Verification of Concurrent Programs: A Temporal Proof System

by

Zohar Manna and Amir Pnueli

Department of Computer Science

Stanford University
Stanford, CA 94305
A proof system based on temporal logic is presented for proving properties of concurrent programs based on the shared-variables computation model. The system consists of three parts: the general uninterpreted part, the domain dependent part and the program dependent part. In the general part we give a complete proof system for first-order temporal logic with detailed proofs of useful theorems. This logic enables reasoning about general time sequences. The domain dependent part characterizes the special properties of the domain over which the program operates. The program dependent part introduces program axioms which restrict the time sequences considered to be execution sequences of a given program.

The utility of the full system is demonstrated by proving invariance, liveness and precedence properties of several concurrent programs. Derived proof principles for these classes of properties, are obtained and lead to a compact representation of proofs.

This paper appears in the Proceedings of the 4-th School of Advanced Programming, Amsterdam, Holland (June 1982).

This research was supported in part by the National Science Foundation under grants MCS79-09495 and MCS80-06930, by DARPA under Contract N00039-82-C-0250, by the United States Air Force Office of Scientific Research under Grant AFOSR-81-0014, and by the Basic Research Foundation of the Israeli Academy of Sciences.
A. INTRODUCTION

In this work we present a proof system based on temporal logic for proving the properties of concurrent programs. We refer the reader to \[MP1\] for a more detailed discussion of the computational model of concurrent programs, and the advantages offered by the language of temporal logic in formulating properties of concurrent programs.

1. THE TEMPORAL LANGUAGE: SYNTAX AND SEMANTICS

We first describe the temporal language we are going to use. This language contains special constructs that are suitable for reasoning about programs.

The language uses a set of basic symbols consisting of individual variables and constants, propositions, and function and predicate symbols. The set is partitioned into two subsets: global and local symbols. Intuitively speaking, the global symbols denote entities that do not change during a program execution. The local symbols, on the other hand, may change their meanings and values in different states throughout the execution. For our purpose, the only local symbols that interest us are local individual variables and propositions. We will have global symbols of all types.

We use the usual set of boolean connectives: $\land$, $\lor$, $\Rightarrow$, $\equiv$, and $\sim$ together with the equality predicate $=$ and the first-order quantifiers $\forall$ and $\exists$. These operators are referred to as the classical operators. The quantifiers $\forall$ and $\exists$ are applied only to global individual variables.

The modal operators used are: $\Box$, $\Diamond$, $\#$, and $\mathcal{U}$, which are called respectively the always, sometime, next and until operators. The first three operators are unary while the $\mathcal{U}$ operator is binary. We use the next operator, $\Diamond$, in two different ways -- as a temporal operator applied to formulas and as a temporal operator applied to terms.

A model $(I, \alpha, \sigma)$ for our language consists of a (global) interpretation $I$, a (global) assignment $\alpha$ and a sequence of states $\sigma$.

- The interpretation $I$ specifies a nonempty domain $D$ and assigns concrete elements, functions and predicates to the (global) individual constants, function and predicate symbols.

- The assignment $\alpha$ assigns a value over the appropriate domain to each of the global individual variables.

- The sequence $\sigma = s_0, s_1, \ldots$ is an infinite sequence of states. Each state $s_i$ assigns values to the local individual variables and propositions.
For a sequence

\[ \sigma = s_0, s_1, \ldots \]

we denote by

\[ \sigma^{(i)} = s_i, s_{i+1}, \ldots \]

the \( i \)-truncated suffix of \( \sigma \).

Given a temporal formula \( w \), we present below an inductive definition of the truth value of \( w \) in a model \( (I, \alpha, a) \). The value of a subformula or term \( \tau \) under \( (I, \alpha, a) \) is denoted by \( \tau|_\sigma \), with \( I \) being implicitly understood.

Consider first the evaluation of terms:

- For a local individual variable or local proposition \( y \):
  \[ y|_\sigma = s_0[y], \]
  i.e., the value assigned to \( y \) in \( s_0 \), the first state of \( \sigma \).

- For a global individual variable \( u \):
  \[ u|_\sigma = \alpha[u], \]
  i.e., the value assigned to \( u \) by \( \alpha \).

- For an individual constant the evaluation is given by \( I \):
  \[ c|_\sigma = I[c]. \]

- For a \( k \)-ary function \( f \):
  \[ f(t_1, \ldots, t_k)|_\sigma = I[f](t_1|_\sigma, \ldots, t_k|_\sigma), \]
  i.e., the value is given by the application of the interpreted function \( I[f] \) to the values of \( t_1, \ldots, t_k \) evaluated in the model \( (I, \alpha, \sigma) \).

- For a term \( t \):
  \[ (\circ t)|_\sigma = t|_{\sigma^{(1)}}, \]
  i.e., the value of \( t \) in \( \sigma = s_0, s_1, \ldots \) is given by the value of \( t \) in the 1-truncated suffix \( \sigma^{(1)} = s_1, s_2, \ldots \).

Consider now the evaluation of formulas:

- For a \( k \)-ary predicate \( p \) (including equality):
  \[ p(t_1, \ldots, t_k)|_\sigma = I[p](t_1|_\sigma, \ldots, t_k|_\sigma), \]
  Mere again, we first evaluate the arguments in the model and then test \( I[p] \) on them.

- For a disjunction:
  \[ (w_1 \lor w_2)|_\sigma = true \quad if \ and \ only \ if \ w_1|_\sigma = true \ or \ w_2|_\sigma = true. \]
  And similarly for the other binary boolean connectives \( \land, \lor, \land \).
• For a negation:
  \((\neg w)^{\alpha}_{\sigma} = true \text{ if and only if } w^{\alpha}_{\sigma} = false.\)

• For a next-time application:
  \((\diamondsuit w)^{\alpha}_{\sigma} = w^{\alpha}_{\sigma(1)}.\)
  Thus 0 \(w\) means: \(w\) will be true in the next instant — read “next \(w\”.

