A COMPLETENESS THEOREM FOR TLA

Preface: A Quick Introduction to TLA

Here's a quick and dirty informal definition and semantics for TLA. A more precise one is given later. If you are already familiar with the new, improved TLA (defined a little differently than in SRC Report No. 57) you can skip to the Introduction.

Values:
I assume a set of values, big enough to contain all the constants of interest. It includes the values 1, TRUE, NAT (the set of all naturals), $\{n \in NAT : n \text{ a prime}\}$, etc.

State, Variable:
A variable is something that assigns a value to every state. I let s.x, denote the value state s assigns to variable s. Or maybe a state s is something that assigns a value s.x to a variable x. Take your pick.

State Function:
An expression made from variables and constants, such as $x^2 + 3*y$. A state function f assigns a value s.f to every state s. For example,

$$s.(x^2 + 3*y) = (s.x)^2 + 3*(s.y).$$

Predicate:
A boolean-valued state function--for example,

$$x^2 < 3*y$$

Action:
A Boolean expression involving variables, primed variables and constants--for example, $x + 1 < 2*y'$. An action maps pairs of states to Booleans. Letting s.A.t denote the value that action A assigns to the pair (s,t), I define

$$s.(x + 1 < 2*y').t \;=\; (s.x) + 1 < 2*(t.y)$$

In other words, the unprimed variables talk about the left-hand state, and the primed variables talk about the right-hand state. Think of s.A.t = TRUE as meaning that an A-step can take state s to state t. An action is valid, written $\models$ A, iff s.A.t is true for all states s and t.

Enabled(A):
For any action A, the predicate Enabled(A) is defined by

$$s.\mathrm{Enabled}(A) \triangleq \exists\, t : s.A.t$$

$f'=f$:
For any state function f, the action $f'=f$, which is sometimes written Unchanged(f), is defined by

$$s.(f'=f).t \;\triangleq\; (t.f) = (s.f)$$

$[A]_f$:
The action $[A]_f$ is defined by

$$[A]_f \triangleq A \lor (f'=f)$$

An $[A]_f$ step is either an A step or leaves f unchanged.

$<A>_f$:
The action $<A>_f$ is defined by

$$<A>_f \triangleq \neg[\neg A]_f$$

It equals A $\wedge$ ($f' \neq f$). An $<A>_f$ step is an A step that changes f.

The Raw Logic:
A Raw Logic formula is any formula made from actions using logical operators and the unary $\Box$ operator--for example

$$A \vee \Box(B \wedge \Box\neg\Box\neg A)$$

where A and B are actions. A Raw Logic formula is a Boolean-valued function on infinite sequences of states. An infinite sequence of states is called a BEHAVIOR. An action A is interpreted as the temporal formula asserting that first step of the behavior is an A step. The formula $\Box A$ asserts that every step is an A step. In general, let $s_0, s_1, \ldots \models F$ denote the value that formula F assigns to the sequence $s_0, s_1, \ldots$. The semantics of Raw Logic formulas is defined as follows, where A is any action and F and G are any formulas:

$$s_0, s_1, s_2, \ldots \models A \triangleq s_0.A.s_1$$
$$s_0, s_1, s_2, \ldots \models \Box F$$
$$\triangleq \forall\, n \geq 0 : s_n, s_{n+1}, s_{n+2}, \ldots \models F$$
$$s_0, s_1, s_2, \ldots \models \neg F \triangleq \neg(s_0, s_1, s_2, \ldots \models F)$$
$$s_0, s_1, s_2, \ldots \models F \vee G \triangleq \text{etc.}$$

A formula F is valid, written $\models F$, iff it is true for all behaviors.

TLA:
The Raw Logic is wonderfully simple, but it is too expressive. It allows you to assert that something is true of the next state, which ruins any effort to heierarchically refine programs. We define TLA to be the subset of Raw Logic formulas obtained by application of $\Box$ and logical operators starting not from arbitrary actions, but from predicates and actions of the form $[A]_f$. For example:

$$P \Rightarrow \neg\Box\neg\Box[A]_f \vee \Box(Q \Rightarrow \Box[B]_g)$$

Observe that $\Box[A]_f$ asserts that every step is either an A step or else leaves f unchanged.

As is usual in temporal logic, we define $\Diamond$ and $\leadsto$ by

$$\Diamond F \triangleq \neg\Box\neg F$$
$$F \leadsto G \triangleq \Box(F \Rightarrow \Box G)$$

The Raw formula $\Diamond A$ is a TLA formulas iff A is a predicate or an action of the form $\Diamond<A>_f$.

Technical point. Since $\Box F \wedge \Box G = \Box(F \wedge G)$ holds for any F and G, it's convenient to let TLA include formulas of the form $\Box(P \wedge \Box[A]_f)$ where P is a predicate.

Introduction

This is a relative completeness proof for TLA, a la Cook. It is not a completeness result for all of TLA, just for the class of formulas that one is interested in proving. The formulas we're interested in are of the form

$$\text{Program} \Rightarrow \text{Property}$$

A Program has the form

$P \wedge \Box[A]_f \wedge$ Fairness

for P a predicate. So far, all the Fairness conditions have have been conjunctions of the form $SF_f(B)$ or $WF_f(B)$, where B implies A and

$$WF_f(B) \triangleq \Box\Diamond\langle B\rangle_f \vee \Box\Diamond\neg Enabled(\langle B\rangle_f)$$
$$SF_f(B) \triangleq \Box\Diamond\langle B\rangle_f \vee \Diamond\Box\neg Enabled(\langle B\rangle_f)$$

The theorem allows the more general class of programs in which Fairness is the conjunction of formulas of the form

$$\Box\Diamond\neg TACT \vee \Diamond\Box TACT \text{ or } \Box\Diamond\neg TACT \vee \Box\Diamond TACT,$$

where TACT denotes any formula of the form $Q \wedge [B]_g$, so $\neg$TACT is a formula of the form $Q \vee \langle B\rangle_g$. The Fairness formula must satisfy the additional requirement that program is machine closed, meaning that for any safety property S:

If $\models (P \wedge \Box[A]_f \wedge \text{Fairness}) \Rightarrow S$
then $\models (P \wedge \Box[A]_f) \Rightarrow S$

(The theorem requires this only when S is of the form $\Box$TACT.) Machine closure, which was called "feasibility" by Apt, Francez, and Katz, is a reasonable requirement for any fairness condition. It can be argued that a condition not satisfying it is not a fairness condition, since it can't be implemented by a memory-less scheduler.

The Property can have any of the following forms:

Predicate
$\Box$Predicate
Predicate $\leadsto$ Predicate
GeneralProgram

where a GeneralProgram is like a Program, except without the machine-closure requirement on its fairness condition. The absence of this requirement is important, for the following reason. To prove that program Π_1 implements program Π_2, one proves $\Pi_1 \Rightarrow \Phi_2$, where Φ_2 is obtained from Π_2 by substituting state functions for variables. This substitution preserves the form of the formula Π_2, but can destroy machine-closure.

Proving relative completeness for safety properties in TLA is pretty much the same as proving it for the Floyd/Hoare method. The completeness results for Hoare's method assumes the expressibility of the predicate sp(S, P) for program statements S and predicates P, where sp is the strongest postcondition operator. Assuming such predicates for arbitrary statements S, which include loops or recursion, is equivalent to assuming the expressibility of sp(A, P) and sin(A, P) for atomic actions A, where sin is the strongest invariant operator.

Proving relative completeness for liveness is somewhat trickier. It requires induction over well-founded sets. Taking a simple, intuitive approach leads to a result whose practical interest is rather doubtful. For example, Mann and Pnueli ("Completing the Temporal Picture") use the axiom of choice to pull a well-founded ordering on the state space out of a hat. Such a construction requires the assumption that every semantic predicate is syntactically expressible.

Getting the precise expressibility assumptions, and avoiding mistakes, required a careful formal exposition.

The Assumptions

In relative completeness results for Hoare logic, one assumes a complete system for reasoning about predicates. In TLA, all the serious reasoning is in the domain of actions. So, we assume a complete system for reasoning about actions. More precisely, letting $\vdash$ denote provability, we assume a set ACT of expressible actions such that $(\models A) \Rightarrow (\vdash A)$ for any action A in ACT. There are various simple assumptions about ACT--such as its being closed under boolean operations. Let PRED denote the set consisting of all predicates in ACT (remember that a predicate is an action that doesn't mention unprimed variables). The least reasonable assumption is that for any P in PRED and A in ACT, sin(A, P) and sp(A, P) are in PRED. Of course, this assumption is what really puts the "relative" in "relative completeness".

The relatively complete logical system consists of the following:

1. The usual assortment of simple propositional temporal logic rules and axioms that you'd expect, since TLA includes simple temporal logic (the logic that's the same as the Raw Logic except starting with predicates, not arbitrary actions).

2. An induction principle, which is what you'd expect for any relatively complete system for proving temporal logic liveness properties.

3. The two TLA axioms:

   $$\vdash (\Box P \equiv P \wedge \Box [P \Rightarrow P']_P)$$

   $$\vdash (A \Rightarrow B) \;\Rightarrow\; \vdash (\Box A \Rightarrow \Box B)$$

   where P is a predicate, and A and B are actions of the form $P \wedge [A]_f$.

The axioms of 3 are the only ones that mention actions. The axioms of 1 only mention arbitrary formulas, and the induction principle of 2 talks only about predicates. These axioms are actually valid for the Raw logic, and in that logic the second axiom of 3 is a special case of the axiom

$$\vdash (F \Rightarrow G) \;\Rightarrow\; \vdash (\Box F \Rightarrow \Box G)$$

from 1, for arbitrary formulas F and G. However, $[A]_f \Rightarrow [B]_g$ isn't a TLA formula (it's a formula in the logic of actions, but not in TLA), so the second axiom of 3 is needed if you want to do all your reasoning completely within TLA.

The induction axiom 2 is tricky enough to be worth mentioning. To get it right, we first have to generalize everything to include logical variables. If you want to describe an n-process algorithm with a TLA formula, for an arbitrary but fixed n, then n is a logical variable of the formula. A logical variable represents an unspecified value that is the same for all states of a behavior.

In the semantics of actions and TLA formulas, Booleans have to be replaced by Boolean-valued formulas involving logical variables. (Formally, Booleans become boolean-valued functions on interpretations, where an interpretation is an assignment of values to all logical variables.) Logical variables pop up all the time when you use TLA in practice. For example, if you have a distributed algorithm with a set Node of nodes, then Node is a logical variable. In fact, if you go really overboard in formalism--as you must to verify things mechanically--then everything that's not a program variable (the kind of variable I first talked about) or a logical operator is a logical variable. In the expression x + 3, the + and the 3 are logical variables. We just happen to have a lot of axioms about the logical variables + and 3, such as 1+1+1 = 3, while we have just a few axioms about the logical variable n (for example $n \in NAT$, $n > 0$).

