text{Definition of } \leadsto \\ & \equiv & \Box(F \wedge (F \Rightarrow \Diamond G)) & \text{Rule STL5 of [8].} \\ & \Rightarrow & \Box\Diamond G & \text{Rule STL4 of [8].} \end{array}$$

$\langle 2 \rangle 3$. Q.E.D.

PROOF: $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$.

$\langle 1 \rangle 4$. Q.E.D.

$\langle 2 \rangle 1.$ $\vdash \mathcal{C}(\mathit{Init} \wedge \Box[N]_v \wedge \Diamond\Box[M]_v \wedge \mathrm{WF}_v(M)) \equiv \mathit{Init} \wedge \Box[N]_v$

PROOF: Proposition 3, since $\vdash M \Rightarrow N$ by definition of M.

$\langle 2 \rangle 2.$ $\vdash \mathit{Init} \wedge \Box[N]_v \wedge \Diamond\Box[M]_v \wedge \mathrm{WF}_v(M) \Rightarrow \mathcal{C}(\Pi) \wedge \Box\Diamond P$

PROOF: $\langle 1 \rangle 1$, $\langle 1 \rangle 3$, and assumption 3.

$\langle 2 \rangle 3.$ $\vdash \mathcal{C}(\Pi) \Rightarrow \mathcal{C}(\mathcal{C}(\Pi) \wedge \Box\Diamond P)$

PROOF: $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, assumption 3, and part 2 of Proposition 1.

$\langle 2 \rangle 4.$ Q.E.D.

PROOF: $\langle 2 \rangle 3$ and Proposition 2.