• For an all-times application:
  \((\square w)^{\alpha}_{\sigma} = true \text{ if and only if for every } k \geq 0, w^{\alpha}_{\sigma(k)} = true,\)
  i.e., \(w\) is true for all suffix sequences of \(\sigma\). Thus \(\square w\) means: \(w\) is true for all future instants (including the present) -- read “always \(w\” or “henceforth \(w\”.

• For a sometime application:
  \((\lozenge w)^{\alpha}_{\sigma} = true \text{ if and only if there exists a } k \geq 0\)
  such that \(w^{\alpha}_{\sigma(k)} = true,\)
  i.e., \(w\) is true on at least one suffix of \(\sigma\). Thus \(0 w\ means: \(w\) will be true for some future instant (possibly the present) -- read “sometime \(w\” or “eventually \(w\”.

• For an until application:
  \(w_1 \sqcup w_2^{\alpha}_{\sigma} = true \text{ if and only if for some } k \geq 0, w_2^{\alpha}_{\sigma(k)} = true \text{ and for all } i, 0 \leq i < k, w_1^{\alpha}_{\sigma(i)} = true.\)
  Thus \(w_1 \sqcup w_2\) means: there is a future instant in which \(w_2\) holds, and such that \(until\) that instant \(w_1\) continuously holds -- read “\(w_1\ until \(w_2\” ([KAM], [GPSS]).

• For a universal quantification:
  \((\forall u. w)^{\alpha}_{\sigma} = true \text{ if and only if for every } d \in D, w^{\alpha'}_{\sigma'} = true,\)
  where \(\alpha' = \alpha \circ [u \leftarrow d]\) is the assignment, obtained from \(\alpha\) by assigning \(d\) to \(u\).

• For an existential quantification:
  \((\exists u. w)^{\alpha}_{\sigma} = true \text{ if and only if for some } d \in D, w^{\alpha'}_{\sigma'} = true,\)
  where \(\alpha' = \alpha \circ [u \leftarrow d]\).

Following are some examples of temporal expressions and their intuitive interpretations:

\(u \lozenge \lozenge v\)
If \(u\) is presently true, \(v\) will eventually become true.

\(\square \lozenge\ \lozenge\ \lozenge\)
Whenever \(u\) becomes true it will eventually be followed by \(v\).

\(\square \square \cdot\)
At some future instant \(w\) will become permanently true.

\(\lozenge (w \land 0 \neg w)\)
There will be a future instant such that \(w\) is true at that instant and false at the next.

\(\square \lozenge w\)
Every future instant is followed by a later one in which \(w\) is true,
thus \( w \) is true infinitely often.

\[ \square u \supset \square \square u \]  If \( u \) ever becomes true, then \( v \) is true at that instant and ever after.

\[ \square u \lor (u \lor v) \]  Either \( u \) holds continuously or it holds until an occurrence of \( v \).

This is the weak form of the until operator that states that \( u \) will hold continuously until the first occurrence of \( v \) if \( v \) ever happens or indefinitely otherwise.

\[ \ominus v 3 (\lnot v) \lor u \]  If \( v \) ever happens, its first occurrence is preceded by (or coincides with) \( u \).

If \( w \) is true under the model \( (I, \alpha, a) \), we say that \( (I, \alpha, a) \) satisfies \( w \) or that \( (C, a, a) \) is a (satisfying) model for \( w \). We denote this by

\[ (I, \alpha, a) \models w. \]

A formula \( w \) is satisfiable if there exists a satisfying model for it.

A formula \( w \) is valid if it is true in every model; in this case we write

\[ \models w. \]

Sometimes we are interested in a restricted class of models \( C \). A formula \( w \) which is true for every model in \( C \) is said to be \( C \)-valid, denoted by

\[ \models_C w. \]

Example:

The formula \( \Diamond (w_1 \land w_2) \lor (0 w_1 \land 0 w_2) \) is valid, i.e.,

\[ \models \Diamond (w_1 \land w_2) \lor (0 w_1 \land 0 w_2). \]

It says that if there exists an instant in which both \( w_1 \) and \( w_2 \) are true then there exists an instant in which \( w_1 \) is true and there exists an instant in which \( w_2 \) is true.

Reversing the implication does not yield a valid formula, i.e.,

\[ \not\models (\Diamond w_1 \land 0 w_2) \lor (w_1 \land w_2). \]

For, consider an interpretation consisting of a sequence of states:

\[ \sigma : s_0, s_1, \ldots \]
such that \( w_1 \) is true on all odd numbered states and false elsewhere, and \( w_2 \) is true on all the even numbered states and false on the odd ones. Then certainly both \( 0 \ w_1 \) and \( 0 \ w_2 \) are true on \( \sigma \), hence \( 0 \ w_1 \ A \ 0 \ w_2 \) is true. On the other hand, there is no state on which both \( w_1 \) and \( w_2 \) are true simultaneously. Hence \( \Diamond (w_1 \ A \ w_2) \) is false. Consequently the implication is false under the interpretation \( \sigma \).

2. THE PROOF SYSTEM

Having defined valid formulas, we naturally look for a deductive system in which validity can be proved. In such a system we take some of the valid formulas as axioms and provide a set of sound inference rules by which we hope to be able to prove the other valid formulas as theorems. A formula \( w \) is a theorem of the system either if it is an axiom of the system or has a proof in which it is derived from the axioms using the inference rules of the system. We denote the fact that, \( w \) is a theorem is provable within the system by \( \vdash w \).