But, I digress. I was talking about the induction principle. An induction principle involves induction over a well-founded ordering on a set. Intuitively, a well-founded ordering on a set S is a relation > such that there does not exist an infinite sequence $c_1 > c_2 > c_3 > \ldots$. More precisely,

$$\text{Well-Founded}(>, S) \triangleq \neg \forall i : (i \in NAT) \Rightarrow \exists c_i : (c_i \in S) \land (c_i > c_{i+1})$$

But, what does this formula mean? For me, the most sensible way to interpret it as a logical formula is to rewrite it as

$$\text{Well-Founded}(v > w, S) \triangleq \forall c : \neg \forall i : (i \in NAT) \Rightarrow (c(i) \in S) \land (c(i) > c(i+1))$$

where $v > w$ is a formula with free logical variables v and w, and $(c(i) > ci+1))$ is the formula obtained by substituting c(i) for v and c(i+1) for w in the formula $v > w$. This is a higher-order formula, involving quantification over a function symbol c.

The completeness result requires, as an assumption, that if the formulas "$v > w$" and "$v \in S$" are expressible, then Well-Founded($v > w$, S) is provable if it's true. I think that if you look closely at Manna and Pnueli's paper, you'll find that they are implicitly assuming this for any formula "$v > w$"--not just for an expressible one.

Anyway, the actual temporal induction principle looks as follows, where P(w) denotes a formula containing w as free logical variables, P(v) denotes the result of substituting v for w, and F is an arbitrary temporal formula.

$$\begin{array}{l} \text{If } \vdash \exists w \in S \\ \quad w \text{ not free in } F \\ \quad \vdash \text{Well-Founded}(>, S) \\ \quad \vdash (F \land w \in S \\ \qquad \Rightarrow (P(w) \leadsto \exists v : (v \in S) \land (w > v) \land P(v))) \\ \text{then } \vdash \neg F \end{array}$$

I've actually lied a bit. I assume this rule when w is a k-tuple of distinct logical variables, and I assume the provability only of Well-Founded($v > w$, VAL^k), where $v > w$ is an expressible formula and VAL^k denotes the set of k-tuples of values. I could have done it the other way by making a few more expressibility assumptions--such as assuming that "$v \in VAL^k$" is expressible--but I think that would have been a little more complicated.

ACTIONS

Primitives

The following are primitive notions, along with their intuitive explanations.

- VAL : A set of values, containing the values TRUE and FALSE (among many others). The semantics of TLA is based on this set.
- ST : A set of states.
- PVBL : A set of program variables. These variables appear in TLA formulas and represent the primitive state components. That is, a state assigns a value in VAL to every variable in PVBL.
- LVBL : An infinite set of logical variables. A logical variable denotes a fixed, unspecified elements of VAL; it represents a program "constant".
- ACT : The set of expressible actions.
- PRED : The set of expressible predicates.
- SFCN : The set of expressible state functions.

Notations

$VAL^k \triangleq$ The set of all k-tuples of elements in VAL.

$PVBL^k \triangleq$ The set of k-tuples of DISTINCT program variables in PVBL.

$LVBL^k \triangleq$ The set of k-tuples of DISTINCT logical variables in LVBL.

Two k-tuples v, w $\in LVBL^k$ are said to be DISJOINT iff they have no components in common.

For v $\in LVBL^k$, f any formula, c in $LVBL^k$ or VAL^k:
f[c/v] $\triangleq$ The result of substituting each component of c for the corresponding component variable of v.

If g(v) denotes a formula, I will let g(c) denote g(v)[c/v].

INTRPT $\triangleq$ The set of mappings [LVBL $\rightarrow$ VAL].

INTRPT is the set of interpretations--substitutions of values for logical variables. For any set S, an element of the set of mappings [INTRPT $\rightarrow$ S] is an object that yields an element of S after substituting values for all logical variables.

Expressibility and Completeness Assumptions

Below are the expressibility assumptions and the semantic interpretations of those assumptions. These semantic interpretations provide a semantics for the (nontemporal) logic of actions. In the following, I assume

$s, t \in ST$
$x \in PVBL^k$
$c \in VAL^k$
$v \in LVBL^k$
$P, Q \in PRED$
$A, B \in ACT$
$f, g \in SFCN$

Formally, the assumptions are universally quantified over these objects. The semantic domains are defined as follows, where [[O]] denotes the "meaning" of an object O.

$$[[x]] : ST \rightarrow VAL^k$$

$$[[f]] : ST \rightarrow [INTRPT \rightarrow VAL]$$

$$[[P]] : ST \rightarrow [INTRPT \rightarrow \{TRUE, FALSE\}]$$

$$[[A]] : ST \times ST \rightarrow [INTRPT \rightarrow \{TRUE, FALSE\}]$$

The following notation is used in place of the semantic brackets [[ ]].

$$s.x \triangleq [[x]](s)$$
$$s.f \triangleq [[f]](s)$$
$$s.P \triangleq [[P]](s)$$
$$s.A.t \triangleq [[A]](s,t)$$

Thus, for example, s.P is an object that yields a Boolean after substituting values for all logical variables. Validity of an action formula is defined by

$$\models A \triangleq \forall\, int \in INTRPT : \forall\, s,t \in ST : s.A.t(int)$$

Thus, validity of A means that s.A.t is true for all substitutions of values for logical variables and all states s and t.

The following are the assumptions and their meanings.

EX(-1). $TRUE \in PRED$

$$s.TRUE \triangleq TRUE$$

EX0. $P, P' \in ACT$
$P \in SFCN$

$$s.P.t \triangleq s.P$$
$$s.P'.t \triangleq t.P$$

A predicate is identified with an action that does not depend on the second [new] state. Since VAL contains the elements TRUE and FALSE, a predicate P is a fortiori "semantically" a state function. The assumption $P \in SFCN$ states that this semantic state function is expressible.

EX1. $sin(A,P) \in PRED$
$sp(A,P) \in PRED$,

$$s.sp(A,P) \triangleq \exists\, t : t.P \wedge t.A.s$$
$$s.sin(A,P) \triangleq \exists\, i \geq 0 : s.sp^i(A,P)$$

where $sp^0(A,P) \triangleq P$
$sp^{i+1}(A,P) \triangleq sp(A, sp^i(A,P))$

The operators sp and sin are the usual strongest postcondition and strongest invariant operators.

EX2. $A \wedge B$, $\neg A$, $A \vee B \in ACT$
$P \wedge Q$, $\neg P$, $P \vee Q \in PRED$

$$s.(A \wedge B).t \triangleq s.A.t \wedge s.B.t$$
$$s.(A \vee B).t \triangleq s.A.t \vee s.B.t$$
$$s.(\neg A).t \triangleq \neg(s.A.t)$$

The corresponding semantic definitions for predicates follow from EX2 and EX0.

EX6. $x = w \in PRED$, and $\neg \models ((\exists\ w : (x = w)) \equiv FALSE)$

$$s.(x = w) \triangleq s.x = w$$

The assumption invalidity assumption asserts that given any finite set of variables x_i and values v_i, there is a state in which each x_i has the value v_i.

EX7. Only a finite number of program variables appear in A.

$$\exists\ k, x : \forall\ r, s, t, u \in ST : (r.x = t.x \wedge s.x = u.x) \Rightarrow r.A.s = t.A.u$$

In the semantic definition, x is any k-tuple of program variables whose components include all the variables that appear in A. (Since there is no quantification over program variables in actions, any variable that appears in A is free in A.)

EX7b. Only a finite number of logical variables appear free in A.

$$\exists\ w \in LVBL^k : v \text{ disjoint from } w \Rightarrow s.A.t \equiv s.A[c/v].t$$

In the semantic definition, w is a k-tuple of logical variables containing all logical variables occurring free in A.

EX8. (a) $NAT \in SFCN$
(b) If $S \in SFCN$ and $u \in LVBL$ then $(u \in S) \in PRED$
(c) If $Q \in PRED$, $x \in PVBL$, $w \in LVBL$ then $Q[w/x] \in PRED$
(d) $v > w \in CONSTPRED$, then Well-Founded$(v>w, VAL^k) \in CONSTPRED$.

CONST PRED = set of all PRED′s with no free logical variables. So, $Q \in CONSTPRED$ iff $(\exists\ s : s.Q) \equiv\ \models Q$

If $w \in LVLB^k$ has components disjoint from v, and $w>v \in PRED$, then Well-Founded$(w>v, VAL^k) \in PRED$.

$$s.\text{Well-Founded}(w>v, S) \triangleq \neg \forall\ i \geq 0 : \exists\ c_i \in S : s.(c_{i+1} > c_i)$$

where $S \subseteq VAL^k$
$$(c_{i+1} > c_i) \triangleq ((w > v)[c_{i+1}/w])[c_i/v]$$

This is the formal definition of well-founded ordering $>$ on a set S of k-tuples of values. Usually, w>v will be a "constant" relation--one that is independent of the state.

EX9. If $w \in LVBL^k$, then $(\exists\ w : P) \in PRED$.

$$s.(\exists\ w : P) \triangleq \exists\ c \in VAL^k : s.P[c/w]$$

EX10. $f' = f \in ACT$

$$s.(f' = f).t \triangleq t.f = s.f$$

EX11. $(f,g) \in SFCN$.

$$s.(f,g) \triangleq (s.f,\ s.g)$$

Note that if SFCN contains any reasonably rich set of state functions, then this assumption requires that VAL contains all ordered pairs of elements in VAL.

Relative Completeness Assumption

A logical system consists of a set of wff′s and a collection of rules for proving that certain wff′s are theorems. We usually let $\vdash$ F denote that the wff F is a theorem of the system. Since we are considering two logical systems, the logic of actions and TLA, there are two logical systems and two "$\vdash$"s. We′ll use $\vdash_{ACT}$ for the logic of actions.

The relative completeness assumption is:

RC1. For all $A \in Act$: $(\models A) \equiv\ \vdash_{ACT} A$

This of course assumes the soundness as well as the completeness of the proof system for actions.

TEMPORAL LOGIC

Notation

$[A]_f \triangleq A \lor (f' = f)$

$<A>_f \triangleq \neg[\neg A]_F$
$= A \land (f' \neq f)$

$ST^\omega \triangleq$ The set of infinite sequences of elements in ST.

For $\sigma \in ST^\omega$, $i \geq 0$:

$\sigma_i \triangleq$ The ith state in σ, where the first state is σ_0.

