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 ∈ NAT : n 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 ⊨ 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
\[ s.\text{Enabled}(A) \triangleq \exists t : s.A.t \]

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
\[ < A > f \triangleq -[-A]f \]
It equals $A \land (f' \neq f)$. 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

$$A \lor \Box (B \land \neg \neg \neg A)$$

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:

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

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 hierarchically 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:

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

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 $\Box F \land \Box G = \Box (F \land G)$ holds for any $F$ and $G$, it’s convenient to let TLA include formulas of the form $\Box (P \land \Box [A]_f)$ 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

\[ P \land \Box [A]_f \land \text{Fairness} \]

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

\[
WF_f(B) \triangleq \Box \Diamond <B>_f \lor \Box \Diamond \neg \text{Enabled}(<B>_f) \\
SF_f(B) \triangleq \Box \Diamond <B>_f \lor \Box \Diamond \neg \text{Enabled}(<B>_f)
\]

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

\[ \Box \Diamond \neg \text{TACT} \lor \Box \Diamond \text{TACT} \lor \Box \Diamond \neg \text{TACT} \lor \Box \Diamond \text{TACT}, \]

where \( \text{TACT} \) denotes any formula of the form \( Q \land [B]_g \), so \( \neg \text{TACT} \) is a formula of the form \( Q \lor <B>_g \). The Fairness formula must satisfy the additional requirement that program is machine closed, meaning that for any safety property \( S \):

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

(The theorem requires this only when \( S \) is of the form \( \Box \Diamond \neg \text{TACT} \).

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 \( \sim \) 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 \( \Pi_1 \) implements program \( \Pi_2 \), one proves \( \Pi_1 \Rightarrow \Phi_2 \), where \( \Phi_2 \) is obtained from \( \Pi_2 \) by substituting state functions for variables. This substitution preserves the form of the formula \( \Pi_2 \), 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 \( \text{sp}(S, P) \) for program statements \( S \) and predicates \( P \), where \( \text{sp} \) is the strongest postcondition operator. Assuming such predicates for arbitrary statements \( S \), which include loops or recursion, is equivalent to assuming the expressibility of \( \text{sp}(A, P) \) and \( \text{sin}(A, P) \) for atomic actions \( A \), where \( \text{sin} \) 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 ⊢ denote provability, we assume a set ACT of expressible actions such that (|= A) ⇒ (⊢ 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:

   ⊢ (□P ≡ P ∧ □[P ⇒ P']_P)

   ⊢ (A ⇒ B) ⇒ ⊢ (□A ⇒ □B)

   where P is a predicate, and A and B are actions of the form P ∧ [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

   ⊢ (F ⇒ G) ⇒ ⊢ (□F ⇒ □G)

from 1, for arbitrary formulas F and G. However, [A]_f ⇒ [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 ∈ 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 c₁ > c₂ > c₃ > ... . More precisely,

Well-Founded(>, S)

\[ \triangleq \neg \forall i : (i \in \text{NAT}) \Rightarrow \exists c_i : (c_i \in S) \land (c_i > c_{i+1}) \]

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

Well-Founded(v > w, S)

\[ \triangleq \forall c : \neg \forall i : (i \in \text{NAT}) \Rightarrow (c(i) \in S) \land (c(i) > c(i+1)) \]

where v > w is a formula with free logical variables v and w, and (c(i) > c(i+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 ∈ 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.

If \( \vdash \exists w \in S \) 
\( w \) not free in \( F \) 
\( \vdash \) Well-Founded(>, S) 
\( \vdash (F \land w \in S) \Rightarrow (P(w) \rightarrow \exists v : (v \in S) \land (w > v) \land P(v))) \)

then \( \vdash \neg F \)

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 \( \text{VAL}^k \) 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 "\( v \in \text{VAL}^k \) 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.

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

**Notations**

- \( \text{VAL}^k \triangleq \) The set of all \( k \)-tuples of elements in \( \text{VAL} \).
- \( \text{PVBL}^k \triangleq \) The set of \( k \)-tuples of DISTINCT program variables in \( \text{PVBL} \).
- \( \text{LVBL}^k \triangleq \) The set of \( k \)-tuples of DISTINCT logical variables in \( \text{LVBL} \).

Two \( k \)-tuples \( v, w \in \text{LVBL}^k \) are said to be DISJOINT iff they have no components in common.

For \( v \in \text{LVBL}^k \), \( f \) any formula, \( c \) in \( \text{LVBL}^k \) or \( \text{VAL}^k \):

- \( f[c/v] \triangleq \) 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 \( g(v)[c/v] \).

- \( \text{INTRPT} \triangleq \) The set of mappings \( \text{[LVBL} \to \text{VAL}] \).

\( \text{INTRPT} \) is the set of interpretations--substitutions of values for logical variables. For any set \( S \), an element of the set of mappings \( \text{[INTRPT} \to S] \) 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 \(O\).

\[ [[x]] : ST \rightarrow VAL^k \]
\[ [[f]] : ST \rightarrow [\text{INTRPT} \rightarrow VAL] \]
\[ [[P]] : ST \rightarrow [\text{INTRPT} \rightarrow \{\text{TRUE}, \text{FALSE}\}] \]
\[ [[A]] : ST \times ST \rightarrow [\text{INTRPT} \rightarrow \{\text{TRUE}, \text{FALSE}\}] \]

The following notation is used in place of the semantic brackets \([\cdot]\).

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

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

\[ \models A \triangleq \forall \text{int} \in \text{INTRPT} : \forall s, t \in ST : s.A.t(\text{int}) \]

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

\[ s.\text{TRUE} \triangleq \text{TRUE} \]

**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.** \(\sin(A,P) \in PRED\)
\(sp(A,P) \in PRED,\)

\[ s.sp(A,P) \triangleq \exists t : t.P \land t.A.s \]
\[ s.\sin(A,P) \triangleq \exists i \geq 0 : s.sp^i(A,P) \]
where \( \text{sp}^0(A, P) \triangleq P \)
\[ \text{sp}^{i+1}(A, P) \triangleq \text{sp}(A, \text{sp}^i(A, P)) \]

The operators \( \text{sp} \) and \( \text{sin} \) are the usual strongest postcondition and strongest invariant operators.

**EX2.** \( A \land B, \neg A, A \lor B \in \text{ACT} \)
\( P \land Q, \neg P, P \lor Q \in \text{PRED} \)

\[
\begin{align*}
\text{s.}(A \land B).t & \triangleq \text{s.A}.t \land \text{s.B}.t \\
\text{s.}(A \lor B).t & \triangleq \text{s.A}.t \lor \text{s.B}.t \\
\text{s.}(-A).t & \triangleq -(\text{s.A}.t)
\end{align*}
\]

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

**EX6.** \( x = w \in \text{PRED}, \text{and} \ \neg \models (\exists w : (x = w)) \equiv \text{FALSE} \)
\[
\text{s.}(x = w) \triangleq \text{s.x} = w
\]

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 \( A \).

\[
\exists k, x : \forall r, s, t, u \in \text{ST} : (r.x = t.x \land s.x = u.x) \Rightarrow r.A.s = t.A.u
\]

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 \( A \).