Our interest in the temporal logic formalism is mainly motivated by the applicability of this logic to proving properties of concurrent programs. Therefore, apart from developing the general basic logical properties of the operators and their interrelations, we will mostly be interested in properties that are valid over computations of a given concurrent program \( P \). Thus, the notion of validity our system will try to capture is that of a formula being true for all possible computations of the given program, and not necessarily over an arbitrary model. This corresponds to the concept of \( A(P) \)-validity where \( A(P) \) is the class of all models corresponding to computations of \( P \).

We structure our proof system into three main layers dependent on the universal validity of the theorems that can be derived in each layer. In the first layer, called the general part, we deal with the general temporal properties of discrete linear sequences (arbitrary models). Theorems proved in that part are valid for all sequences over arbitrary domains. They universally hold for arbitrary computations of all programs over such domains, as well as for sequences which cannot even be derived as the computations of a program. In the next layer the domain part, we restrict our attention to a particular domain \( D \) and provide tools for proving validity over models all of which are interpreted over \( D \). The third, most restrictive layer is the program part. Here we restrict our attention to a particular program \( P \) and develop tools for proving validity only over models whose sequences are legal computations of \( P \).

In a forthcoming paper, the program dependent part is proved to be complete relative to the general temporal theory over the data domain. We also show that its dependence on the particular computation model studied is modular, by presenting a similar system for proving properties of CSP programs.
We start the general part by describing first the axiomatic system for propositional temporal logic in which we do not admit predicates or quantification.

3. THE PROPOSITIONAL TEMPORAL SYSTEM ($\Box, \Diamond, \lozenge$ AND $\mathbf{u}$)

The proof system for the propositional part, consists of the following axioms:

**AXIOMS:**

\begin{align*}
\text{A1.} & \quad \vdash \sim \Diamond w \equiv \Box \sim w \\
\text{A2.} & \quad I \cdot \Box (w_1 \supset w_2) \supset (\Box w_1 \supset \Box w_2) \\
\text{A3.} & \quad \vdash \Box w \supset w \\
\text{A4.} & \quad \Box 0 \equiv \sim 0 \mathbf{u} \\
\text{A5.} & \quad \vdash \Box (w_1 \supset w_2) \supset (\Box w_1 \supset \Box w_2) \\
\text{A6.} & \quad \vdash \Box \mathbf{u} w \supset \Box \mathbf{u} w \\
\text{A7.} & \quad \vdash \Box w \supset \Box \Box w \\
\text{A8.} & \quad (\Box \Diamond w_1 \supset \Box \Diamond w_2) \supset (\Box \Diamond w_1 \supset \Box \Diamond w_2) \\
\text{A9.} & \quad \vdash (w_1 \mathbf{u} w_2) \equiv [w_2 \mathbf{v} (w_1 \mathbf{u} (w_1 \mathbf{u} w_2))] \\
\text{A10.} & \quad \vdash (w_1 \mathbf{u} w_2) \supset \Diamond w_2.
\end{align*}

Axiom A1 defines 0 as the dual of Cl; it states that at all times $w$ is false if and only if it is not the case that sometimes $w$ holds. Axiom A2 states that if universally $w_1$ implies $w_2$ then if at all times $w_1$ is true then so is $w_2$. Axiom A3 establishes the present as part of the future by stating that if $w$ is true at all future instants it must be true at the present. Axiom A4 establishes 0 as self-dual. Consequently it implies that the next instant exists and is unique, and restricts our models to linear sequences (no branching). Axiom A5 is the analogue of A2 for the 0 operator. Axiom A6 states that the next instant is one of the future states. Axiom A7 states that if $w$ holds in all future instants it also holds in all instants which lie in the future of the next instant. Axiom A8 is the "computational induction" axiom; it states that if a property is inherited over one step transitions, it is invariant, over any suffix sequence whose first state satisfies $w$. Axiom A9 characterizes the until operator by distributing its effect into what is implied for the present and what is implied for the next instant. Axiom A10 simply states that "$w_1$ until $w_2$" implies that $w_2$ will eventually happen.
**INFERENCE RULES:**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1.</td>
<td>Propositional Tautology</td>
<td>If ( u ) is an instance of a propositional tautology then ( \vdash u )</td>
</tr>
<tr>
<td>R2.</td>
<td>Modus Ponens</td>
<td>If ( \vdash u \supset v ) and ( \vdash u ) then ( \vdash v )</td>
</tr>
<tr>
<td>R3.</td>
<td>Insertion</td>
<td>If ( \vdash u ) then ( \vdash \Box u )</td>
</tr>
</tbody>
</table>

All these rules are sound. The soundness of R1 and R2 is obvious. Note that in R1 we also include temporal instances of tautologies; we may substitute an arbitrary temporal formula for a proposition letter in obtaining an instance. For example, the formula \( \Box w \supset \Box w \) is a temporal instance of the tautology \( p \supset p \). To justify R3, we recall that validity of \( w \) means that \( w \) is true in all models, hence \( \Box w \) is also valid.

**DERIVED RULES AND THEOREMS:**

Before giving some theorems that can be proved in this system, we develop several useful derived rules:

**Propositional Reasoning** -- PR

\[
\begin{align*}
\vdash & (u_1 \land u_2 \land \ldots \land u_n) \supset v \\
\vdash & u_1, \vdash u_2, \ldots, \text{ and } \vdash u_n \\
\hline
\vdash & \psi
\end{align*}
\]

The notation above is used to describe inference rules. IL has the general form

\[
\vdash \varphi_1, \vdash \varphi_2, \ldots, \vdash \varphi_m \\
\hline
\vdash \psi
\]

and means that if we have already proved \( \varphi_1, \ldots, \varphi_m \) (the assumptions or premises of the rule), we are allowed by this rule to infer \( \psi \) (the conclusion or consequent of the rule).