$\sigma^{+i} \triangleq$ The sequence σ_i, σ_{i+1}, ... in ST^ω

Temporal Logic

The wffs of a simple temporal logic consist of a set of formulas TEMPORAL defined in terms of a set ELEM of elementary formulas by the following BNF grammar:

$$\text{TEMPORAL} \triangleq \text{ELEM} \mid \neg\text{TEMPORAL} \mid \text{TEMPORAL} \lor \text{TEMPORAL} \mid \text{TEMPORAL} \land \text{TEMPORAL} \mid \Box\text{TEMPORAL}$$

In the following, I assume

$F, G \in \text{TEMPORAL}$
$\sigma \in \text{ST}^{\omega}$

The semantic domain for temporal formulas is defined by

$$[[F]] : \text{ST}^{\omega} \to (\text{INTRPRT} \to \{\text{TRUE}, \text{FALSE}\})$$

I will eliminate the semantic brackets by writing

$$\sigma \models F \triangleq [[F]](\sigma)$$

The semantics of the temporal logic is defined in terms of the semantics of elementary formulas by:

$$\begin{aligned}
\sigma &\models \neg F &&\triangleq \neg(\sigma \models F)\\
\sigma &\models F \lor G &&\triangleq (\sigma \models F) \lor (\sigma \models G)\\
\sigma &\models F \land G &&\triangleq (\sigma \models F) \land (\sigma \models G)\\
\sigma &\models \Box F &&\triangleq \forall\, i \geq 0 : \sigma^{+i} \models F
\end{aligned}$$

Validity is defined by

$$\models F \triangleq \forall\, \sigma \in \text{ST}^{\omega} : \sigma \models F$$

The temporal logic RAW is defined by letting ELEM $\triangleq$ ACT, where, for $A \in \text{ACT}$,

$$\sigma \models A \triangleq \sigma_0 .A. \sigma_1$$

Since an action A in ACT is a RAW formula, we have defined $\models A$ twice: once in defining the semantics of actions, and just now in defining the semantics of RAW formulas. It is easy to check that the two definitions are equivalent.

The temporal logic TLA is defined by letting

$$\begin{aligned}
\text{TACT} &\triangleq [\text{ACT}]_{\text{SFCN}} \mid \text{PRED} \land [\text{ACT}]_{\text{SFCN}}\\
\text{ELEM} &\triangleq \text{PRED} \mid \Box\text{TACT}
\end{aligned}$$

It follows from EX2, and EX10 that TACT is a subset of ACT, so EX0 implies that TLA is a subset of RAW. The semantics of TLA are then defined by the semantics of RAW.

Note: We would get the same logic by defining ELEM just to be PRED or $[\text{ACT}]_{\text{SFCN}}$. However, that would leave us in the somewhat embarrassing position of being able to write the formula $\Box P \land \Box[A]_f$ but not being able to write the equivalent formula $\Box(P \land \Box[A]_f)$.

The following derived operators are defined, for any $F, G \in \text{RAW}$.

$$\begin{aligned}
\Diamond F &\triangleq \neg\Box\neg F\\
F \leadsto G &\triangleq \Box(F \Rightarrow \Diamond G)
\end{aligned}$$

Expressibility assumption EX2 implies $\Diamond\langle A\rangle_f \in \text{TLA}$ for any $A \in \text{ACT}$ and $f \in \text{SFCN}$.

AXIOMS AND DEDUCTION RULES FOR TLA

We let $\vdash_{TLA}$ denote the provability relation for TLA.
The first deduction rule is:

STL0. For any $P \in$ PRED: $(\vdash_{ACT} P) \Rightarrow (\vdash_{TLA} P)$

Assuming that all our complete logical system for TLA is sound, so $\vdash_{TLA}$ P implies $\models$ P, it follows from the relative completeness assumption RC that $\vdash_{TLA}$ P implies $\vdash_{ACT}$ P. Hence, STL0 implies that $\vdash_{TLA}$ and $\vdash_{ACT}$ are equivalent for predicates in PRED. Since these predicates are the only elements of both ACT and TLA, we will drop the subscripts and use the same provability symbol $\vdash$ for both actions and TLA formulas.

In the following rules and axioms, F and G are assumed to be arbitrary formulas in TLA.

The next part of the logical system of TLA consists of modus ponens ($\vdash$ F and $\vdash$ (F $\Rightarrow$ G) imply $\vdash$ G) and the axioms of propositional calculus. Instead of giving these axioms explicitly, we simply state the following rule:

PROPCALC: If F is derivable by Modus Ponens and substitution of TLA formulas for atoms in tautologies of propositional logic, then $\vdash$ F.

A rule of the form F, G $\vdash$ H means that $\vdash$ H can be derived from $\vdash$ F and $\vdash$ G. We sometimes write this rule in the form

$$\frac{F,\ G}{H}$$

The remaining axioms and rules are:

STL1. $\vdash \Box(F \land G) \equiv \Box F \land \Box G$
STL2. $\vdash \Box\Box F = \Box F$
STL4. $\vdash \Diamond\Box F \land \Diamond\Box G \equiv \Diamond\Box(F \land G)$
STL6. $\vdash \Box\Diamond\Box F = \Diamond\Box F$
STL7. $(F \Rightarrow G) \vdash (\Box F \Rightarrow \Box G)$
STL8. $\vdash (\Box F \Rightarrow F)$
TLA1. For all $P \in$ PRED : $\vdash (\Box P \equiv P \land \Box[P \Rightarrow P']_P)$
TLA2. For all A, B $\in$ TACT :
$\vdash(A \Rightarrow B) \Rightarrow \vdash(\Box A \Rightarrow \Box B)$

LATTICE:

$$\frac{\begin{array}{l} w \in LVBL^k \\ \exists\, w \in S \\ w \text{ not free in } F \\ \text{Well-Founded}(<, S) \\ (F \land w \in S) \Rightarrow (P(w) \leadsto \exists\, v : (v \in S) \land (v < w) \land P(v)) \end{array}}{\neg F}$$

Note: All these rules and axioms are valid for formulas F and G in RAW, not just for formulas in TLA. Extending the rules and axioms to RAW would Make TLA2 a special case of STL7.

The following are derived rules and axioms.

INV : For $P \in$ PRED, $A \in$ TACT :

$(\vdash P \land A \Rightarrow P') \Rightarrow (\vdash P \land \Box A \Rightarrow \Box P)$

1. $(\vdash P \land A \Rightarrow P') \Rightarrow (\vdash \Box A \Rightarrow \Box[P \Rightarrow P']_P)$

   Pf: $\vdash P \land A \Rightarrow P'$

   $\Rightarrow \vdash A \Rightarrow (P \Rightarrow P')$ (PROPCALC)

   $\Rightarrow \vdash A \Rightarrow [P \Rightarrow P']_P$ (PROPCALC)

   $\Rightarrow \vdash \Box A \Rightarrow \Box[P \Rightarrow P']_P$ (TLA2)

2. $(\vdash P \land A \Rightarrow P') \Rightarrow (\vdash P \land \Box A \Rightarrow P \land \Box[P \Rightarrow P']_P)$

   Pf: 1 and PROPCALC.

3. QED

   Pf: 2, TLA1, and PROPCALC.

Lemma PRETACT: $A, B \in TACT \Rightarrow A \land B \in TACT$

Pf: ...

PROG: For $P, Q \in PRED$, $A \in TLACT$, $B \in \neg TLACT$:

If $\vdash P \land A \Rightarrow P'$ and $\vdash P \land A \land B \Rightarrow Q'$, then

$\vdash \Box A \land \Box\Diamond B \Rightarrow (P \leadsto Q)$

Assume: A. $\vdash P \land A \Rightarrow P'$

B. $\vdash P \land A \land B \Rightarrow Q'$

To Prove: $\vdash \Box A \land \Box\Diamond B \Rightarrow (P \leadsto Q)$

Note: I will use PROPCALC without explicit mention.

1. $\vdash \neg Q \land [\neg Q \Rightarrow \neg Q']_{\neg Q} \Rightarrow \neg Q'$

   Pf: By RC, and

   $\neg Q \land [\neg Q \Rightarrow \neg Q']_{\neg Q}$

   $\equiv \neg Q \land ((\neg Q \Rightarrow \neg Q') \lor (\neg Q = \neg Q'))$

   $\equiv \neg Q \land (\neg Q \equiv \neg Q')$

   $\equiv \neg Q' \land (\neg Q \equiv \neg Q')$

   $\Rightarrow \neg Q'$

2. QED

   Pf: $\vdash P \land A \land B \Rightarrow Q'$ (Assumption B)

   $\Rightarrow \vdash \neg Q' \land P \land A \Rightarrow \neg B$

   $\Rightarrow \vdash \neg Q \land [\neg Q \Rightarrow \neg Q']_{\neg Q} \land P \land A \Rightarrow \neg B$ (1)

   $\Rightarrow \vdash \Box(\neg Q \land [\neg Q \Rightarrow \neg Q']_{\neg Q} \land P \land A) \Rightarrow \Box\neg B$

   (TLA2 and Lemma PRETACT)

   $\Rightarrow \vdash \Box(\neg Q) \land \Box[\neg Q \Rightarrow \neg Q']_{\neg Q} \land \Box(P \land A) \Rightarrow \Box\neg B$

   (STL1)

   $\Rightarrow \vdash \Box(\neg Q) \land \Box P \land \Box A \Rightarrow \Box\neg B$ (TLA1 and STL8)

   $\Rightarrow \vdash \Box P \land \Box A \land \Diamond B \Rightarrow \Diamond Q$ (def of $\Diamond$)

   $\Rightarrow \vdash \Box P \land \Box A \land \Box\Diamond B \Rightarrow \Diamond Q$ (STL8)

   $\Rightarrow \vdash P \land \Box A \land \Box\Diamond B \Rightarrow \Diamond Q$ (Assumption B, INV)

   $\Rightarrow \vdash \Box A \land \Box\Diamond B \Rightarrow (P \Rightarrow \Diamond Q)$

   $\Rightarrow \vdash \Box(A \land \Diamond B) \Rightarrow (P \Rightarrow \Diamond Q)$ (STL1, def of $\leadsto$)

STL3. $F \vdash \Box F$

Pf: From STL7, with $F \rightarrow TRUE$, $G \rightarrow F$, STL0, and PROPCALC.

NOTE ADDED 17 Nov 93: I think this proof is wrong because we can't deduce $\vdash \Box true$ from STL0 and PROPCALC. I think the best fix is to change the conclusion of STL0 to $\Box F$.