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

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

**EX8.** (a) \( \text{NAT} \in \text{SFCN} \)
(b) If \( S \in \text{SFCN} \) and \( u \in \text{LVBL} \) then \((u \in S) \in \text{PRED} \)
(c) If \( Q \in \text{PRED} \), \( x \in \text{PVBL} \), \( w \in \text{LVBL} \) then \( Q[w/x] \in \text{PRED} \)
(d) \( v > w \in \text{CONSTPRED} \), then

\[
\text{Well-Founded}(v>w, \text{VAL}^k) \in \text{CONSTPRED}.
\]

\[
\text{CONSTPRED} = \text{set of all PRED's with no free logical variables. So, } Q \in \text{CONSTPRED} \text{ iff } (\exists s : s.Q) \models \models Q
\]

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

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

where \( S \subseteq \text{VAL}^k \)
\[
(c_{i+1} > c_i) \triangleq ((w>v)[c_{i+1}/w])[c_i/v]
\]
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 $w \in \text{LVBL}^k$, then $(\exists w : P) \in \text{PRED.}$

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

EX10. $f' = f \in \text{ACT}$

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

EX11. $(f,g) \in \text{SFCN.}$

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

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 $\vdash_{\text{ACT}}$ for the logic of actions.

The relative completeness assumption is:

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

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

TEMPORAL LOGIC

Notation

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

$$\langle A \rangle_f \triangleq \neg [\neg A]_f$$

$$= A \land (f' \neq f)$$

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

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

$$\sigma_i \triangleq \text{The ith state in } \sigma, \text{ where the first state is } \sigma_0.$$

$$\sigma^{i+1} \triangleq \text{The sequence } \sigma_i, \sigma_{i+1}, \ldots \text{ in } \text{ST}^\omega$$

Temporal Logic

The wffs of a simple temporal logic consist of a set of formulas $\text{TEMPORAL}$ defined in terms of a set ELEM of elementary formulas by the following BNF grammar:
TEMPORAL $\triangleq$ ELEM | $\neg$TEMPORAL | TEMPORAL $\lor$ TEMPORAL 
| TEMPORAL $\land$ TEMPORAL | $\Box$TEMPORAL

In the following, I assume 
$F, G \in$ TEMPORAL 
$\sigma \in$ ST

The semantic domain for temporal formulas is defined by 

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

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:

$$\sigma \models \neg F \triangleq \neg(\sigma \models F)$$

$$\sigma \models F \lor G \triangleq (\sigma \models F) \lor (\sigma \models G)$$

$$\sigma \models F \land G \triangleq (\sigma \models F) \land (\sigma \models G)$$

$$\sigma \models \Box F \triangleq \forall i \geq 0 : \sigma_i \models F$$

Validity is defined by 

$$\models F \triangleq \forall \sigma \in ST^\omega : \sigma \models F$$

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

$$TACT \triangleq [\text{ACT}]\text{SFCN} \lor \text{PRED} \land [\text{ACT}]\text{SFCN}$$

$$\text{ELEM} \triangleq \text{PRED} \lor \Box TACT$$

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 $F, G \in$ RAW.

$$\Diamond F \triangleq \neg\Box \neg F$$

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

Expressibility assumption EX2 implies $\Diamond <A>_f \in$ TLA for any $A \in$ ACT and $f \in$ SFCN.
We let $\vdash_{\text{TLA}}$ denote the provability relation for TLA.
The first deduction rule is:

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

Assuming that all our complete logical system for TLA is sound, so
$\vdash_{\text{TLA}} P$ implies $\models P$, it follows from the relative completeness
assumption RC that $\vdash_{\text{TLA}} P$ implies $\vdash_{\text{ACT}} P$. Hence, STL0
implies that $\vdash_{\text{TLA}}$ and $\vdash_{\text{ACT}}$ 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

$\begin{array}{c}
F, G \\
\hline
H
\end{array}$

The remaining axioms and rules are:

**STL1.** $\vdash \Box (F \land G) \equiv \Box F \land \Box G$

**STL2.** $\vdash \Box \Box F \equiv \Box F$

**STL4.** $\vdash \Diamond \Box F \land \Diamond \Box G \equiv \Diamond (F \land G)$

**STL6.** $\vdash \Box \Diamond \Box F = \Diamond \Box F$

**STL7.** $(F \Rightarrow G) \vdash (\Box F \Rightarrow \Box G)$

**STL8.** $(F \Rightarrow G) \vdash F$

**TLA1.** For all $P \in \text{PRED}$:

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

**TLA2.** For all $A, B \in \text{TACT}$:

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

**LATTICE:**

$w \in \text{LVBL}^k$

$\exists w \in S$

$w$ not free in $F$

Well-Founded($<, S$)

$(F \land w \in S) \Rightarrow (P(w) \Rightarrow \exists v : (v \in S) \land (v < w) \land P(v))$

$\neg F$

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 $P \in \text{PRED}$, $A \in \text{TACT}$:
\begin{align*}
(\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) \\
\text{Pf:} & \vdash P \land A \Rightarrow P' \\
& \Rightarrow \vdash A \Rightarrow (P \Rightarrow P') \quad \text{(PROPCALC)} \\
& \Rightarrow \vdash A \Rightarrow [P \Rightarrow P']_P \quad \text{(PROPCALC)} \\
& \Rightarrow \vdash \Box A \Rightarrow \Box[P \Rightarrow P']_P \quad \text{(TLA2)} \\
2. (\vdash P \land A \Rightarrow P') & \Rightarrow (\vdash P \land \Box A \Rightarrow P \land \Box[P \Rightarrow P']_P) \\
\text{Pf:} & 1 \text{ and PROPCALC.} \\
3. \text{QED} \\
\text{Pf:} & 2, \text{ TLA1, and PROPCALC.} \\
\end{align*}

Lemma PRETACT: \(A, B \in \text{TACT} \Rightarrow A \land B \in \text{TACT}\) \\
\text{Pf:} ... \\

\textbf{PROG: For} \(P, Q \in \text{PRED}, A \in \text{TACT}, B \in \text{-TACT}:\) \\
\text{If} \(\vdash P \land A \Rightarrow P'\) \text{ and} \(\vdash P \land A \land B \Rightarrow Q',\) \text{then} \\
\(\vdash \Box A \land \Box \Box B \Rightarrow (P \Rightarrow Q)\) \\
\text{Assume:} \\
A. \(\vdash P \land A \Rightarrow P'\) \\
B. \(\vdash P \land A \land B \Rightarrow Q'\) \\
\text{To Prove:} \(\vdash \Box A \land \Box \Box B \Rightarrow (P \Rightarrow Q)\) \\
\text{Note: I will use PROPCALC without explicit mention.} \\
1. \(\vdash \neg Q \land [-Q \Rightarrow \neg Q']_Q \Rightarrow \neg Q'\) \\
\text{Pf:} By RC, and \\
\(\vdash \neg Q \land [-Q \Rightarrow \neg Q']_Q\) \\
\(\equiv \neg Q \land \((\neg Q \Rightarrow \neg Q') \lor (\neg Q \equiv \neg Q')\) \\
\(\equiv \neg Q \land \neg Q \equiv \neg Q') \\
\(\equiv \neg Q' \land (\neg Q \equiv \neg Q')\) \\
\(\Rightarrow \neg Q'\) \\