**Proof:**

The rule PR follows from the propositional tautology (Rule R1)

\[
1- [\,(u_1 \land u_2 \land \ldots \land u_n) \supset v) \supset [u_1 \supset (u_2 \supset \ldots (u_n \supset v) \ldots\,)]
\]

by applying MI' (Rule R2) \( n + 1 \) times. \( \square \)
Whenever we apply this derived rule without explicitly indicating the premise

$$\vdash (u_1 \land u_2 \land \ldots \land u_n) \supset v,$$

it means that the premise is an instance of a propositional tautology.

\[ \square \text{Insertion} \rightarrow 01 \]

\[
\begin{array}{c}
\vdash u \\
\hline \\
\vdash o \ u
\end{array}
\]

Proof:

1. \( \vdash u \)  
   given
2. \( \vdash \square u \)  
   by 01
3. \( \vdash ou \)  
   by A6 and MP

The first theorem that we derive in the system is:

\[ T_1. \ \vdash w \supset \diamond w \]

Proof:

1. \( \vdash (\square \neg w) \supset w \)  
   by A3
2. \( \vdash w \supset (\neg \square \neg w) \)  
   by PR
3. \( \vdash w \supset ow \)  
   by A6 and PR

The theorem implies (by MP) the derived rule

\[ \diamond \text{Insertion} \rightarrow 01 \]

\[
\begin{array}{c}
\vdash u \\
\hline \\
\vdash o \ u
\end{array}
\]

\[ T_2. \ \vdash \square w \supset o \ w \]

Proof:

1. \( \vdash (\square \neg w) \supset (o \ w) \)  
   by A6

9
The following three rules (and a similar rule for the until operator presented later) show that all the temporal operators are monotonic in the sense that an argument may be replaced by a weaker statement yielding a weaker expression.

\[
\begin{array}{ccc}
\text{□□ Rules} & \text{t-u ⊆ v} & \text{t-u ≡ v} \\
\hline
\hline
(a) \quad & \quad & \\
\text{t □ u 3 □ v} & \quad & \quad \\
(b) \quad & \quad & \\
\text{t □ u ≡ □ v} & \quad & \quad \\
\end{array}
\]

Proof of (a):

1. t-u ⊆ v 
2. t□u □ v by □I
3. (t□u □ v) ⊆ (□u □ v) by A2
4. t □ u ⊆ □ v by 2, 3 and MP

Rule (b) then follows by propositional reasoning by using the tautology

\[
[(\text{t □ v}) \land (\text{v □ u})] \equiv (u \equiv v).
\]

\[
\begin{array}{ccc}
\text{□□□ Rules} & \text{t-u ⊆ v} & \text{t-u ≡ v} \\
\hline
\hline
(a) \quad & \quad & \\
\text{t □ u □ v} & \quad & \quad \\
(b) \quad & \quad & \\
\text{t □ u ≡ □ v} & \quad & \quad \\
\end{array}
\]

Proof of (a):

1. t-u ⊆ v 
2. t □ v 3 □ u by PR
3. t □ v ⊆ □ v by □ I□I
4. t □ v ⊆ t □ u by Al and PR
5. t □ u ⊆ □ v by PR

Rule (b) then follows by propositional reasoning.
0 0 Rules

\[
\frac{\Gamma \vdash u \supset v}{\vdash \circ \circ u \supset \circ \circ v} \quad (a)
\]

\[
\frac{\Gamma \vdash u \equiv v}{\vdash \circ \circ u \equiv \circ \circ v} \quad (b)
\]

Proof of (a):

1. \( \vdash u \supset v \)
2. \( \vdash \circ (u \supset v) \) \quad by 01
3. \( \vdash \circ u \supset \circ v \) \quad by A5 and MP

Rule (b) follows by propositional reasoning.

Computational Induction Rule \( - \circ CI \)

\[
\frac{\vdash u \supset o u}{\vdash \circ \circ u \supset o \circ u}
\]

Proof:

1. \( \vdash u \supset o u \) \quad given
2. \( o \circ (u \supset o u) \) \quad by \( \square l \)
3. \( \vdash \circ (u \supset o u) \supset (u \supset \circ u) \) \quad by \( \Lambda 8 \)
4. \( \vdash u \supset \circ u \) \quad by 2, 3 and MP

Derived Computational Induction Rule \( - \circ DCI \)

\[
\frac{\vdash u \supset (v \land \circ u)}{\vdash u \supset \circ \circ v}
\]

Proof:

1. \( \vdash u \supset (v \land \circ u) \) \quad given
2. \( \vdash u \supset \circ u \) \quad by PR
3. \( \vdash u \supset \circ u \) \quad by CI
4. \( \vdash u \supset v \) \quad by 1 and PR
5. \( \vdash u \supset \circ v \) \quad by \( \square \square \)
6. \( \vdash u \supset \Box v \) by 3, 5 and PR

The following two theorems show that the Cl and 0 operators are both idempotent:

**T3.** \( \Diamond \Box \Diamond \equiv \Box \Box w \)

**Proof:**

1. \( \vdash \Box \Box w \supset \Box w \) by A3
2. \( \vdash \Box w \supset \Diamond \Box w \) by A7
3. \( \vdash \Box w \supset \Box \Box w \) by CI
4. \( \vdash \Box w \equiv \Box \Diamond w \) by 1, 3 and PR