STL5. $\vdash \Box\Diamond F \lor \Box\Diamond G \equiv \Box(\Diamond F \lor \Diamond G)$

Pf: $\Box\Diamond F \lor \Box\Diamond G$

$\equiv \neg(\neg\Box\Diamond F \land \neg\Box\Diamond G)$ (PROPCALC)

$\equiv \neg(\Diamond\Box\neg F \land \Diamond\Box\neg G)$ (Def of $\Diamond$)

$\equiv \neg(\Diamond\Box(\neg F \land \neg G)$ (STL4)

$\equiv \neg(\Diamond\Box\neg(F \lor G)$ (PROPCALC)

$\equiv \Box(\Diamond F \lor \Diamond G)$ (Def of $\Diamond$)

STL9. $\vdash \Box F \land \Box\Diamond G \Rightarrow \Diamond(F \land G)$

PF: $\Box F \land \Box\Diamond G$

$\Rightarrow \Box\Diamond F \land \Box\Diamond G$ STL14)

$\Rightarrow \Box\Diamond(F \land G)$ (STL5)

$\Rightarrow \Diamond(F \land G)$ (STL8)

STL10. $F \Rightarrow G \vdash \Diamond F \Rightarrow \Diamond G$

Pf: Assume: $\vdash F \Rightarrow G$

To Prove: $\vdash \Diamond F \Rightarrow \Diamond G$

1. $\vdash \neg G \Rightarrow \neg F$
   Pf: PROPCALC
2. $\vdash \Box\neg G \Rightarrow \Box\neg F$
   Pf: 1 and STL7
3. $\vdash \neg\Box\neg F \Rightarrow \neg\Box\neg G$
   Pf: PROPCALC
4. QED
   Pf: 3 and def of $\Diamond$

STL11. $((P \land F) \Rightarrow \Diamond Q) \vdash \Box F \Rightarrow (P \leadsto Q)$

Pf: $\vdash P \land F \Rightarrow \Diamond Q$

$\Rightarrow \vdash F \Rightarrow (P \Rightarrow \Diamond Q)$ (PROPCALC)

$\Rightarrow \vdash \Box F \Rightarrow (P \leadsto Q)$ (STL7 and Def of $\leadsto$)

STL12. $F \Rightarrow G \vdash F \leadsto G$

Pf: $\vdash F \Rightarrow G$

$\Rightarrow \vdash \Box(F \Rightarrow G)$ (STL8)

$\Rightarrow \vdash \Box(F \Rightarrow \Diamond G)$ (STL8)

$\equiv \vdash F \leadsto G$ (Def of $\leadsto$)

STL13. $\vdash (F \leadsto G) \land (G \leadsto H) \Rightarrow (F \leadsto H)$

Pf: ...

STL14. $\vdash F \Rightarrow \Diamond F$

Pf: By STL8, with F replaced by $\neg F$.

STL15. $\vdash \Diamond\Diamond F \equiv \Diamond F$

Pf: By STL2, with F replaced by $\neg F$,

STL16. $\vdash \Diamond\Box\neg TRUE \equiv \neg TRUE$

$\vdash \Box\Diamond TRUE \equiv TRUE$

STL17. $\vdash \Box FALSE \equiv FALSE$

STL18. $(\vdash F \Rightarrow G) \Rightarrow \vdash F \leadsto G$

The following result asserts that these rules are all sound.

Proposition SOUND: $\forall F \in TLA : (\vdash F) \Rightarrow (\models F)$

Proof: Obvious (?!).

Classes of TLA Formulas

We now define some classes of formulas. We introduce the obvious notation by which, for any sets $\mathcal{F}$ and $\mathcal{G}$ of formulas, $\mathcal{F} \Rightarrow \mathcal{G}$ denotes the set of all formulas $F \Rightarrow G$ with $F \in \mathcal{F}$ and $G \in \mathcal{G}$. Thus, for example $\mathcal{F} \lor \mathcal{F}$ denotes all formulas $F \lor G$ with F and G in $\mathcal{F}$, which is not the same as $\mathcal{F}$.

For any set $\mathcal{F}$ of formulas, $\mathcal{F}^*$ is defined to be the set of finite conjunctions of formulas in $\mathcal{F}$. More precisely,

$\mathcal{F}^0 \triangleq \{TRUE\}$

$\mathcal{F}^{i+1} \triangleq \mathcal{F}^i \land \mathcal{F}$, for $i \geq 0$

$\mathcal{F}^* \triangleq \exists i \geq 0 : \mathcal{F}^i$

In TLA, a program is represented by a formula F of the form

$$P \wedge \Box[N]_x \wedge WF_1 \wedge \ldots \wedge WF_m \wedge SF_1 \wedge \ldots \wedge SF_n$$

$$\begin{aligned} \text{where } WF_i &= \Box\Diamond\langle A_i\rangle_x \vee \Box\Diamond\neg Enabled(\langle A_i\rangle_x) \\ SF_i &= \Box\Diamond\langle B_i\rangle_x \vee \Diamond\Box\neg Enabled(\langle B_i\rangle_x) \\ A_i &\Rightarrow N \\ B_i &\Rightarrow N \end{aligned}$$

and x is a k-tuple of program variables. Such a formula F is "machine closed", meaning that for any formula G that represents a safety property, $\models F \Rightarrow G$ iff $\models P \wedge \Box[N]_x \Rightarrow G$.

All TLA properties that we prove have the form $F \Rightarrow G$ for such an F. To prove that one program implies another, we prove a formula $F \Rightarrow G$ where F is a formula representing a program, and G is obtained from a formula representing a program by substituting state functions for the program variables. Because of this substitution, G need not be machine closed.

There are thus two classes of "program formulas" of interest, machine-closed formulas of the class defined above, and the more general class of formulas obtainable from machine-closed formulas by substituting state functions for variables. We abstract and generalize these two sets of formulas by the formulas PGM and MCPGM.

We define the class PGM by

$$PGM \triangleq PRED \wedge \Box TACT \wedge WF^* \wedge SF^*$$

$$\begin{aligned} \text{where } WF &\triangleq \Box\Diamond\neg TACT \vee \Box\Diamond\neg TACT \\ SF &\triangleq \Box\Diamond\neg TACT \vee \Diamond\Box TACT \end{aligned}$$

We define MCPGM to be the subset of PGM consisting of all formulas $F \in PGM$ satisfying the following condition

$$\begin{aligned} &\exists\, G \in PRED \wedge \Box TACT : \\ &\quad \wedge\ (\vdash F \Rightarrow G) \\ &\quad \wedge\ \forall\, A \in TACT : (\models F \Rightarrow \Box A) \Rightarrow (\models G \Rightarrow \Box A) \end{aligned}$$

The Completeness Theorem

A logic is relatively complete for a formula set of formulas $\mathcal{F}$ iff, for every formula $F \in \mathcal{F}$, if F is valid then it is provable. We write this as Comp($\mathcal{F}$), defined by

$$Comp(\mathcal{F}) \triangleq \forall\, F \in \mathcal{F} : (\models F) \Rightarrow (\vdash F)$$

The relative completeness assumption RC asserts Comp(ACT). Our completeness result is:

THEOREM:

1. $Comp(MCPGM \Rightarrow PRED)$
2. $Comp(MCPGM \Rightarrow \Box TACT)$
3. $Comp(MCPGM \Rightarrow (PRED \leadsto PRED))$
4. $Comp(MCPGM \Rightarrow PGM)$

THE PROOF

The Relation $[\!=$

We define a "provable subset" relation $[\!=$ among sets of formulas, where $\mathcal{F}$ $[\!=$ $\mathcal{G}$ means that every formula in $\mathcal{F}$ is provably equivalent to a formula in $\mathcal{G}$.

$$\mathcal{F}\ [\!=\ \mathcal{G} \ \stackrel{\Delta}{=}\ \forall\ \mathrm{F} \in \mathcal{F} : \exists\ \mathrm{G} \in \mathcal{G} : \ \vdash\ (\mathrm{F} \equiv \mathrm{G})$$

$$\mathcal{F}\ [\!=\!]\ \mathcal{G} \ \stackrel{\Delta}{=}\ (\mathcal{F}\ [\!=\ \mathcal{G}) \ \wedge\ (\mathcal{G}\ [\!=\ \mathcal{F})$$

Lemma PSUB: For any subsets $\mathcal{F}$, $\mathcal{F}_i$, and $\mathcal{G}_i$ of TLA:

1. $\mathcal{F}\ [\!=\ \mathcal{F}$
2. $(\mathcal{F}_1\ [\!=\ \mathcal{F}_2)\ \wedge\ (\mathcal{F}_2\ [\!=\ \mathcal{F}_3) \Rightarrow (\mathcal{F}_1\ [\!=\ \mathcal{F}_3)$
3. $(\mathcal{F}_1\ [\!=\ \mathcal{G}_1)\ \wedge\ (\mathcal{F}_2\ [\!=\ \mathcal{G}_2)$
   $\Rightarrow\ \wedge\ \neg\mathcal{F}_1\ [\!=\ \neg\mathcal{G}_1$
   $\wedge\ (\mathcal{F}_1\ \wedge\ \mathcal{F}_2)\ [\!=\ (\mathcal{G}_1\ \wedge\ \mathcal{G}_2)$
   $\wedge\ (\mathcal{F}_1\ \Rightarrow\ \mathcal{F}_2)\ [\!=\ (\mathcal{G}_1\ \Rightarrow\ \mathcal{G}_2)$
4. $(\forall\ i \geq 0 : \mathcal{F}_1\ [\!=\ \mathcal{G}_i) \Rightarrow \mathrm{F}^*\ [\!=\ \mathrm{G}^*$

Pf: This follows easily from PROPCALC.

Lemma PSUBCOMP and ANDCOMP: For any subsets $\mathcal{F}$ and $\mathcal{G}$ of TLA:

0. $(\mathcal{F}\ [\!=\ \mathcal{G})\ \wedge\ \mathrm{Comp}(\mathcal{G}) \Rightarrow \mathrm{Comp}(\mathcal{F})$.
1. $\mathrm{Comp}(\mathcal{F}) \equiv \mathrm{Comp}(\mathcal{F}^*)$
2. $\{\mathrm{TRUE}\}\ [\!=\ \mathcal{G} \Rightarrow (\mathrm{Comp}(\mathcal{F} \Rightarrow \mathcal{G}^*) \equiv \mathrm{Comp}(\mathcal{F} \Rightarrow \mathcal{G}))$

Pf of 0:

$(\mathcal{F}\ [\!=\ \mathcal{G})\ \wedge\ \mathrm{Comp}(\mathcal{G})$
$\quad \equiv\ \wedge\ \forall\ \mathrm{F} \in \mathcal{F} : \exists\ \mathrm{G} \in \mathcal{G} : \ \vdash\ \mathrm{F} \equiv \mathrm{G}$
$\quad\quad \wedge\ \forall\ \mathrm{G} \in \mathcal{G} : \ \models\ \mathrm{G} \Rightarrow\ \vdash\ \mathrm{G}$
$\quad \Rightarrow \forall\ \mathrm{F} \in \mathcal{F} : \exists\ \mathrm{G} \in \mathcal{G} :$
$\quad\quad\quad (\vdash\ \mathrm{F} \equiv \mathrm{G})\ \wedge\ (\models\ \mathrm{G} \Rightarrow\ \vdash\ \mathrm{G})$
$\quad \Rightarrow \forall\ \mathrm{F} \in \mathcal{F} : \ \models\ \mathrm{F} \Rightarrow$
$\quad\quad \exists\ \mathrm{G} \in \mathcal{G} : (\vdash\ \mathrm{F} \equiv \mathrm{G})\ \wedge\ (\models\ \mathrm{G} \Rightarrow\ \vdash\ \mathrm{G})$
$\quad \Rightarrow \forall\ \mathrm{F} \in \mathcal{F} : \ \models\ \mathrm{F} \Rightarrow$
$\quad\quad \exists\ \mathrm{G} \in \mathcal{G} : (\vdash\ \mathrm{F} \equiv \mathrm{G})\ \wedge\ \vdash\ \mathrm{G}$ (Prop SOUND)
$\quad \Rightarrow \forall\ \mathrm{F} \in \mathcal{F} : \ \models\ \mathrm{F} \Rightarrow\ \vdash\ \mathrm{F}$ (PROPCALC)
$\quad \equiv \mathrm{Comp}(\mathcal{F})$