2. \text{QED} \\
\text{Pf:} \vdash P \land A \land B \Rightarrow Q' \quad \text{(Assumption B)} \\
\Rightarrow \vdash \neg Q' \land P \land A \Rightarrow \neg B \\
\Rightarrow \vdash \neg Q \land [-Q \Rightarrow \neg Q']_Q \land P \land A \Rightarrow \neg B \quad (1) \\
\Rightarrow \vdash \Box(\neg Q \land [-Q \Rightarrow \neg Q']_Q \land P \land A) \Rightarrow \Box \neg B \quad \text{(TLA2 and Lemma PRETACT)} \\
\Rightarrow \vdash \Box(\neg Q) \land \Box(\neg Q \Rightarrow -Q')_Q \land \Box(P \land A) \Rightarrow \Box \neg B \quad (STL1) \\
\Rightarrow \vdash \Box\Box(\neg Q) \land \Box P \land \Box A \Rightarrow \Box \neg B \quad \text{(TLA1 and STL8)} \\
\Rightarrow \vdash \Box\Box P \land \Box A \land \Diamond B \Rightarrow \Box Q \quad \text{(Def of} \Diamond) \\
\Rightarrow \vdash \Box\Box P \land \Box A \land \Box \Diamond B \Rightarrow \Box Q \quad \text{(STL8)} \\
\Rightarrow \vdash \Box P \land \Box A \land \Box \Diamond B \Rightarrow \Box Q \quad \text{(Assumption B, INV)} \\
\Rightarrow \vdash \Box(\Box A \land \Box \Diamond B) \Rightarrow (P \Rightarrow \Box Q) \\
\Rightarrow \vdash \Box(\Box A \land \Box \Diamond B) \Rightarrow (P \Rightarrow \Box Q) \quad \text{(STL1, def of} \Rightarrow) \\

\textbf{STL3.} \vdash \Box F\) \\
\text{Pf:} From STL7, with \(F \rightarrow \text{TRUE}, G \rightarrow F, \text{STL0, and PROPCALC.}\) \\

\textbf{NOTE ADDED 17 Nov 93:} I think this proof is wrong because we can’t deduce \(\vdash \Box \text{true}\) from STL0 and PROPCALC. I think the best fix is to change the conclusion of STL0 to \(\Box F.\) \\

\textbf{STL5.} \(\vdash \Box \Diamond F \lor \Box \Diamond G \equiv \Box(\Diamond F \lor \Diamond G)\) \\
\text{Pf:} \(\Box \Diamond F \lor \Box \Diamond G\) \\
\(\equiv \neg(-\Box \Diamond F \land \Box \Diamond G) \quad \text{(PROPCALC)}\) \\
\(\equiv \neg(\Box \Diamond \neg F \land \Box \Diamond \neg G) \quad \text{(Def of} \Diamond)\) \\
\(\equiv \neg(\Box \Diamond \neg(F \land \neg G) \quad \text{(STL4)}\) \\
\(\equiv \neg(-\Box \Diamond \neg(F \lor G) \quad \text{(PROPCALC)}\) \\
\(\equiv \Box(\Diamond F \lor \Diamond G) \quad \text{(Def of} \Diamond)\)
STL9. \( \vdash \Box F \land \Box \Diamond G \Rightarrow \Diamond (F \land G) \)

PF: \( \Box F \land \Box \Diamond G \)
\[ \Rightarrow \Box \Diamond F \land \Box \Diamond G \]  
\[ \Rightarrow \Box \Diamond (F \land G) \]  
\[ \Rightarrow \Diamond (F \land G) \]  
\( \text{(STL5)} \)  
\( \text{(STL8)} \)

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: \( \vdash P \land F \Rightarrow \Diamond Q \)
\[ \Rightarrow \vdash F \Rightarrow (P \Rightarrow \Diamond Q) \]  
\( \text{(PROPCALC)} \)
\[ \Rightarrow \vdash \Box F \Rightarrow (P \leadsto Q) \]  
\( \text{(STL7 and Def of \( \leadsto \))} \)

STL12. \( F \Rightarrow G \vdash F \leadsto G \)

PF: \( \vdash F \Rightarrow G \)
\[ \Rightarrow \vdash \Box (F \Rightarrow G) \]  
\( \text{(STL8)} \)
\[ \Rightarrow \vdash \Box (F \Rightarrow \Diamond G) \]  
\( \text{(STL8)} \)
\[ \equiv \vdash F \leadsto G \]  
\( \text{(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. \( \vdash \Diamond \Box \neg \text{TRUE} \equiv \neg \text{TRUE} \)
\[ \vdash \Diamond \Box \text{TRUE} \equiv \text{TRUE} \]

STL17. \( \vdash \Box \text{FALSE} \equiv \text{FALSE} \)

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

The following result asserts that these rules are all sound.

Proposition SOUND: \( \forall F \in \text{TLA}: (\vdash F) \Rightarrow (\models F) \)
Proof: Obvious (?)!

Classes of TLA Formulas

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

For any set \( F \) of formulas, \( F^* \) is defined to be the set of finite conjunctions of formulas in \( F \). More precisely,
\[
\begin{align*}
F^0 & \triangleq \{ \text{TRUE} \} \\
F^{i+1} & \triangleq F^i \land F, \text{ for } i \geq 0 \\
F^* & \triangleq \exists i \geq 0 : F^i
\end{align*}
\]
In TLA, a program is represented by a formula $F$ of the form

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

where $WF_i = \Box \Diamond <A_i>_x \lor \Box \Diamond \neg\text{Enabled}(<A_i>_x)$

$SF_i = \Box \Diamond <B_i>_x \lor \Box \Diamond \neg\text{Enabled}(<B_i>_x)$

$A_i \Rightarrow N$

$B_i \Rightarrow N$

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 $\models P \land \Box [N]_x \Rightarrow G$.

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

$$PGM \doteq PRED \land \Box TACT \land WF^* \land SF^*$$

where $WF \doteq \Box \Diamond \neg\text{TACT} \lor \Box \Diamond \neg\text{TACT}$

$SF \doteq \Box \Diamond \neg\text{TACT} \lor \Box \Diamond \text{TACT}$

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

$$\exists G \in PRED \land \Box TACT :$$
$$\land (\vdash F \Rightarrow G)$$
$$\land \forall A \in TACT : (\models F \Rightarrow \Box A) \Rightarrow (\models G \Rightarrow \Box A)$$

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 $\text{Comp}(\mathcal{F})$, defined by

$$\text{Comp}(\mathcal{F}) \doteq \forall F \in \mathcal{F} : (\models F) \Rightarrow (\vdash F)$$

The relative completeness assumption $RC$ asserts $\text{Comp}(\text{ACT})$.