**T4.** \( \mathbf{I} \cdot o \cdot w \equiv o \cdot o \cdot w \)

**Proof:**

1. \( \vdash \sim \Diamond w \equiv \Box \sim w \) by Al
2. \( \vdash \Box \sim w \equiv 0 \cdot 0 - w \) by T3
3. \( \vdash \Box \sim w \equiv \cdots \) by 1 and EI
4. \( \vdash \mathbf{I} \cdot \mathbf{E} \cdot \mathbf{I} \cdot \mathbf{O} \cdot w \equiv \sim \Diamond \Diamond w \) by Al
5. \( \vdash \sim \Diamond w \equiv \sim \Diamond \Diamond w \) by 1, 2, 3, 4 and PR
6. \( \vdash \Diamond w \equiv \Diamond \Diamond w \) by PR

Because of these last two theorems we can collapse any string of consecutive identical modalities such as \( \Box \ldots \Box \) for \( 0 \ldots 0 \) into a single modality of the same type.

The following theorem establishes that \( \Box \) is the dual of 0. Note that A1 states that 0 is the dual of CI, i.e., \( 0 \equiv \sim \Box \sim w \).

**T5.** \( \vdash (O \cdot w) \equiv (\sim \Box w) \)

**Proof:**

1. \( \vdash (\sim \sim w) \equiv w \) by PT
2. \[ \vdash (\Box \sim \sim w) \equiv \Box \ \ast \]  
   by \( \Box \) Cl 

3. \[ \vdash (\sim \Diamond \sim w) \equiv \Box \ \ast \]  
   by Al and PR 

4. \[ \vdash (\Diamond \sim w) \equiv (\sim \Diamond \Diamond) \]  
   by PR 

T8. \( \Box \Diamond \Box \ w_2 \supset \Box \Box \ w_1 \supset \Diamond \ w_2 \) 

Proof: 

1. \[ \vdash (w_1 \supset w_2) \equiv (\sim w_2 \supset \sim w_1) \]  
   by PT 

2. \[ \vdash (\Box w_1 \supset \Box w_2) \equiv (\Box (\sim w_2 \supset \sim w_1)) \]  
   by \( \Box \) IE 

3. \[ \vdash \sim w_2 \supset \sim w_1 \supset (\Box w_2 \supset \Box \sim w_1) \]  
   by A2 

4. \[ \vdash (\Box \sim w_2 \supset \Box \sim w_1) \equiv (\sim \Diamond w_2 \supset \sim w_1) \]  
   by Al and PR 

5. \[ \vdash (\sim \Diamond w_2 \supset \sim w_1) \equiv (\Diamond w_1 \supset \Diamond w_2) \]  
   by PT 

6. \[ \vdash (\Diamond w_2 \supset \Diamond w_1) \supset (\Diamond w_1 \supset \Diamond w_2) \]  
   by 2, 3, 4, 5 anti PR 

The following theorems show the interaction between the temporal and the boolean operators.

T7. \( \Box \Diamond (w_1 \wedge w_2) \equiv (\Box w_1 \wedge \Box w_2) \) 

Proof: 

1. \[ \vdash (w_1 \wedge w_2) \supset w_1 \]  
   by PT 

2. \[ \vdash (\Box \Diamond (w_2 \supset \Box w_1) \wedge w_2) \]  
   by \( \Box \) IC 

3. \[ \vdash (w_1 \wedge w_2) \supset w_2 \]  
   by PT 

4. \[ \vdash (\Diamond w_1 \wedge \Diamond w_2) \supset w_2 \]  
   by \( \Diamond \) IE 

5. \[ \vdash \Box (w_1 \wedge w_2) \supset (\Box w_1 \wedge \Box w_2) \]  
   by 2, 4 and PR 

6. \[ \vdash w_1 \supset (w_2 \supset \Box w_1 \wedge \Box w_2) \]  
   by PT 

7. \[ \vdash (w_2 \supset (w_1 \wedge w_2)) \supset (w_1 \wedge w_2) \]  
   by \( \Box \Box \) 

8. \[ \vdash (w_2 \supset (w_1 \wedge w_2)) \supset (w_1 \wedge w_2) \]  
   by A2 

9. \[ \vdash \Box w_1 \supset (\Box w_2 \supset \Box (w_1 \wedge w_2)) \]  
   by 7, 8 and PR 

10. \[ \vdash (\Box w_1 \wedge \Box w_2) \supset \Box (w_1 \wedge w_2) \]  
    by PR
11. \( \vdash [w_1 A w_2] \equiv [\Box w_1 A \Diamond w_2] \) by 5, 10 and PR

T8. \( \vdash \Diamond (w_1 \lor w_2) \equiv (\Diamond w_1 \lor \Diamond w_2) \)

Proof:

1. \( \vdash [w_1 \lor w_2] \equiv [\Box w_1 \lor w_2] \) by PT and Cl
2. \( \vdash [w_1 A \lor w_2] \equiv [\Box w_1 A \lor \Box w_2] \) by T7
3. \( \vdash [\Box w_1 A \lor \Box w_2] \equiv [\Box w_1 \lor \Box w_2] \) by PR
4. \( \vdash [\Box w_1 \lor \Box w_2] \equiv [\Box w_1 \lor \Box w_2] \) by 1, 2, 3 and PR
5. \( \vdash [w_1 \lor w_2] \equiv [\Diamond w_1 \lor \Diamond w_2] \) by A1 and PR
6. \( \vdash [w_1 \lor w_2] \equiv [\Diamond w_1 \lor \Diamond w_2] \) by PR

Note that because of the universal character of Cl it can be distributed over A (Theorem T7), while 0, which is of existential character can be distributed over V (Theorem T8). Next, we show that interchanging a temporal operator with a boolean operator of the opposite character yields implication in one direction only; the implication is not necessarily true in the other direction.