Pf of 1:

1.1. $\mathrm{Comp}(\mathcal{F}^*) \Rightarrow \mathrm{Comp}(\mathcal{F})$
  Pf: By part 0, since $\mathcal{F}\ [\!=\ \mathcal{F}^*$
1.2. $\mathrm{Comp}(\mathcal{F}) \Rightarrow \mathrm{Comp}(\mathcal{F}^*)$
  1.2.1. Comp({TRUE})
    Pf: PROPCALC implies $\vdash$ TRUE.
  1.2.2. For $i > 0$, $\mathrm{Comp}(\mathcal{F})\ \wedge\ \mathrm{Comp}(\mathcal{F}^i) \Rightarrow \mathrm{Comp}(\mathcal{F}^{i+1})$
    Assume: $\mathrm{Comp}(\mathcal{F})$, $\mathrm{Comp}(\mathcal{F}^i)$, $\mathrm{F} \in \mathcal{F}$, and $\mathrm{G} \in \mathcal{F}^i$
    To Prove: $\models\ \mathrm{F}\ \wedge\ \mathrm{G} \Rightarrow\ \vdash\ \mathrm{F}\ \wedge\ \mathrm{G}$
    Pf: $\models\ \mathrm{F}\ \wedge\ \mathrm{G}$
      $\Rightarrow\ \models\ \mathrm{F}\ \wedge\ \models\ \mathrm{G}$ (def of $\models$)
      $\Rightarrow\ \vdash\ \mathrm{F}\ \wedge\ \vdash\ \mathrm{G}$ (assumption)
      $\Rightarrow\ \vdash\ \mathrm{F}\ \wedge\ \mathrm{G}$ (PROPCALC)
  1.2.3. QED
    Pf: 1.2.1, 1.2.2, Induction, and definition of $\mathcal{F}^*$.
1.3. QED

Pf of 2:

2.1. $\mathrm{Comp}(\mathcal{F} \Rightarrow \mathcal{G}) \Rightarrow \mathrm{Comp}(\mathcal{F} \Rightarrow \mathcal{G}^*)$
  Pf: By Part 0, since $(\mathcal{F} \Rightarrow \mathcal{G}^*)\ [\!=\ (\mathcal{F} \Rightarrow \mathcal{G})$.
2.2. $(\{\mathrm{TRUE}\}\ [\!=\ \mathcal{G})\ \wedge\ \mathrm{Comp}(\mathcal{F} \Rightarrow \mathcal{G}^*) \Rightarrow \mathrm{Comp}(\mathcal{F} \Rightarrow \mathcal{G})$

2.2.1. $(\{\mathtt{TRUE}\} \;[\!\!= \mathcal{G}) \Rightarrow$
$(\mathcal{F} \Rightarrow \mathcal{G}^0) \;[\!\!= (\mathcal{F} \Rightarrow \mathcal{G})$

2.2.2. $(\mathcal{F} \Rightarrow \mathcal{G}^i) \;[\!\!= (\mathcal{F} \Rightarrow \mathcal{G})^i$
$\Rightarrow (\mathcal{F} \Rightarrow \mathcal{G}^{i+1}) \;[\!\!= (\mathcal{F} \Rightarrow \mathcal{G})^{i+1}$

Pf: By PROPCALC, which implies
$\vdash (F \Rightarrow (G_1 \land G_2)) \equiv (F \Rightarrow G_1) \land (F \Rightarrow G_2)$

2.2.3. $(\{\mathtt{TRUE}\} \;[\!\!= \mathcal{G}) \Rightarrow ((\mathcal{F} \Rightarrow \mathcal{G}^*) \;[\!\!= (\mathcal{F} \Rightarrow \mathcal{G})^*)$

Pf: 2.2.1, 2.2.2, and mathematical induction.

2.2.4. $(\{\mathtt{TRUE}\} \;[\!\!= \mathcal{G}) \land \mathrm{Comp}((\mathcal{F} \Rightarrow \mathcal{G})^*) \Rightarrow \mathrm{Comp}(\mathcal{F} \Rightarrow \mathcal{G}^*)$

Pf: 2.2.3 and Part 0.

2.2.5. QED

Pf: 2.2.4 and Part 1, with $(\mathcal{F} \Rightarrow \mathcal{G})$ substituted for $\mathcal{F}$.

Lemma TACT: $\mathrm{TACT}^* \;[\!\!= \mathrm{TACT}$

Proof: Use Lemma PRETACT.

1. For any A and B in ACT and any f and g in SFCN:
   (a) $[(A \lor (f' = f)) \land (B \lor (g' = g)]_{(f,g)} \in \mathrm{TACT}$
   (b) $\models ([A]_f \land [B]_g$
   $\equiv [(A \lor (f' = f)) \land (B \lor (g' = g)]_{(f,g)})$

   Pf: Part (a) follows from EX2, EX10, and EX11. Part (b) follows, by a simple calculation, from the definition of validity for actions.
2. QED

   Pf: Follows easily by induction from 1 and EX2.

Reduction and Completeness

Lemma REDDEF: For any subsets $\mathcal{F}$ and $\mathcal{G}$ of TLA:
$\forall F \in \mathcal{F} : \exists G \in \mathcal{G} : \land (\models F) \Rightarrow (\models G)$
$\land (\vdash G) \Rightarrow (\vdash F)$
$\Rightarrow (\mathrm{Comp}(\mathcal{G}) \Rightarrow \mathrm{Comp}(\mathcal{F}))$

Pf: Trivial.

Lemma COMPRED:
$\mathrm{Comp}(\mathcal{F}) \land \mathrm{Comp}(\mathcal{G}) \Rightarrow \mathrm{Comp}(\mathcal{F} \land \mathcal{G})$

Pf: Trivial.

Lemma BOXRED: For any sets $\mathcal{F}$ and $\mathcal{G}$ of formulas in TLA:
$\mathrm{Comp}(\Box\mathcal{F} \Rightarrow \mathcal{G}) \Rightarrow \mathrm{Comp}(\Box\mathcal{F} \Rightarrow \Box\mathcal{G})$

Assume: A. $\mathrm{Comp}(\Box\mathcal{F} \Rightarrow \mathcal{G})$
B. $F \in \mathcal{F}$, $G \in \mathcal{G}$, and $\models \Box F \Rightarrow \Box G$

To Prove: $\vdash \Box F \Rightarrow \Box G$

1. $\models \Box F \Rightarrow G$

   Pf: Assumption B, STL 8 and Lemma SOUND.
2. $\vdash (\Box F \Rightarrow G) \Rightarrow \vdash (\Box F \Rightarrow \Box G)$

   Pf: STL7 and STL2.
3. QED

   Pf: 1, 2, and assumptions.

Lemma SIN: For any $P \in \mathrm{PRED}$, $A \in \mathrm{TACT}$, $G \in \mathrm{TLA}$, and $F \in \mathrm{WF}^* \land \mathrm{SF}^*$:

1. $\wedge \models (\mathtt{P} \wedge \Box \mathtt{A} \wedge \mathtt{F} \Rightarrow \Box \mathtt{G}) \Rightarrow \models \Box(\mathtt{sin(A, P)} \wedge \Box \mathtt{A} \wedge \mathtt{F} \Rightarrow \Box \mathtt{G})$
2. $\wedge \vdash \Box(\mathtt{sin(A, P)} \wedge \Box \mathtt{A} \wedge \mathtt{F} \Rightarrow \Box \mathtt{G}) \Rightarrow \vdash (\mathtt{P} \wedge \Box \mathtt{A} \wedge \mathtt{F} \Rightarrow \Box \mathtt{G})$

Pf: LET $\mathtt{I} \triangleq \mathtt{sin(A, P)}$
Assume $\mathtt{P} \in \mathtt{PRED}$, $\mathtt{A} \in \mathtt{TACT}$, ...

0. For all $\tau \in \mathtt{ST}^{\omega}$ and $\mathtt{n} \geq 0$:
$(\tau \models \mathtt{F}) \equiv (\tau^{+\mathtt{n}} \models \mathtt{F})$
Pf: Assume: $\tau \in \mathtt{ST}^{\omega}$, $\mathtt{n} \geq 0$,
0.1. $\forall\, \mathtt{H} \in \Box\Diamond\neg\mathtt{TACT} \vee \Diamond\Box\mathtt{TACT}$:
$(\tau \models \mathtt{H}) \equiv (\tau^{+\mathtt{n}} \models \mathtt{H})$
Pf: Follows easily from definitions.
0.2. $\forall\, \mathtt{H} \in \Box\Diamond\neg\mathtt{TACT} \vee \Box\Diamond\mathtt{TACT}$:
$(\tau \models \mathtt{H}) \equiv (\tau^{+\mathtt{n}} \models \mathtt{H})$
Pf: Follows easily from definitions.
0.3. QED
Pf: By 0.1 and 0.2, since $\sigma \models \mathtt{H}_1 \wedge \mathtt{H}_2$ equals
$\sigma \models \mathtt{H}_1 \wedge \sigma \models \mathtt{H}_2$ by def of $\models$.

1. $\vdash \mathtt{P} \wedge \Box \mathtt{A} \Rightarrow \Box \mathtt{I}$
1.1. $\vdash \mathtt{P} \Rightarrow \mathtt{I}$
Pf: $\models \mathtt{P} \Rightarrow \mathtt{I}$ and Relative Completeness Assumption 2.
1.2. $\vdash \mathtt{I} \wedge \mathtt{A} \Rightarrow \mathtt{I}'$
1.2.1. ...
1.3. QED
Rule INV

2. $\forall\, \sigma \in \mathtt{ST}^{\omega}$:
$\sigma \models \Box \mathtt{I} \wedge \Box \mathtt{A} \wedge \mathtt{F}$
$\Rightarrow \exists\, \tau, \mathtt{n} : \wedge\ \sigma = \tau^{+\mathtt{n}}$
$\wedge\ \tau \models \mathtt{P} \wedge \Box \mathtt{A} \wedge \mathtt{F}$
Assume: 3A. $\sigma \models \Box \mathtt{I} \wedge \Box \mathtt{A} \wedge \mathtt{F}$
To Prove: $\exists\, \tau, \mathtt{n} : \wedge\ \sigma = \tau^{+\mathtt{n}}$
$\wedge\ \tau \models \mathtt{P} \wedge \Box \mathtt{A} \wedge \mathtt{F}$
2.1. $\sigma_0.\mathtt{I}$
Pf: by assumption, definition of $\models$ and $\Box$.
2.2. Choose τ_0, ..., $\tau_\mathtt{n}$ such that
$\tau_0.\mathtt{P}$, $\tau_{\mathtt{i-1}}.\mathtt{A}.\tau_\mathtt{i}$,
and $\tau_\mathtt{n} = \sigma_0$.
Pf: 2.1, definition of sin(P, A).
2.3. Let $\tau_{\mathtt{n+i}} \triangleq \sigma_\mathtt{i}$, then
$\tau \models \mathtt{P} \wedge \Box \mathtt{A}$
Pf: By 2.2 and assumption 3A.
2.4. $\tau \models \mathtt{F}$
Pf: $\sigma \models \mathtt{F}$ by assumption 3A, and
$\tau \models \mathtt{F}$ follows by 0.
2.5. QED
Pf: 2.3, 2.4.