Our completeness result is:

**THEOREM:**
1. $\text{Comp}(\text{MCPGM} \Rightarrow \text{PRED})$
2. $\text{Comp}(\text{MCPGM} \Rightarrow \Box TACT)$
3. $\text{Comp}(\text{MCPGM} \Rightarrow \text{(PRED }\sim\text{ PRED}))$
4. $\text{Comp}(\text{MCPGM} \Rightarrow \text{PGM})$

**THE PROOF**

The Relation $\models$
We define a "provable subset" relation \( \models \) among sets of formulas, where \( \mathcal{F} \models \mathcal{G} \) means that every formula in \( \mathcal{F} \) is provably equivalent to a formula in \( \mathcal{G} \).

\[
\mathcal{F} 
\models \mathcal{G} \quad \iff \quad \forall F \in \mathcal{F} : \exists G \in \mathcal{G} : \vdash (F \equiv G)
\]

\[
\mathcal{F} \models \mathcal{G} \quad \iff \quad (\mathcal{F} 
\models \mathcal{G}) \land (\mathcal{G} \models \mathcal{F})
\]

Lemma PSUB: For any subsets \( \mathcal{F}, \mathcal{F}_1, \) and \( \mathcal{G}_1 \) of TLA:
1. \( \mathcal{F} \models \mathcal{F} \)
2. \( (\mathcal{F}_1 \models \mathcal{F}_2) \land (\mathcal{F}_2 \models \mathcal{F}_3) \Rightarrow (\mathcal{F}_1 \models \mathcal{F}_3) \)
3. \( (\mathcal{F}_1 \models \mathcal{G}_1) \land (\mathcal{F}_2 \models \mathcal{G}_2) \)
\[ \Rightarrow \quad \mathcal{F}_1 \models \mathcal{G}_1 \land \mathcal{F}_2 \models \mathcal{G}_2 \]
4. \( \forall i \geq 0 : \mathcal{F}_1 \models \mathcal{G}_i \Rightarrow \mathcal{F}^* \models \mathcal{G}^* \)

Proof of Lemma PSUB:
1. \( \mathcal{F} \models \mathcal{F} \)
2. \( (\mathcal{F}_1 \models \mathcal{F}_2) \land (\mathcal{F}_2 \models \mathcal{F}_3) \Rightarrow (\mathcal{F}_1 \models \mathcal{F}_3) \)
3. \( (\mathcal{F}_1 \models \mathcal{G}_1) \land (\mathcal{F}_2 \models \mathcal{G}_2) \)
\[ \Rightarrow \quad \mathcal{F}_1 \models \mathcal{G}_1 \land \mathcal{F}_2 \models \mathcal{G}_2 \]
4. \( \forall i \geq 0 : \mathcal{F}_1 \models \mathcal{G}_i \Rightarrow \mathcal{F}^* \models \mathcal{G}^* \)

Pf: This follows easily from PROPCALC.

Lemma PSUBCOMP and ANDCOMP: For any subsets \( \mathcal{F} \) and \( \mathcal{G} \) of TLA:
0. \( (\mathcal{F} \models \mathcal{G}) \land \text{Comp}(\mathcal{G}) \Rightarrow \text{Comp}(\mathcal{F}) \)
1. \( \text{Comp}(\mathcal{F}) \equiv \text{Comp}(\mathcal{F}^*) \)
2. \( \{\text{TRUE}\} \models \mathcal{G} \Rightarrow (\text{Comp}(\mathcal{F} \Rightarrow \mathcal{G}) \equiv \text{Comp}(\mathcal{F} \Rightarrow \mathcal{G})) \)

Proof of 0:
\[
(\mathcal{F} \models \mathcal{G}) \land \text{Comp}(\mathcal{G})
\]
\[ \equiv \quad \forall F \in \mathcal{F} : \exists G \in \mathcal{G} : \vdash F \equiv G \]
\[ \land \forall G \in \mathcal{G} : \models G \Rightarrow \vdash G \]
\[ \Rightarrow \quad \forall F \in \mathcal{F} : \exists G \in \mathcal{G} :
\]
\[ (\vdash F \equiv G) \land (\models G \Rightarrow \vdash G) \]
\[ \Rightarrow \quad \forall F \in \mathcal{F} : \models F \Rightarrow \exists G \in \mathcal{G} :
\]
\[ (\vdash F \equiv G) \land (\models G \Rightarrow \vdash G) \]
\[ \Rightarrow \quad \forall F \in \mathcal{F} : \vdash F \Rightarrow \exists G \in \mathcal{G} :
\]
\[ \models F \Rightarrow \vdash F \quad \text{(def of } \models \text{)} \]
\[ \vdash F \land G \quad \text{(assumption)} \]
\[ \vdash F \land G \quad \text{(PROPCALC)} \]
\[ \equiv \text{Comp}(\mathcal{F}) \]

Proof of 1:
1.1. \( \text{Comp}(\mathcal{F}^*) \Rightarrow \text{Comp}(\mathcal{F}) \)
Proof: By part 0, since \( \mathcal{F} \models \mathcal{F}^* \)
1.2. \( \text{Comp}(\mathcal{F}) \Rightarrow \text{Comp}(\mathcal{F}^*) \)

1.2.1. \( \text{Comp}\{\text{TRUE}\} \)
Proof: PROPCALC implies \( \vdash \text{TRUE} \).
1.2.2. For \( i > 0 \), \( \text{Comp}(\mathcal{F}) \land \text{Comp}(\mathcal{F}^i) \Rightarrow \text{Comp}(\mathcal{F}^{i+1}) \)
Assume: \( \text{Comp}(\mathcal{F}), \text{Comp}(\mathcal{F}^i), F \in \mathcal{F}, \) and \( G \in \mathcal{F}^i \)
To Prove: \( \models F \land G \Rightarrow \vdash F \land G \)
Proof:
\[ \models F \land G \]
\[ \Rightarrow \models F \land \models G \quad \text{(def of } \models \text{)} \]
\[ \Rightarrow \vdash F \land \vdash G \quad \text{(assumption)} \]
\[ \Rightarrow \vdash F \land G \quad \text{(PROPCALC)} \]
1.2.3. QED
Proof: 1.2.1, 1.2.2, Induction, and definition of \( \mathcal{F}^* \).
1.3. QED

Proof of 2:
2.1. \( \text{Comp}(\mathcal{F} \Rightarrow \mathcal{G}) \Rightarrow \text{Comp}(\mathcal{F} \Rightarrow \mathcal{G}^*) \)
Proof: By Part 0, since \( (\mathcal{F} \Rightarrow \mathcal{G}^*) \models (\mathcal{F} \Rightarrow \mathcal{G}) \).
2.2. \( \{\text{TRUE}\} \models \mathcal{G} \land \text{Comp}(\mathcal{F} \Rightarrow \mathcal{G}) \Rightarrow \text{Comp}(\mathcal{F} \Rightarrow \mathcal{G}) \)
2.2.1. \{\text{TRUE} \models G\} \Rightarrow (F \Rightarrow G) \\
2.2.2. (\neg F \models G) \Rightarrow (\neg F \models G^i) \Rightarrow (\neg F \models G^{i+1}) \models (\neg F \models G)^{i+1} \models (\neg F \models G) \\
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. \{\text{TRUE} \models G\} \Rightarrow ((F \Rightarrow G^i) \models (F \Rightarrow G)^i) \models (F \Rightarrow G) \\
Pf: 2.2.1, 2.2.2, and mathematical induction. \\
2.2.4. \{\text{TRUE} \models G\} \land \text{Comp}((\neg F \models G)^i) \Rightarrow \text{Comp}(\neg F \models G) \\
Pf: 2.2.3 and Part 0. \\
2.2.5. QED \\
Pf: 2.2.4 and Part 1, with \((\neg F \models G)\) substituted for \(F\). \\