T9. \( \vdash [\Box w_1 \lor \Box w_2] \equiv [w_1 \lor w_2] \)

Proof:

1. \( \vdash [w_1 \lor w_2] \equiv [\Box w_1 \lor w_2] \) by PT and Cl
2. \( \vdash [w_1 \lor w_2] \equiv [\Box w_1 \lor w_2] \) by PT and 0
3. \( \vdash [w_1 \lor w_2] \equiv [\Box w_1 \lor w_2] \) by 1, 2 and PR

T10. \( \vdash [w_1 A w_2] \equiv [\Diamond w_1 A \Diamond w_2] \)

Proof:

1. \( \vdash [w_1 A w_2] \equiv [\Diamond w_1 A \Diamond w_2] \) by PT and 0
2. \( \vdash [w_1 A w_2] \equiv [\Diamond w_1 A \Diamond w_2] \) by PT and 0
3. \( \vdash [w_1 A w_2] \equiv [\Diamond w_1 A \Diamond w_2] \) by 1, 2 and PR
T11. \( \vdash (\Box w_1 A 0 w_2) \supseteq (w_1 A w_2) \)

Proof:

1. \( \vdash w_1 \supseteq (w_2 \supseteq (w_1 \land w_2)) \) by PT
2. \( \vdash \Box w_1 \supseteq (w_2 \supseteq (w_1 A w_2)) \) by \( \Box \Box \)
3. \( \vdash \Box (w_2 \supseteq (w_1 A w_2)) \supseteq (\Diamond w_2 \supseteq (w_1 A w_2)) \) by T6
4. \( \vdash \Box w_1 \supseteq (\Diamond w_2 \supseteq (w_1 A w_2)) \) by 2, 3 and PR
5. \( \vdash (\Box w_1 A \Diamond w_2) \supseteq (w_1 A w_2) \) by PR

Next we consider the commutativity properties of the next operator 0. In view of A4, 0 is self-dual and can be considered to be of both existential and universal character. Indeed it commutes with every other boolean or temporal operator as well as with quantifiers.

T12. \( \vdash (w_1 A w_2) \equiv (\Diamond w_1 A 0 w_2) \)

Proof:

1. \( \vdash w_1 \supseteq (w_2 \supseteq (w_1 A w_2)) \) by PT
2. \( \vdash O w_1 \supseteq O (w_2 \supseteq (w_1 A w_2)) \) by 0 0
3. \( \vdash O (w_2 \supseteq (w_1 A w_2)) \supseteq (0 w_2 \supseteq O (w_1 A w_2)) \) by A5
4. \( \vdash O w_1 A (O w_2 \supseteq O (w_1 A w_2)) \) by 2, 3 and PR
5. \( \vdash (O w_1 A O w_2) \supseteq O (w_1 A w_2) \) by PR
6. \( \vdash (w_1 \land w_2) \supseteq w_1 \) by PT
7. \( \vdash O (w_1 A w_2) \supseteq O w_1 \) by 0 0
8. \( \vdash (w_1 \land w_2) \supseteq w_2 \) by PT
9. \( \vdash O (w_1 A w_2) \supseteq O w_2 \) by 0 0
10. \( \vdash O (w_1 A w_2) \supseteq (O w_1 A O w_2) \) by 7, 9 and PR
11. \( \vdash O (w_1 A w_2) \equiv (O w_1 A O w_2) \) by 5, 10 and PR

T13. \( \vdash O (w_1 v w_2) \equiv (O w_1 v O w_2) \)
Proof:

1. \( \vdash O(\neg w_1 \land \neg w_2) \equiv [(\neg O w_1) \land (\neg O w_2)] \) by T12
2. \( \vdash O(\neg w_1 \land \neg w_2) \equiv [(\neg O w_1) \land (\neg O w_2)] \) by A4 and PR
3. \( \vdash \neg (w_1 \lor w_2) \equiv [(\neg O w_1) \land (\neg O w_2)] \) by 0 0 and PR
4. \( \vdash \neg O(w_1 \lor w_2) \equiv \neg (O w_1 \lor O w_2) \) by A4 and PR
5. \( \vdash O(w_1 \lor w_2) \equiv (O w_1 \lor O w_2) \) by PR

T14. \( \vdash O(w_1 \lor w_2) \equiv (0 \lor w_1 \lor O w_2) \)

Proof:

1. \( \vdash O(\neg w_1 \lor w_2) \equiv (\neg O w_1) \lor (O w_2) \) by T13
2. \( \vdash O(\neg w_1 \lor w_2) \equiv (\neg O w_1) \lor (O w_2) \) by A4 and PR
3. \( \vdash \neg (w_1 \lor w_2) \equiv (O w_1 \lor O w_2) \) by 0 0 and PR

T15. \( \vdash O(w_1 \equiv w_2) \equiv (O w_1 \equiv 0 \lor w_2) \)

Proof:

1. \( \vdash [O(w_1 \lor w_2) \land O(w_2 \lor w_1)] \equiv [(O w_1 \lor 0 \lor w_2) \land (O w_2 \lor 0 \lor w_1)] \) by T12 and PR
2. \( \vdash O(w_1 \lor w_2) \lor (O w_1 \lor w_2) \equiv [(O w_1 \lor O w_2) \lor (O w_2 \lor O w_1)] \) by 0 0 and PR

The previous theorems show that the next operator, 0, commutes with each of the boolean operators. The following two theorems establish commutation of 0 with the temporal operators Cl and 0.