3. $\models (\mathtt{P} \wedge \Box \mathtt{A} \wedge \mathtt{F} \Rightarrow \Box \mathtt{G})$
$\Rightarrow \models (\Box \mathtt{I} \wedge \Box \mathtt{A} \wedge \mathtt{F} \Rightarrow \Box \mathtt{G})$
3.1. $\forall\, \sigma$:
$\wedge \models (\mathtt{P} \wedge \Box \mathtt{A} \wedge \mathtt{F} \Rightarrow \Box \mathtt{G})$
$\wedge\ \sigma \models (\Box \mathtt{I} \wedge \Box \mathtt{A} \wedge \mathtt{F})$
$\Rightarrow \sigma \models \Box \mathtt{G}$

Assume: 4A.1. $\models (\mathtt{P} \wedge \Box \mathtt{A} \wedge \mathtt{F} \Rightarrow \Box \mathtt{G})$
4A.2. $\sigma \models (\Box \mathtt{I} \wedge \Box \mathtt{A} \wedge \mathtt{F})$
To Prove: $\sigma \models \Box \mathtt{G}$

```
        3.1.1. Choose τ s.t.
                 1.∧  σ = τ^{+n}
                 2.∧  τ ⊨ P ∧ □A ∧ F
               Pf: By 2.
        3.1.2. τ ⊨ □G
               Pf: By assumption 4A.1, 3.1.1.2, and definition
                   of ⊨.
        3.1.3. QED
               By 3.1.2, 3.1.1.1, and definition of ⊨ □G.

     3.2. QED
          By 3.1, since
          ⊨ (□I ∧ □A ∧ F ⇒ □G) ≜
            ∀ σ : σ ⊨ (□I ∧ □A ∧ F)
                ⇒ σ ⊨ □G

  4. ⊢ (□I ∧ □A ∧ F ⇒ □G) ⇒ ⊢ (P ∧ □A ∧ F ⇒ □G)
     Pf: 1 and PROPCALC.

  5. QED
     3 and 4.
```

Lemma SINRED: For any set $\mathcal{F}$ of formulas in TLA:
Comp(□TACT ∧ WF* ∧ SF* ⇒ □$\mathcal{F}$) ⇒ Comp(PGM ⇒ □$\mathcal{F}$)

Pf: Lemma SIN and Lemma REDDEF.

```
Lemma R1.  ∧ Comp(MCPGM ⇒ PRED)
           ∧ Comp(MCPGM ⇒ □TACT)
           ∧ Comp(MCPGM ⇒ WF)
           ∧ Comp(MCPGM ⇒ SF)
             ⇒ Comp(MCPGM ⇒ PGM)
Pf: 1. ∧ Comp(MCPGM ⇒ PRED)
       ∧ Comp(MCPGM ⇒ □TACT)
       ∧ Comp(MCPGM ⇒ WF*)
       ∧ Comp(MCPGM ⇒ SF*)
       ⇒ Comp(MCPGM ⇒ PGM)
      Pf: Lemma COMPRED and definition of PGM.
    2. ∧ Comp(MCPGM ⇒ WF) ⇒ Comp(MCPGM ⇒ WF*)
       ∧ Comp(MCPGM ⇒ SF) ⇒ Comp(MCPGM ⇒ SF*)
       Pf: By Lemma ANDCOMP, since STL16 implies
           {TRUE} [= WF and {TRUE} [= SF.
    3. QED
       Pf: 1 and 2.
```

```
Lemma WFRED: WF [= SF
 1. F ∈ WF ≡ ∃ A, B ∈ TACT : F = □◇¬A ∨ □◇¬B
    Pf: Def of WF.

 2. ∀ A, B ∈ TACT :
      ⊢ (□◇¬A ∨ □◇¬B) ≡ (□◇¬(A ∧ B) ∨ ◇□¬TRUE)
      Pf: STL5,, STL16, and PROPCALC.

 3. F ∈ WF ⇒ ∃ G ∈ SF : ⊢ F ≡ G
    Pf: 1, 2, EX2, EX-1, and def of SF.

 4. QED
    Pf: 3 and def of [=
```

Lemma IORED: For any sets $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ of formulas in TLA:
Comp(□$\mathcal{F}^*$ ∧ □$\mathcal{G}$ ⇒ □$\mathcal{H}$) ⇒

Comp($\Box\,\mathcal{F}^* \;\wedge\; (\Diamond\Box\,\mathcal{F})^* \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box\,\mathcal{H}$)

1. $\Box\,\mathcal{F}^* \;\wedge\; (\Diamond\Box\,\mathcal{F})^* \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box\,\mathcal{H}$

$\mathrel{[\!=}\; \Box(\mathcal{F}^*) \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box(\Diamond\neg(\mathcal{F}^*) \;\wedge\; \Box\,\mathcal{H})$ (STL1 and STL6)

Pf: $\Box\,\mathcal{F}^* \;\wedge\; (\Diamond\Box\,\mathcal{F})^* \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box\,\mathcal{H}$

$\mathrel{[\!=}\; \Box(\mathcal{F}^*) \;\wedge\; \Diamond\Box(\mathcal{F}^*) \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box\,\mathcal{H}$ (STL4)

$\mathrel{[\!=\!]}\; \Box(\mathcal{F}^*) \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; (\Box\Diamond\neg(\mathcal{F}^*) \;\wedge\; \Box\,\mathcal{H})$ (PROPCALC)

2. $\Box(\mathcal{F}^*) \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Diamond\neg(\mathcal{F}^*) \;\wedge\; \Box\,\mathcal{H}$

$\mathrel{[\!=}\; \Box(\mathcal{F}^*) \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box\,\mathcal{H}$

Pf: $\Box(\mathcal{F}^*) \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Diamond\neg(\mathcal{F}^*) \;\wedge\; \Box\,\mathcal{H}$

$\mathrel{[\!=\!]}\; \Box(\mathcal{F}^*) \;\wedge\; \Box(\mathcal{F}^*) \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box\,\mathcal{H}$ (PROPCALC)

$\mathrel{[\!=\!]}\; \Box(\mathcal{F}^*) \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box\,\mathcal{H}$ (STL1)

3. QED

Pf: Comp($\Box\,\mathcal{F}^* \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box\,\mathcal{H}$)

$\Rightarrow$ Comp($\Box\,\mathcal{F}^* \;\wedge\; (\Diamond\Box\,\mathcal{F})^* \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box\,\mathcal{H}$)

(2 and PSUBCOMP.0)

$\Rightarrow$ Comp($\Box(\mathcal{F}^*) \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Diamond\neg(\mathcal{F}^*) \;\wedge\; \Box\,\mathcal{H}$)

(1 and PSUBCOMP.0)

$\Rightarrow$ Comp($\Box(\mathcal{F}^*) \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box(\Diamond\neg(\mathcal{F}^*) \;\wedge\; \Box\,\mathcal{H})$)

(STL1 and Lemma BOXRED)

$\Rightarrow$ Comp($\Box\,\mathcal{F}^* \;\wedge\; (\Diamond\Box\,\mathcal{F})^* \;\wedge\; \Box\mathcal{G} \;\Rightarrow\; \Box\,\mathcal{H}$)

1 and Lemma PSUBCOMP.0.

Lemma PGMRED: For any set $\mathcal{F}$ of formulas in TLA:

Comp($\Box\,\mathrm{TACT} \;\wedge\; (\Box\Diamond\neg\mathrm{TACT})^* \;\Rightarrow\; \Box\,\mathcal{F}$) $\Rightarrow$ Comp(PGM $\Rightarrow\; \Box\,\mathcal{F}$)

1. For any subsets $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{I}$ of TLA:

$\mathcal{F} \;\wedge\; (\mathcal{G} \vee \mathcal{H})^* \Rightarrow \mathcal{I}$

$\mathrel{[\!=}\; (\mathcal{F} \;\wedge\; \mathcal{G}^* \;\wedge\; \mathcal{H}^* \Rightarrow \mathcal{I})^*$

1.1. $\mathcal{F} \;\wedge\; (\mathcal{G} \vee \mathcal{H})^0 \Rightarrow \mathcal{I}$

$\mathrel{[\!=}\; (\mathcal{F} \;\wedge\; \mathcal{G}^* \;\wedge\; \mathcal{H}^*)^* \Rightarrow \mathcal{I})^*$

Pf: Trivial

1.2. If $\mathcal{F} \;\wedge\; (\mathcal{G} \vee \mathcal{H})^{\mathrm{i}} \Rightarrow \mathcal{I}$

$\mathrel{[\!=}\; (\mathcal{F} \;\wedge\; \mathcal{G}^* \;\wedge\; \mathcal{H}^* \Rightarrow \mathcal{I})^*$

then $\mathcal{F} \;\wedge\; (\mathcal{G} \vee \mathcal{H})^{\mathrm{i}+1} \Rightarrow \mathcal{I}$

$\mathrel{[\!=}\; (\mathcal{F} \;\wedge\; \mathcal{G}^* \;\wedge\; \mathcal{H}^* \Rightarrow \mathcal{I})^*$

Pf: $\mathcal{F} \;\wedge\; (\mathcal{G} \vee \mathcal{H})^{\mathrm{i}+1} \Rightarrow \mathcal{I}$

$\mathrel{[\!=\!]}\; \mathcal{F} \;\wedge\; (\mathcal{G} \vee \mathcal{H})^{\mathrm{i}} \;\wedge\; (\mathcal{G} \vee \mathcal{H}) \Rightarrow \mathcal{I}$

$\mathrel{[\!=\!]}\; \wedge\; \mathcal{G} \Rightarrow (\mathcal{F} \;\wedge\; (\mathcal{G} \vee \mathcal{H})^{\mathrm{i}} \Rightarrow \mathcal{I})$

$\wedge\; \mathcal{H} \Rightarrow (\mathcal{F} \;\wedge\; (\mathcal{G} \vee \mathcal{H})^{\mathrm{i}} \Rightarrow \mathcal{I})$

$\mathrel{[\!=}\; \wedge\; \mathcal{G} \Rightarrow (\mathcal{F} \;\wedge\; \mathcal{G}^* \;\wedge\; \mathcal{H}^* \Rightarrow \mathcal{I})^*$

$\wedge\; \mathcal{H} \Rightarrow (\mathcal{F} \;\wedge\; \mathcal{G}^* \;\wedge\; \mathcal{H}^* \Rightarrow \mathcal{I})^*$

$\mathrel{[\!=}\; \wedge\; (\mathcal{G} \Rightarrow (\mathcal{F} \;\wedge\; \mathcal{G}^* \;\wedge\; \mathcal{H}^* \Rightarrow \mathcal{I}))^*$

$\wedge\; (\mathcal{H} \Rightarrow (\mathcal{F} \;\wedge\; \mathcal{G}^* \;\wedge\; \mathcal{H}^* \Rightarrow \mathcal{I}))^*$

$\mathrel{[\!=}\; \wedge\; (\mathcal{F} \;\wedge\; \mathcal{G}^* \;\wedge\; \mathcal{H}^* \Rightarrow \mathcal{I})^*$

$\wedge\; (\mathcal{F} \;\wedge\; \mathcal{G}^* \;\wedge\; \mathcal{H}^* \Rightarrow \mathcal{I})^*$

$\mathrel{[\!=\!]}\; (\mathcal{F} \;\wedge\; \mathcal{G}^* \;\wedge\; \mathcal{H}^* \Rightarrow \mathcal{I})^*$

1.3. QED

Pf: 1.1, 1.2, and mathematical induction.