Lemma TACT: TACT^* \models TACT \\
Proof: Use Lemma PRETACT. \\
1. For any A and B in ACT and any f and g in SFCN: \\
(a) \models [(A \lor (f' = f)) \land (B \lor (g' = g))]_{(f,g)} \in TACT \\
(b) \models [(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 \(F\) and \(G\) of TLA: \\
\forall F \in F : \exists G \in G : \models (\vdash F \Rightarrow (\vdash G) \\
\Rightarrow (\text{Comp}(G) \Rightarrow \text{Comp}(F)) \\
Pf: Trivial. \\

Lemma COMPRED: \\
\text{Comp}(F) \land \text{Comp}(G) \Rightarrow \text{Comp}(F \land G) \\
Pf: Trivial. \\

Lemma BOXRED: For any sets \(F\) and \(G\) of formulas in TLA: \\
\text{Comp}(\Box F \Rightarrow G) \Rightarrow \text{Comp}(\Box F \Rightarrow \Box G) \\
Assume: A. \text{Comp}(\Box F \Rightarrow G) \\
B. F \in F, G \in 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 \text{PRED}, A \in \text{TACT}, G \in \text{TLA}, 
and F \in \text{WF}^* \land \text{SF}^*: 

1. \( \land \vdash (P \land \Box A \land F \Rightarrow \Box G) \Rightarrow \vdash (\sin(A, P) \land \Box A \land F \Rightarrow \Box G) \)

2. \( \vdash (\sin(A, P) \land \Box A \land F \Rightarrow \Box G) \Rightarrow \vdash (P \land \Box A \land F \Rightarrow \Box G) \)

Pf: LET I \( \triangleq \sin(A, P) \)

Assume \( P \in \text{PRED}, A \in \text{TACT}, \ldots \)

0. For all \( \tau \in \text{ST}^\omega \) and \( n \geq 0 \):
\[
(\tau \models \Box \land F) \equiv (\tau \models (\tau^+ \models H))
\]

Pf: Assume: \( \tau \in \text{ST}^\omega, n \geq 0 \),

0.1. \( \forall H \in \Box \Box \lnot \text{TACT} \lor \Box \Box \text{TACT} : 
(\tau \models H) \equiv (\tau \models (\tau^+ \models H)) 
\]

Pf: Follows easily from definitions.

0.2. \( \forall H \in \Box \Box \lnot \text{TACT} \lor \Box \Box \text{TACT} : 
(\tau \models H) \equiv (\tau \models (\tau^+ \models H)) 
\]

Pf: Follows easily from definitions.

0.3. QED

Pf: By 0.1 and 0.2, since \( \sigma \models H_1 \land H_2 \) equals \( \sigma \models H_1 \land \sigma \models H_2 \) by def of \( \models \).

1. \( \vdash P \land \Box A \Rightarrow \Box I \)

1.1. \( \vdash P \Rightarrow I \)

Pf: \( \vdash P \Rightarrow I \) and Relative Completeness Assumption 2.

1.2. \( \vdash I \land A \Rightarrow I' \)

1.2.1. ...

1.3. QED

Rule INV

2. \( \forall \sigma \in \text{ST}^\omega : 
\sigma \models \Box I \land \Box A \land F 
\Rightarrow \exists \tau, n : \land \sigma = \tau^+ 
\land \tau \models P \land \Box A \land F 
\)

Assume: 3A. \( \sigma \models \Box I \land \Box A \land F \)

To Prove: \( \exists \tau, n : \land \sigma = \tau^+ 
\land \tau \models P \land \Box A \land F \)

2.1. \( \sigma_0.1 \)

Pf: by assumption, definition of \( \models \) and \( \Box \).

2.2. Choose \( \tau_0, \ldots, \tau_n \) such that
\( \tau_0 \models P, \tau_{i-1} \models A, \tau_i, \) and \( \tau_n = \sigma_0 \).

Pf: 2.1, definition of \( \sin(P, A) \).

2.3. Let \( \tau_{n+i} \triangleq \sigma_i \), then
\( \tau \models P \land \Box A \)

Pf: By 2.2 and assumption 3A.

2.4. \( \tau \models F \)

Pf: \( \sigma \models F \) by assumption 3A, and
\( \tau \models F \) follows by 0.

2.5. QED

Pf: 2.3, 2.4.

3. \( \models (P \land \Box A \land F \Rightarrow \Box G) 
\models (\Box I \land \Box A \land F \Rightarrow \Box G) 
\)

3.1. \( \forall \sigma : 
\land \models (P \land \Box A \land F \Rightarrow \Box G) 
\land \sigma \models (\Box I \land \Box A \land F) 
\Rightarrow \sigma \models \Box G 
\)

Assume: 4A.1. \( \models (P \land \Box A \land F \Rightarrow \Box G) 
4A.2. \sigma \models (\Box I \land \Box A \land F) 

To Prove: \( \sigma \models \Box G \)
3.1.1. Choose \( \tau \) s.t.
1. \( \land \sigma = \tau + n \)
2. \( \land \tau \models P \land A \land F \)
\( \text{Pf: By 2.} \)

3.1.2. \( \tau \models G \)
\( \text{Pf: By assumption 4A.1, 3.1.1.2, and definition of} \models. \)

3.1.3. QED
\( \text{By 3.1.2, 3.1.1.1, and definition of} \models 2G. \)

3.2. QED
By 3.1, since \( \models (\box TACT \land WF \land SF \Rightarrow 2G) \Rightarrow (P \land A \land F \Rightarrow G) \)
\( \text{Pf: 1 and PROPCALC.} \)

5. QED
3 and 4.

Lemma SINRED: For any set \( F \) of formulas in TLA:
\( \text{Comp}(\box TACT \land WF \land SF \Rightarrow 2F) \Rightarrow \text{Comp}(PGM \Rightarrow 2F) \)
\( \text{Pf: Lemma SIN and Lemma REDDEF.} \)

Lemma R1.
\( \land \text{Comp}(\text{MCPGM} \Rightarrow \text{PRED}) \land \text{Comp}(\text{MCPGM} \Rightarrow \box TACT) \land \text{Comp}(\text{MCPGM} \Rightarrow WF) \land \text{Comp}(\text{MCPGM} \Rightarrow SF) \Rightarrow \text{Comp}(\text{MCPGM} \Rightarrow PGM) \)
\( \text{Pf: 1.} \land \text{Comp}(\text{MCPGM} \Rightarrow \text{PRED}) \land \text{Comp}(\text{MCPGM} \Rightarrow \box TACT) \land \text{Comp}(\text{MCPGM} \Rightarrow WF) \land \text{Comp}(\text{MCPGM} \Rightarrow SF) \Rightarrow \text{Comp}(\text{MCPGM} \Rightarrow PGM) \)
\( \text{Pf: Lemma COMPRED and definition of PGM.} \)