T16. \( \vdash O \Box w \equiv \Box w \)

Proof:

1. \( \vdash \Box w \equiv (w \equiv O w) \) by PT
2. $\Box \Box w \equiv \Box \Box \Box w$ by A□
3. $\Box \Box \Box w \equiv \Box \Box \Box w$ by A7
4. $\Box \Box (w \equiv \Box w) \equiv \Box (w \equiv \Box w)$ by A8 and 0 0
5. $\Box (w \equiv \Box w) \equiv (\Box w \equiv \Box w)$ by A5
6. $\Box \Box \Box w \equiv (\Box w \equiv \Box \Box w)$
7. $\Box \Box \Box w \equiv o w$
8. $t \equiv o w \equiv o o w$
9. $\Box \Box \Box w \equiv o o o w$
10. $\Box w \equiv \Box w \equiv \Box w$ by A1, A4, C□, 0 0 and PR
11. $\Box w \equiv o w$
12. $\Box \Box \Box w \equiv n o w$
13. $\Box \Box \Box w \equiv \Box \Box w$
14. $\Box \Box \Box w \equiv \Box \Box w$

T 17. $\Box \Box \Box w \equiv \Box \Box w$

Proof:
1. $\Box \Box \Box \sim w \equiv 0 0 - w$ by T16
2. $\Box \Box \Box w \equiv \sim \Box \Box w$ by A1, A4, C□, 0 0 and PR
3. $\Box \Box \Box w \equiv o o w$

T 18. $\Box \Box \Box w \equiv \Box \Box w$

Proof:
1. $\Box \Box \Box w \equiv \Box \Box w$ by A□
2. $t \equiv o w \equiv o u w$
3. $t \equiv o w \equiv \Box \Box \Box w$
4. $t \equiv o w \equiv 0 0 \Box \Box w$
5. $t \equiv o w \equiv 0 0 \Box \Box w$ by 1'17 and PR

17
5. \( \vdash \Box w \equiv \Diamond \Box w \) by 3, 4 and PR

6. \( \vdash \Box w \equiv \Box \Diamond \Box w \) by CI

7. \( \vdash \Box \Diamond \Box w \equiv \Diamond \Box w \) by 1, 6 and PR

T19. \( \vdash \Box \Diamond \Box w \equiv \Box \Diamond w \)

Proof: By duality from T18.

These last two theorems together with T3 and T4 (\( \Box \Diamond w \equiv \Box \Diamond w \equiv \Diamond \Box w \equiv \Diamond \Box w \), respectively) give us a normal prefix form for a string of the form
\[ m_1 m_2 \ldots m_k(w), \]
where each \( m_i \) is either \( \Box \) or \( \Diamond \). We use first T2 and T3 to collapse any substring of the form \( \Box^n \Box \) and \( \Diamond^n \) to a single \( \Box \) or 0. What remains must be a string of alternating \( \Box \) and 0. If it contains more than one operator then it is equivalent by T18 and T19 to a string with just two operators -- the last two. Consequently any string such as the above must be equivalent to one of the following four possibilities:
\[ \Box \Diamond \Box w, \Box \Diamond w, \Box \Diamond \Box w \text{ or } \Box \Diamond \Box w. \]

In the more general case that the string also contains some occurrences of the next-time operator 0, we may use the commutation of 0 with both \( \Box \) and \( \Diamond \) to obtain the four normal forms:
\[ \Diamond^k \Box w, \Diamond^k \Diamond w, \Diamond^k \Box \Diamond w \text{ and } \Diamond^k \Box \Box w \]
for some \( k \geq 0 \).

T20. \( \vdash \Box \Diamond \Box w \equiv (w A \Box \Diamond w) \)

Proof:

1. \( \vdash \Box w \equiv w \) by A3

2. \( \vdash \Box \Diamond w \equiv \Diamond o w \) by A7

3. \( \vdash \Box w \equiv \Box (w A \Box \Diamond w) \) by 1, 2 and PR

4. \( \vdash \Box \Diamond w \equiv \Diamond (w A \Box \Diamond w) \) by A0

5. \( \vdash \Diamond (w A \Box \Diamond w) \equiv \Diamond (w A \Box \Diamond w) \) by PR
6. \( \vdash (w \ A \ O \ w) \supset \Box \ (w \ A \ O \ w) \)  \hspace{2cm} by CI
7. \( \Box (w \ A \ O \ w) \supset \Box \ w \)  \hspace{2cm} by PT and CI
8. \( \vdash (w \ A \ O \ w) \supset \Box \ w \)  \hspace{2cm} by 6, 7 and PR
9. \( \vdash \Box \ w \equiv (w \ A \ O \ w) \)  \hspace{2cm} by 3, 8 and PR

T21. \( \vdash O \ w \equiv (w \ v \ O \ w) \)

Proof:

1. \( \vdash \Box \sim w \equiv (\sim w \ A \ O \sim w) \)  \hspace{2cm} by T20
2. \( \vdash \sim \Diamond w \equiv \sim (w \ v \ - \ 0 \ 0 \ - \ w) \)  \hspace{2cm} by A1 and PR
3. \( \vdash \sim O \sim w \ 3 \ 0 \ 0 \ w \)  \hspace{2cm} by T20, A1, 0 0 and PR
4. \( \vdash \Diamond w \equiv (w \ v \ O \ w) \)  \hspace{2cm} by 2, 3 and PR

Theorems T20 and T21 give a fixpoint characterization of the \( \Box \) and 0 operators respectively. They each give an equation using only boolean operators, the formula \( w \) and the operator 0. The solutions to these equations are CI \( \wedge \) and 0 \( w \) respectively. This shows that in some sense 0 is the most basic operator since the other operators may be defined by means of fixpoint equations using 0. Axiom A9 similarly characterizes the \( \vee \) operator by a fixpoint equation.