2. $\Box\,\mathrm{TACT} \;\wedge\; \mathrm{WF}^* \;\wedge\; \mathrm{SF}^* \;\Rightarrow\; \Box\,\mathcal{F}$

$\mathrel{[\!=}\; (\Box\,\mathrm{TACT} \;\wedge\; (\Box\Diamond\neg\mathrm{TACT})^* \;\Rightarrow\; \Box\,\mathcal{F})$

Pf: $\Box\,\mathrm{TACT} \;\wedge\; \mathrm{WF}^* \;\wedge\; \mathrm{SF}^* \;\Rightarrow\; \Box\,\mathcal{F}$

$\mathrel{[\!=}\; (\Box\,\mathrm{TACT} \;\wedge\; (\Box\Diamond\neg\mathrm{TACT})^* \;\wedge\; \mathrm{SF}^* \;\Rightarrow\; \Box\,\mathcal{F})$

(By 1, with $\mathcal{G} = \mathcal{H} = \Box\Diamond\neg\mathrm{TACT}$)

$\mathrel{[\!=}\; (\Box\,\mathrm{TACT} \;\wedge\; (\Box\Diamond\neg\mathrm{TACT})^* \;\wedge\; (\Diamond\Box\,\mathrm{TACT})^* \;\Rightarrow\; \Box\,\mathcal{F})$

(By 1, with $\mathcal{G} = \Box\Diamond\neg\mathrm{TACT}$, $\mathcal{H} = \ldots$)

```
        [⊨ (□TACT ∧ (□◇¬TACT)* ⇒ □ℱ)
                                        (Lemma IORED and STL1)

  3. Comp(□TACT ∧ (□◇¬TACT)* ⇒ □ℱ)
       ⇒ Comp(□TACT ∧ WF* ∧ SF* ⇒ □ℱ)
     PF: 2 and Lemma PSUBCOMP.0.

  4. Comp(□TACT ∧ WF* ∧ SF* ⇒ □ℱ) ⇒ (PGM ⇒ □ℱ)
     Pf: Lemma SINRED.

  5. QED
     Pf: 3 and 4.

Lemma R2. Comp(PGM ⇒ SF) ⇒ Comp(PGM ⇒ WF)
  1. (PGM ⇒ WF) [⊨ (PGM ⇒ SF)
      Pf: Lemma WFRED and Lemma PSUB.3.
  2. QED
     Pf: 1 and Lemma PSUBCOMP.0.

Lemma R3. Comp(¬(□TACT ∧ (□◇¬TACT)*)) ⇒ Comp(PGM ⇒ SF)

 1. Comp(¬(□TACT ∧ (□◇¬TACT)*))
      ⇒ Comp(□TACT ∧ (□◇¬TACT)* ⇒ ◇□TACT)
    Pf: By Lemma PSUBCOMP.0, since
         □TACT ∧ (□◇¬TACT)* ⇒ ◇□TACT
          [=] ¬(□TACT ∧ (□◇¬TACT)* ∧ □◇¬TACT)
          [⊨ ¬(□TACT ∧ (□◇¬TACT)*)

 2. Comp(□TACT ∧ (□◇¬TACT)* ⇒ ◇□TACT)
       ⇒ Comp(□TACT ∧ (□◇¬TACT)* ∧ ◇□TACT ⇒ ◇□TACT)
    Pf: Lemma IORED and STL6.

 3. Comp(□TACT ∧ (□◇¬TACT)* ∧ ◇□TACT ⇒ ◇□TACT)
      ⇒ Comp(□TACT ∧ (□◇¬TACT)* ⇒ SF)
    Pf: By Lemma PSUBCOMP.0, since
          □TACT ∧ (□◇¬TACT)* ⇒ SF
             [=] □TACT ∧ (□◇¬TACT)* ⇒ □◇¬TACT ∨ ◇□TACT
             [=] □TACT ∧ (□◇¬TACT)* ∧ ◇□TACT ⇒ ◇□TACT
 4. Comp(□TACT ∧ (□◇¬TACT)* ⇒ SF) ⇒ Comp(PGM ⇒ SF)
    Pf: By Lemma PGMRED,  since STL5 and STL2 imply SF [=] □SF.

 5. QED
    Pf: 1 - 4 and transitivity of ⇒.

Lemma R4. Comp(¬(□TACT ∧ (□◇¬TACT)*)) ⇒
    ∧ Comp(PGM ⇒ PRED)
    ∧ Comp(PGM ⇒ (PRED ⇝ PRED))

  1. Comp(¬(□TACT ∧ (□◇¬TACT)*)) ⇒ Comp(PGM ⇒ FALSE)
     Pf: By Lemma PGMRED and STL17.

  2. Comp(PGM ⇒ FALSE) ⇒ Comp(PGM ⇒ PRED)
     Pf: By Lemma PSUBCOMP.0, since
           PGM ⇒ PRED
             [=] ¬PRED ∧ PGM ⇒ FALSE
             [⊨ PGM ⇒ FALSE
  3. Comp(PGM ⇒ PRED) ⇒ Comp(PGM ⇒ (PRED ⇒ ◇PRED)
     Pf: By Lemma PSUBCOMP.0, since
           PGM ⇒ (PRED ⇒ ◇PRED)
            [=] (PGM ∧ ¬◇PRED) ⇒ ¬PRED
            [=] (PGM ∧ □PRED) ⇒ PRED
```

$[\!\!= \texttt{PGM} \Rightarrow \texttt{PRED}$ (By Lemma TACT)

4. $\texttt{Comp(PGM} \Rightarrow \texttt{(PRED} \Rightarrow \Diamond\texttt{PRED)} \Rightarrow \texttt{Comp(PGM} \Rightarrow \texttt{(PRED} \leadsto \texttt{PRED))}$
   Pf: By Lemma SINRED, Lemma PSUBCOMP.0, and Lemma PSUB.3, since
   $\Box\texttt{TACT} \wedge \texttt{WF}^* \wedge \texttt{SF}^* \;[\!\!= \texttt{PGM}$.

5. QED
   Pf: 1 - 4.

Lemma R5. $\texttt{Comp(MCPGM} \Rightarrow \Box\texttt{TACT)}$

1. $\texttt{Comp(PRED} \wedge \Box\texttt{TACT} \Rightarrow \Box\texttt{TACT)} \Rightarrow \texttt{Comp(MCPGM} \Rightarrow \Box\texttt{TACT)}$
   Pf: By definition of MCPGM.

2. $\texttt{Comp(}\Box\texttt{TACT} \Rightarrow \Box\texttt{TACT)} \Rightarrow \texttt{Comp(PRED} \wedge \Box\texttt{TACT} \Rightarrow \Box\texttt{TACT)}$
   Pf: By Lemma SIN and Lemma REDDEF.

3. $\texttt{Comp(}\Box\texttt{TACT} \Rightarrow \Box\texttt{TACT)}$
   Assume: $\texttt{F} \in \Box\texttt{TACT} \Rightarrow \Box\texttt{TACT},\ \models \texttt{F}$
   To Prove: $\vdash \texttt{F}$
   3.1. For all $\texttt{G} \in \Box\texttt{TACT}$ there exist $\texttt{P} \in \texttt{PRED}$,
   $\texttt{A} \in \texttt{ACT}$, and $\texttt{f} \in \texttt{SFCN}$ such that
   $\vdash \texttt{G} \equiv \Box\texttt{(P} \wedge \texttt{[(P = P}') \wedge \texttt{A]}_{\texttt{f,P}}\texttt{)}$.
   Pf: It follows from TLA1, STL8, and the definition of $\texttt{[A]}_{\texttt{f}}$
   that
   $\vdash \Box\texttt{(P} \wedge \Box\texttt{[B]}_{\texttt{f}}\texttt{)} \equiv$
   $\Box\texttt{(P} \wedge \texttt{[(P}'\texttt{=P)} \wedge \texttt{(B} \vee \texttt{(f}'\texttt{=f)]}_{\texttt{f,P}}\texttt{)}$
   The result then follows from the definition of
   TACT, EX(-1), and $\models \texttt{[TRUE]}_{\texttt{f}} \equiv \texttt{TRUE}$,
   which by Assumption RC implies $\vdash \texttt{[TRUE]}_{\texttt{f}} \equiv \texttt{TRUE}$.

   3.2. Choose $\texttt{P, Q} \in \texttt{PRED}$; $\texttt{A, B} \in \texttt{ACT}$, and
   $\texttt{f, g} \in \texttt{SFCN}$ such that
   $\texttt{F} = \Box\texttt{(P} \wedge \texttt{[(P}'\texttt{=P)} \wedge \texttt{A]}_{\texttt{f}}\texttt{)} \Rightarrow \Box\texttt{B}$
   Pf: By 3.1.