2. \( \land \text{Comp}(\text{MCPGM} \Rightarrow WF) \Rightarrow \text{Comp}(\text{MCPGM} \Rightarrow WF^*) \land \text{Comp}(\text{MCPGM} \Rightarrow SF) \Rightarrow \text{Comp}(\text{MCPGM} \Rightarrow SF^*) \)
\( \text{Pf: By Lemma ANDCOMP, since STL16 implies} \{\text{TRUE}\} \models WF \text{ and} \{\text{TRUE}\} \models SF. \)

3. QED
\( \text{Pf: 1 and 2.} \)

Lemma WFRED: WF \models SF
1. \( F \in WF \equiv \exists A, B \in TACT : F = \box -A \lor \box -B \)
\( \text{Pf: Def of WF.} \)

2. \( \forall A, B \in TACT : \models (\box -A \lor \box -B) \equiv (\box -(A \land B) \lor \box -\text{TRUE}) \)
\( \text{Pf: STL5, STL16, and PROPCALC.} \)

3. \( F \in WF \Rightarrow \exists G \in SF : \models F \equiv G \)
\( \text{Pf: 1, 2, EX2, EX-1, and def of SF.} \)

4. QED
\( \text{Pf: 3 and def of} \models \)

Lemma IORED: For any sets \( F, G \) and \( H \) of formulas in TLA:
\( \text{Comp}(\box F^* \land \box G \Rightarrow \box H) \Rightarrow \)
Lemma PGMRED: For any set $F_1$ and Lemma PSUBCOMP.0.

1. $\Box F^* \land (\Diamond \Box F)^* \land \Box G \Rightarrow \Box H\)  

$\triangleright (\Box F^* \land \Box G \Rightarrow \Box F \land (\Diamond \Box F)^* \land \Box H\)  

Pf: $\Box F^* \land (\Diamond \Box F)^* \land \Box G \Rightarrow \Box H\) 

[= $\Box F^* \land \Box G \Rightarrow \Box (\Diamond \Box F)^* \land \Box H\) (STL1 and STL6)  

Pf: $\Box F^* \land (\Diamond \Box F)^* \land \Box G \Rightarrow \Box H\)  

[= $\Box F^* \land \Box G \Rightarrow \Box (\Diamond \Box F)^* \land \Box H\) (STL4) 

[= $\Box (\Diamond \Box F)^* \land \Box G \Rightarrow \Box (\Diamond \Box F)^* \land \Box H\) (PROPCALC) 

2. $\Box (\Diamond \Box F)^* \land \Box G \Rightarrow \Box (\Diamond \Box F)^* \land \Box H\) 

Pf: $\Box (\Diamond \Box F)^* \land \Box G \Rightarrow \Box (\Diamond \Box F)^* \land \Box H\)  

[= $\Box (\Diamond \Box F)^* \land \Box G \Rightarrow \Box (\Diamond \Box F)^* \land \Box H\) (PROPCALC) 

[= $\Box (\Diamond \Box F)^* \land \Box G \Rightarrow \Box (\Diamond \Box F)^* \land \Box H\) (STL1) 

1 and Lemma PSUBCOMP.0.

Lemma PGMRED: For any set $F$ of formulas in TLA: 

$\Box (\Box TACT \land (\Box \Diamond \Box TACT)^* \Rightarrow \Box F) \Rightarrow \Box (PGM \Rightarrow \Box F)\)  

1. For any subsets $F$, $G$, $H$, and $I$ of TLA: 

$F \land (G \lor H)^* \Rightarrow I\) 

[= $(F \land G)^* \land H^* \Rightarrow I)\) 

1.1 $F \land (G \lor H)^0 \Rightarrow I\) 

[= $(F \land G)^* \land H^* \Rightarrow I)\) 

Pf: Trivial 

1.2 If $F \land (G \lor H)^i \Rightarrow I\) 

[= $(F \land G)^* \land H^* \Rightarrow I)\) 

then $F \land (G \lor H)^{i+1} \Rightarrow I\) 

[= $(F \land G)^* \land H^* \Rightarrow I)\) 

Pf: $F \land (G \lor H)^{i+1} \Rightarrow I\) 

[= $F \land (G \lor H)^i \land (G \lor H) \Rightarrow I\) 

[= $\land H \Rightarrow (F \land (G \lor H)^i \Rightarrow I)\) 

[= $\land G \Rightarrow (F \land (G)^* \land H^* \Rightarrow I)\) 

Pf: $\land G \Rightarrow (F \land (G)^* \land H^* \Rightarrow I)\) 

[= $(F \land (G)^* \land H^* \Rightarrow I)\) 

[= $(F \land (G)^* \land H^* \Rightarrow I)\) 

[= $(F \land (G)^* \land H^* \Rightarrow I)\) 

1.3 QED 

Pf: 1.1, 1.2, and mathematical induction.

2. $\Box TACT \land WF^* \land SF^* \Rightarrow \Box F$ 

[= $(\Box TACT \land (\Box \Diamond \Box TACT)^* \Rightarrow \Box F)\) 

Pf: $\Box TACT \land WF^* \land SF^* \Rightarrow \Box F\) 

[= $(\Box TACT \land (\Box \Diamond \Box TACT)^* \land SF^* \Rightarrow \Box F)\) 

(By 1, with $G = H = \Box \Diamond \Box TACT)\) 

[= $(\Box TACT \land (\Box \Diamond \Box TACT)^* \land (\Box TACT)^* \Rightarrow \Box F)\) 

(By 1, with $G = \Box \Diamond \Box TACT, H = \Box F)\)
\[ \begin{align*}
\text{\texttt{[= (\Box TACT \land (\Box \Diamond \neg TACT)^* \Rightarrow \Box \Diamond F)}} \]
\end{align*}\]

(Lemma IORED and STL1)

3. Comp(\Box TACT \land (\Box \Diamond \neg TACT)^* \Rightarrow \Box \Diamond F)
   \Rightarrow \text{Comp}(\Box TACT \land WF^* \land SF^* \Rightarrow \Box \Diamond F)
   
   PF: 2 and Lemma PSUBCOMP.0.

4. Comp(\Box TACT \land WF^* \land SF^* \Rightarrow \Box \Diamond F) \Rightarrow \text{PGM} \Rightarrow \Box \Diamond F
   
   Pf: Lemma SINRED.

5. QED
   
   Pf: 3 and 4.

Lemma R2. Comp(\text{PGM} \Rightarrow SF) \Rightarrow \text{Comp}(\text{PGM} \Rightarrow WF)

1. (\text{PGM} \Rightarrow WF) \Rightarrow (\text{PGM} \Rightarrow SF)
   
   Pf: Lemma WFRED and Lemma PSUB.3.

2. QED
   
   Pf: 1 and Lemma PSUBCOMP.0.

Lemma R3. Comp(\neg (\Box TACT \land (\Box \Diamond \neg TACT)^*)) \Rightarrow \text{Comp}(\text{PGM} \Rightarrow SF)

1. Comp(\neg (\Box TACT \land (\Box \Diamond \neg TACT)^*))
   \Rightarrow \text{Comp}(\Box TACT \land (\Box \Diamond \neg TACT)^* \Rightarrow \Box \Box TACT)
   
   Pf: By Lemma PSUBCOMP.0, since
   
   \[\Box TACT \land (\Box \Diamond \neg TACT)^* \Rightarrow \Box \Box TACT\]
   
   \[\begin{align*}
   \text{[=]} & \quad \neg (\Box TACT \land (\Box \Diamond \neg TACT)^* \land \Box \Diamond \neg TACT) \\
   \text{[=]} & \quad \Box TACT \land (\Box \Diamond \neg TACT)^* \Rightarrow \Box \Box TACT
   \end{align*}\]

2. Comp(\Box TACT \land (\Box \Diamond \neg TACT)^* \Rightarrow \Box \Box TACT)
   \Rightarrow \text{Comp}(\Box TACT \land (\Box \Diamond \neg TACT)^* \land \Box \Box TACT \Rightarrow \Box \Box TACT)
   
   Pf: Lemma IORED and STL6.

3. Comp(\Box TACT \land (\Box \Diamond \neg TACT)^* \land \Box \Box TACT \Rightarrow \Box \Box TACT)
   \Rightarrow \text{Comp}(\Box TACT \land (\Box \Diamond \neg TACT)^* \land \Box \Box TACT \Rightarrow SF)
   
   Pf: By Lemma PSUBCOMP.0, since
   
   \[\Box TACT \land (\Box \Diamond \neg TACT)^* \Rightarrow SF\]
   
   \[\begin{align*}
   \text{[=]} & \quad \Box TACT \land (\Box \Diamond \neg TACT)^* \Rightarrow \Box \Diamond -TACT \lor \Box \Box TACT \\
   \text{[=]} & \quad \Box TACT \land (\Box \Diamond \neg TACT)^* \land \Box \Box TACT \Rightarrow \Box \Box TACT
   \end{align*}\]

4. Comp(\Box TACT \land (\Box \Diamond \neg TACT)^* \Rightarrow SF) \Rightarrow \text{Comp}(\text{PGM} \Rightarrow SF)
   
   Pf: By Lemma PGMRED, since STL5 and STL2 imply SF \text{[=]} \Box SF.

5. QED
   
   Pf: 1 - 4 and transitivity of \Rightarrow.

Lemma R4. Comp(\neg (\Box TACT \land (\Box \Diamond \neg TACT)^*)) \Rightarrow

\begin{align*}
\land & \text{Comp}(\text{PGM} \Rightarrow \text{PRED}) \\
\land & \text{Comp}(\text{PGM} \Rightarrow (\text{PRED} \leadsto \text{PRED}))
\end{align*}

1. Comp(\neg (\Box TACT \land (\Box \Diamond \neg TACT)^*)) \Rightarrow \text{Comp}(\text{PGM} \Rightarrow False)
   
   Pf: By Lemma PGMRED and STL17.

2. Comp(\text{PGM} \Rightarrow \text{FALSE}) \Rightarrow \text{Comp}(\text{PGM} \Rightarrow \text{PRED})
   
   Pf: By Lemma PSUBCOMP.0, since
   
   \text{PGM} \Rightarrow \text{PRED}
   
   \[\text{[=]} \quad \neg \text{PRED} \land \text{PGM} \Rightarrow \text{FALSE}\]
   
   \[\text{[=]} \quad \text{PGM} \Rightarrow \text{FALSE}\]

3. Comp(\text{PGM} \Rightarrow \text{PRED}) \Rightarrow \text{Comp}(\text{PGM} \Rightarrow (\text{PRED} \Rightarrow \Diamond \text{PRED})
   
   Pf: By Lemma PSUBCOMP.0, since
   
   \text{PGM} \Rightarrow (\text{PRED} \Rightarrow \Diamond \text{PRED})
   
   \[\text{[=]} \quad \text{PGM} \land \neg \Diamond \text{PRED} \Rightarrow \neg \text{PRED}\]
   
   \[\text{[=]} \quad \text{PGM} \land \Box \text{PRED} \Rightarrow \text{PRED}\]
[= PGM ⇒ PRED (By Lemma TACT)

4. Comp(PGM ⇒ (PRED ⇒ ◯PRED)) ⇒ Comp(PGM ⇒ (PRED ⇒ ◯PRED))
Pf: By Lemma SINRED, Lemma PSUBCOMP.0, and Lemma PSUB.3, since
□TACT ∧ WF* ∧ SF* [= PGM.

5. QED
Pf: 1 - 4.

Lemma R5. Comp(MCPGM ⇒ □TACT)
1. Comp(PRED ∧ □TACT ⇒ □TACT) ⇒ Comp(MCPGM ⇒ □TACT)
Pf: By definition of MCPGM.

2. Comp(□TACT ⇒ □TACT) ⇒ Comp(PRED ∧ □TACT ⇒ □TACT)
Pf: By Lemma SIN and Lemma REDDEF.

3. Comp(□TACT ⇒ □TACT)
Assume: F ∈ □TACT ⇒ □TACT
To Prove: ⊢ F
3.1. For all G ∈ □TACT there exist P ∈ PRED,
A ∈ ACT, and f ∈ SFCN such that
⊢ G ≡ □(P ∧ [(P = P') ∧ A]f,p).
Pf: It follows from TLA1, STL8, and the definition of [A]f that
⊢ □(P ∧ □[B]f) ≡
□(P ∧ [(P' = P) ∧ (B V (f = f))]f,p)
The result then follows from the definition of TACT, EX(-1), and ⊢ [TRUE]f ≡ TRUE,
which by Assumption RC implies ⊢ [TRUE]f ≡ TRUE.

3.2. Choose P, Q ∈ PRED; A, B ∈ ACT, and
f, g ∈ SFCN such that
F = □(P ∧ [(P' = P) ∧ A]f,p) ⇒ B
Pf: By 3.1.

3.3. ⊢ (P ∧ [(P' = P) ∧ A]f) ⇒ B
Assume: s,t ∈ ST and s.(P ∧ [(P' = P) ∧ A]f,p).t
To Prove: s.B.t
3.3.1. t.P
Pf: s.(P ∧ [(P' = P) ∧ A]f,p).t implies
s.P and s.P = t.P.
3.3.2. t.(P ∧ [(P' = P) ∧ A]f,p).t
Pf: By 3.3.1.
3.3.3. Define σ ∈ STω by
σ0 = s and σi = t for all i > 0.
Then σ |= □(P ∧ [(P' = P) ∧ A]f)
Pf: 3.3.2, assumption, and def of |=.
3.3.4. σ |= □B
Pf: 3.3.3 and assumption |= F.
3.3.5. σ0.B.σ1
Pf: 3.3.4 and definition of |= □B.
3.3.6. QED
Pf: 3.3.5 and 3.3.3.
3.4. ⊢ (P ∧ [(P' = P) ∧ A]f) ⇒ B
Pf: 3.3 and assumption RC.
3.5. QED
Pf: 3.4 and TLA2.