   3.3. $\models \texttt{(P} \wedge \texttt{[(P}'\texttt{=P)} \wedge \texttt{A]}_{\texttt{f}}\texttt{)} \Rightarrow \texttt{B}$
   Assume: $\texttt{s,t} \in \texttt{ST}$ and $\texttt{s.(P} \wedge \texttt{[(P}'\texttt{=P)} \wedge \texttt{A]}_{\texttt{f,P}}\texttt{).t}$
   To Prove: s.B.t
   3.3.1. t.P
   Pf: $\texttt{s.(P} \wedge \texttt{[(P}'\texttt{=P)} \wedge \texttt{A]}_{\texttt{f,P}}\texttt{).t}$ implies
   s.P and s.P = t.P.
   3.3.2. $\texttt{t.(P} \wedge \texttt{[(P}'\texttt{=P)} \wedge \texttt{A]}_{\texttt{f,P}}\texttt{).t}$
   Pf: By 3.3.1.
   3.3.3. Define $\sigma \in \texttt{ST}^{\omega}$ by
   $\sigma_0 = \texttt{s}$ and $\sigma_i = \texttt{t}$ for all $i > 0$.
   Then $\sigma \models \Box\texttt{(P} \wedge \texttt{[(P}'\texttt{=P)} \wedge \texttt{A]}_{\texttt{f}}\texttt{)}$
   Pf: 3.3.2, assumption, and def of $\models$.
   3.3.4. $\sigma \models \Box\texttt{B}$
   Pf: 3.3.3 and assumption $\models$ F.
   3.3.5. $\sigma_0\texttt{.B.}\sigma_1$
   Pf: 3.3.4 and definition of $\models \Box\texttt{B}$.
   3.3.6. QED
   Pf: 3.3.5 and 3.3.3.

   3.4. $\vdash \texttt{(P} \wedge \texttt{[(P}'\texttt{=P)} \wedge \texttt{A]}_{\texttt{f}}\texttt{)} \Rightarrow \texttt{B}$
   Pf: 3.3 and assumption RC.

   3.5. QED
   Pf: 3.4 and TLA2.

4. QED
   Pf: 1 - 3.

Lemma SPSIN.

1. $\models \mathtt{sin}(A, P) \land A \Rightarrow \mathtt{sin}(A, P)'$
2. $\models P \land A \Rightarrow \mathtt{sp}(A; P)'$

Lemma MAIN: If $N, \neg A_1, \ldots, \neg A_n \in \mathrm{TACT}$, then

$\models \neg(\Box N \land \Box\Diamond A_1 \land \ldots \land \Box\Diamond A_n)$

$\Rightarrow \vdash \neg(\Box N \land \Box\Diamond A_1 \land \ldots \land \Box\Diamond A_n)$

Assume: $\models \neg(\Box N \land \Box\Diamond A_1 \land \ldots \land \Box\Diamond A_n)$

To Prove: $\vdash \neg(\Box N \land \Box\Diamond A_1 \land \ldots \land \Box\Diamond A_n)$

1. Let $x \in \mathrm{PVBL}^k$ include all program variables free in N and the A_i, and let $w \in \mathrm{LVBL}^k$ not include an logical variables free in N or any of the A_i.
   Pf: The existence of x and w follow from EX7 and EX7b.

2. Define
   $P_0(w) \triangleq x = w$
   $P_i(w) \triangleq \mathtt{sin}(N, \mathtt{sp}(N \land A_i, P_{i-1})$ for $1 \le i \le n$
   Then $P_i \in \mathrm{PRED}$ for $0 \le i \le n$

   Pf: EX6 and EX1.

3. Choose $v \in \mathrm{LVBL}^k$, disjoint from w and not containing any logical variables free in N or the A_i.
   Pf: EX7b.

4. Define $w \succ v \triangleq P_n(w)[v/x]$
   Then $\models$ Well-Founded($\succ$, VAL^k).
   Pf: Define a sequence $s_0, \ldots, s_p$ to be G-LIVE iff
   $\land\ \forall i \in (0 \,..\, p] : s_{i-1}.N.s_i$
   $\land\ \forall j \in [1 \,..\, n] :$
   $\qquad \exists i \in (0 \,..\, p] : s_{i-1}.A_j.s_i$

   4.0. $w \succ v \equiv\ \models (x = v) \Rightarrow P_n(w)$

   4.0.1. For any Q : if $u \in \mathrm{LVBL}^k$, and u does not appear free in Q, then $Q \equiv \forall u : (v = u) \Rightarrow Q[u/v]$
   Pf: Ordinary logic.

   4.0.2. For any predicate Q,
   $Q[u/x] \equiv \forall s : ((s.x = u) \Rightarrow s.Q)$
   Pf: Obvious.

   4.0.3. QED
   Pf: $P_n(w)[v/x]$
   $\equiv \forall u : (u = v) \land P_n(w)[v/x][u/v]$ (by 1)
   $\equiv \forall u : (u = v) \land P_n(w)[u/x]$ (obvious)
   $\equiv \forall s : (s.x = u) \Rightarrow s.((x = v) \land P_n(w))$ (2)
   $\equiv \forall s : s.((x = v) \land P_n(w))$
   (x contains all free program variable in $P_n(w)$)
   $\equiv\ \models (x = v) \land P_n(w)$ (def of $\models$)

   4.1. For any state s and any $d \in \mathrm{VAL}^k$, if $s.P_n(d)$ then there exists a G-LIVE sequence $s_0, \ldots, s_p$ s.t. $s_p = s$ and $s_0.x = d$

   4.1.1. $\forall j \in [1 \,..\, n] : t_0.P_j(d) \Rightarrow$
   $\exists t_1, \ldots, t_q :$
   $\land\ t_q.\mathtt{sp}(N \land A_j; P_{j-1})(d)$
   $\land\ \forall i \in (0 \,..\, q] : t_i.N.t_{i-1}$
   Pf: Def of sin, since $\mathtt{sin}(\ldots)(d) = \mathtt{sin}(\ldots(d))$.

   4.1.2. $\forall j \in [1 \,..\, n] : t_0.\mathtt{sp}(N \land A; P_{j-1})(d)$

$\Rightarrow \exists\ t_1 : \land\ t_1.(N \land A_j).t_0$
$\land\ t_1.P_{j-1}(d)$

Pf: Def of sp.

4.1.3. $t_0.P_0(d) \Rightarrow \exists\ t_1, \ldots, t_q :$
$\land\ t_q.x = d$
$\land\ \forall\ i \in (0\ ..\ q] : t_i.N.t_{i-1}$

Pf: Def of sin, since sin(...)(d) = sin(...(d)).

4.1.4. QED

Pf: Use 4.1.1 - 4.1.3 to construct the sequence backwards, starting from s.

4.2. If d$\rhd$c then for any state s_0 s.t. $s_0.x = d$ there exists a G-LIVE sequence $s_0, \ldots, s_p$ s.t. $s_p.x = c$.

Pf: Assume d$\rhd$c and $s_0.x = d$.

4.2.1. If $u_0, \ldots, u_p$ is a G-LIVE sequence s.t. $u_0.x = d$ and $u_p.x = c$, and $t_i.x = u_i.x$ for all i, then $t_0, \ldots, t_p$ is a G-LIVE sequence s.t. $t_0.x = d$ and $t_p.x = c$.

Pf: Follows from hypothesis that x includes all the free variables: more precisely, that the x are chosen according to EX7.

4.2.2. Choose a G-LIVE sequence $t_0, \ldots, t_p$ s.t. $t_0.x = d$ and $t_p.x = c$

Pf: By 4.0, d$\rhd$c implies $(s.x = c) \Rightarrow s.P_n(d)$. The assumption that VAL is nonempty (it contains TRUE) implies that there exists a state s with $s.x = c$. (The nonemptiness of VAL is implied by the existence of c if $k > 0$.)

4.2.3. QED

Pf: Let $s_i = t_i$ for $i > 0$. Then $s_0, \ldots, s_p$ is a G-LIVE sequence by 4.2.1 and 4.2.2., and $s_p.x = c$. by 4.2.2.

4.3. If $c_1 > c_2 > \ldots$ then there exists $\sigma \in ST^{\omega}$ such that

(a) $\forall\ i \geq 0 : \sigma_i.N.\sigma_{i+1}$

(b) $\forall\ j \in [1\ ..\ n]$ there exist infinitely many $j \geq 0$ such that $\sigma_i.A_j.\sigma_{i+1}$

Pf: By EX6, there exists a state s_0 such that $s_0.x = c_1$. We can then apply 4.2 inductively yp 4.2, there exist G-LIVE sequences τ(i) such that the last state of τ(i) is the first state of τ(i+1). Let σ be the behavior obtained by concatenating the behaviors τ(i).

4.4. QED

Pf: Assume $c_1 > c_2 > \ldots$, and let σ be the sequence obtained in 4.3. By definition of $\models$,
$\sigma \models \Box N \land \Box\Diamond A_1 \land \ldots \land \Box\Diamond A_n$,
contradicting the hypothesis
$\models \neg(\Box N \land \Box\Diamond A_1 \land \ldots \land \Box\Diamond A_n)$.

5. QED

5.1. $\forall\ j \in [1\ ..\ n]$:
$\vdash (\Box N \land \Box\Diamond A_1 \land \ldots \land \Box\Diamond A_n) \Rightarrow (P_{j-1} \leadsto P_j)$

Pf: Assume $j \in [1\ ..\ n]$

LET R $\triangleq$ sp(N $\wedge$ A; P_{j-1})

5.1.1. $\models P_{j-1} \wedge N \Rightarrow (P_{j-1})'$

Pf: Lemma SPSIN.1

5.1.2. $\models P_{j-1} \wedge N \wedge A_j \Rightarrow R'$

Pf: Lemma SPSIN.2

5.1.3. $\vdash \Box N \wedge \Box\Diamond A_j \Rightarrow (P_{j-1} \leadsto R)$

Pf: Assumption RC and Rule PROG.

5.1.4. $R \Rightarrow P_j$

Pf: Lemma SPSIN.3

5.1.5. $\Box N \wedge \Box\Diamond A_j \Rightarrow (P_{j-1} \leadsto P_j)$

Pf: STL12 and STL13.

5.1.6. QED

Pf: 5.1.5 and PROPCALC

5.2. $\vdash (\Box N \wedge \Box\Diamond A_1 \wedge \ldots \wedge \Box\Diamond A_n) \Rightarrow ((x = w) \leadsto P_n(w))$

Pf: 5.1 and STL13, since P_0 equals $x = w$ by definition.

5.3. $\vdash P_n(w) \Rightarrow \exists\, v \in VAL^k : (w > v) \wedge x = v$

5.3.1. $\models P_n(w) \equiv \exists\, v \in VAL^k : x = v \wedge P_n(w)$

Pf: Def of $\models$.

5.3.2. $\models x = v \wedge P_n(w) \equiv (w > v)$

Pf: Definition of $>$

5.3.3. $\models P_n(w) \Rightarrow \exists\, v \in VAL^k : (w > v) \wedge x = v$

Pf: 5.3.1 and 5.3.2.

5.3.4. QED

Pf: EX9 and assumption RC.

5.4. QED

Pf: 4, 5.2, 5.3, STL18, EX8, and the LATTICE Rule, with $x = w$ substituted for P(w), and VAL^k substituted for S.

Lemma R6. Comp($\neg(\Box TACT \wedge (\Box\Diamond\neg TACT)^*)$)

Pf: Lemma MAIN and Lemma REDDEF.

PROOF OF THEOREM:

Follows from Lemmas R1 - R6, since MCPGM $\subseteq$ PGM.