4. QED
Pf: 1 - 3.
Lemma SPSIN.
1. \( \models \sin(A, P) \land A \Rightarrow \sin(A, P)' \)
2. \( \models P \land A \Rightarrow \sp(A; P)' \)

Lemma MAIN: If \( N, \neg A_1, \ldots, \neg A_n \in \text{TACT} \), then
\[
\models \neg(\Box N \land \Box \diamond A_1 \land \ldots \land \Box \diamond A_n) \\
\models \vdash \neg(\Box N \land \Box \diamond A_1 \land \ldots \land \Box \diamond A_n)
\]
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 \( x \in \text{PVBL}^k \) include all program variables free in \( N \) and the \( A_i \), and let \( w \in \text{LVBL}^k \) not include any 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
   \[
P_0(w) \overset{\Delta}{=} x = w \\
P_i(w) \overset{\Delta}{=} \sin(N, \sp(N \land A_i, P_{i-1})) \text{ for } 1 \leq i \leq n
   \]
   Then \( P_i \in \text{PRED} \) for \( 0 \leq i \leq n \)
   Pf: EX6 and EX1.

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

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

4.0. \( w\cdot v \equiv \models (x = v) \Rightarrow P_n(w) \)
4.0.1. For any \( Q \) if \( u \in \text{LVBL}^k \), 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: \( P_n(w)[v/x] \)
   \[
   \equiv \forall u : (u = v) \land P_n(w)[v/x][u/v] \text{ (by 1)} \\
   \equiv \forall u : (u = v) \land P_n(w)[u/x] \text{ (obvious)} \\
   \equiv \forall s : (s.x = u) \Rightarrow s.(x = v) \land P_n(w) \text{ (2)} \\
   \equiv \forall s : s.(x = v) \land P_n(w) \text{ (x contains all free program variable in } P_n(w)) \\
   \equiv \models (x = v) \land P_n(w) \text{ (def of } \models) \\
   \]
4.1. For any state \( s \) and any \( d \in \text{VAL}^k \), 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 \ldots n] : t_0.P_j(d) \Rightarrow \\
\exists t_1, \ldots, t_q : \\
\land t_q.\sp(N \land A_j; P_{j-1})(d) \\
\land \forall i \in (0 \ldots q) : t_i.N.t_{i-1}
   \]
   Pf: Def of \( \sin \), since \( \sin(...)(d) = \sin(...(d)) \).
4.1.2. \( \forall j \in [1 \ldots n] : t_0.\sp(N \land A; P_{j-1})(d) \)
⇒ ∃ t₁ : \( \land t₁.(N \land A_j).t₀ \land t₁.Pj-1(d) \)

Pf: Def of sp.

4.1.3. \( t₀.P₀(d) \Rightarrow ∃ t₁, \ldots, t_q : \land t_q.x = d \land \forall i \in (0 \ldots q] : t₁.N.ti-1 \)

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 \( d > c \) then for any state \( s₀ \) s.t. \( s₀.x = d \) there exists a G-LIVE sequence \( s₀, \ldots, s_p \) s.t. \( s_p.x = c. \)

Pf: Assume \( d > c \) and \( s₀.x = d. \)

4.2.1. If \( u₀, \ldots, u_p \) is a G-LIVE sequence s.t.

\( u₀.x = d \) and \( u_p.x = c \), and \( t_i.x = u_i.x \) for all \( i \), then \( t₀, \ldots, t_p \) is a G-LIVE sequence s.t.

\( t₀.x = d \) 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 \( t₀, \ldots, t_p \) s.t.

\( t₀.x = d \) and \( t_p.x = c \)

Pf: By 4.0, \( d > c \) implies \( (s.x = c) \Rightarrow s.P_n(d). \)

The assumption that \( VAL \) is nonempty (it contains \( \text{TRUE} \)) 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 \( s₀, \ldots, s_p \) 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 \( c₁ > c₂ > \ldots \) then there exists \( σ \in ST^ω \) such that

(a) \( \forall i \geq 0 : σ_i.N.σ_{i+1} \)

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

Pf: By \( EX6, \) there exists a state \( s₀ \) such that

\( s₀.x = c₁. \) We can then apply 4.2 inductively

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 \( c₁ > c₂ > \ldots \), and let \( σ \) be the sequence obtained in 4.3. By definition of \( \models \),

\( σ \models □N \land □◇A₁ \land \ldots \land □◇Aₙ, \)

contradicting the hypothesis

\( \models ¬(□N \land □◇A₁ \land \ldots \land □◇Aₙ). \)

5. QED

5.1. \( \forall j \in [1 \ldots n] : \)

\( \vdash (□N \land □◇A₁ \land \ldots \land □◇Aₙ) \Rightarrow (Pj-1 \leadsto P_j) \)

Pf: Assume \( j \in [1 \ldots n] \)
LET \( R \triangleq \text{sp}(N \land A; P_{j-1}) \)

5.1.1. \( \models P_{j-1} \land N \Rightarrow (P_{j-1})' \)
Pf: Lemma SPSIN.1

5.1.2. \( \models P_{j-1} \land N \land A_j \Rightarrow R' \)
Pf: Lemma SPSIN.2

5.1.3. \( \vdash \Box N \land \Box A_j \Rightarrow (P_{j-1} \rightarrow R) \)
Pf: Assumption RC and Rule PROG.

5.1.4. \( R \Rightarrow P_j \)
Pf: Lemma SPSIN.3

5.1.5. \( \Box N \land \Box A_j \Rightarrow (P_{j-1} \rightarrow P_j) \)
Pf: STL12 and STL13.

5.1.6. QED
Pf: 5.1.5 and PROPCALC

5.2. \( \vdash (\Box N \land \Box A_1 \land \ldots \land \Box A_n) \Rightarrow ((x = w) \rightarrow P_n(w)) \)
Pf: 5.1 and STL13, since \( P_0 \) equals \( x = w \) by definition.

5.3. \( \vdash P_n(w) \Rightarrow \exists v \in \text{VAL}^k : (w > v) \land x = v \)

5.3.1. \( \models P_n(w) \equiv \exists v \in \text{VAL}^k : x = v \land P_n(w) \)
Pf: Def of \( \models \).

5.3.2. \( \models x = v \land P_n(w) \equiv (w > v) \)
Pf: Definition of >

5.3.3. \( \models P_n(w) \Rightarrow \exists v \in \text{VAL}^k : (w > v) \land x = v \)
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 \( \text{VAL}^k \) substituted for \( S \).

Lemma R6. Comp\(-((\Box \text{TACT} \land (\Box \Diamond \neg \text{TACT})^*))\)
Pf: Lemma MAIN and Lemma REDDEF.

PROOF OF THEOREM:
Follows from Lemmas R1 - R6, since MCPGM \( \subseteq \text{PGM} \).