7'22. \( \vdash (w \ A \Diamond \sim w) \supset \Diamond (w \ A \ O \sim w) \).

This is the dual of the “computational induction” axiom A8. It states that if \( w \) is true now and is false sometime in the future, then there exists some instant such that \( w \) is true at that instant and false at the next.

Proof:

1. \( \top \Box (w \supset \Box w) \supset \Box \Diamond \Box \Diamond \Diamond \)  \hspace{2cm} by A8
2. \( \top \sim (w \supset \Box \Diamond ) \supset \sim \Box (w \supset O \ w) \)  \hspace{2cm} by PR
3. \( \vdash (w \ A \sim \Box w) \supset \Diamond \sim (w \supset O \ w) \)  \hspace{2cm} by T5 and PR
4. \( \vdash \Diamond \sim (w \supset O \ w) \equiv \Diamond (w \ A \sim \Box w) \)  \hspace{2cm} by PT and 0 0
5. \( \vdash (w \ A \sim \Box w) \supset \Diamond (w \ A \sim O \ w) \)  \hspace{2cm} by 3, 4 and PR
6. \( \vdash (w \ A \Diamond \sim w) \supset \Diamond (w \ A \ O \sim w) \)  \hspace{2cm} by T5, A4 and PR
The following derived rules correspond to proof rules existing in most axiomatic verification systems:

<table>
<thead>
<tr>
<th>Consequence Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>□Q rule</td>
</tr>
<tr>
<td>( \vdash u_1 \triangleright u_2 )</td>
</tr>
<tr>
<td>( \vdash u_2 \triangleright v_2 )</td>
</tr>
<tr>
<td>( \vdash u_1 \triangleright u_2 )</td>
</tr>
<tr>
<td>( \vdash v_1 \triangleright v_2 )</td>
</tr>
</tbody>
</table>

Proof of \( \Diamond Q \):

1. \( \vdash u_1 \triangleright u_2 \) given
2. \( \vdash u_2 \triangleright v_1 \) given
3. \( \vdash v_1 \triangleright v_2 \) given
4. \( \vdash \Diamond v_1 \triangleright \Diamond v_2 \) by 3 and 0 0
5. \( \vdash u_1 \triangleright \Diamond v_2 \) by 2, 4 and PR

The \( \Box \) Q and \( \Box Q \) rules are proved similarly by the \( \Box Cl \)-rule and 0 O-rule, respectively.

<table>
<thead>
<tr>
<th>Concatenation Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Box ) rule</td>
</tr>
<tr>
<td>( \vdash u \triangleright \Box v )</td>
</tr>
<tr>
<td>( \vdash v \triangleright \Box w )</td>
</tr>
<tr>
<td>( \vdash u \triangleright \Box w )</td>
</tr>
</tbody>
</table>

Proof of \( \Diamond C \):

1. \( \vdash u \triangleright \Box v \) given
2. \( \vdash v \triangleright \Box w \) given
3. \( \vdash \Box v \triangleright \Box w \) by 2 and \( \Box Cl \)
4. \( \vdash \Box v \triangleright \Box * \) by T3 and \( PR \)
5. \( \vdash u \triangleright \Box w \) by 1, 4 and \( PR \)

The OC rule is proved similarly by the 0 O-rule. Note that the corresponding OC rule does not hold.
Until Derived Rules and Theorems:

**Right Until Introduction -- RUI**

\[
\begin{align*}
1 & : w \rightarrow w \\
1 & : [v \rightarrow (u \rightarrow w)] \\
\hline
1 & : (u \rightarrow v)
\end{align*}
\]

Proof:

1. \( t - w \rightarrow \Diamond v \) given

2. \( \vdash w \supset [v \rightarrow (u \rightarrow w)] \) given

3. \( 1 - [v \rightarrow (u \rightarrow O(u \rightarrow v))] \supset (u \rightarrow v) \)

4. \( \vdash \sim(u \rightarrow v) \supset [\sim v \rightarrow (u \rightarrow v)] \)

5. \( \vdash [w \rightarrow (u \rightarrow v)] \supset [\sim v \rightarrow O(w \rightarrow O(u \rightarrow v))] \)

6. \( \vdash [w \rightarrow (u \rightarrow v)] \supset [\sim v \rightarrow O(w \rightarrow \sim(u \rightarrow v))] \)

7. \( t - [w \rightarrow (u \rightarrow v)] \supset \Diamond \sim v \)

8. \( t - [w \rightarrow (u \rightarrow v)] \supset \Diamond \sim v \)

9. \( \vdash w \supset (u \rightarrow v) \)

The RUI rule, together with axioms A9 and A10, can be viewed as a characterization of the \( u \rightarrow v \) construct as a maximal solution of the two implications:

\[
(x) \begin{cases} x \supset [v \rightarrow (u \rightarrow x)] \\ x \supset \Diamond v \end{cases}
\]

The ordering by which maximality is defined is the ordering induced by defining \( \text{false} \sqsubseteq \text{true} \).

Axioms A9 and A10 imply that

\[
(u \rightarrow v) \supset [v \rightarrow (u \rightarrow O(u \rightarrow v))]
\]

\[
(u \rightarrow v) \supset O \sim v
\]

Thus they show \( x = u \rightarrow v \) to be a solution of the implications (t). The rule RUI states that any other solution \( x = w \) must satisfy \( w \supset (u \rightarrow v) \) which implies that whenever \( w \) is true so is \( u \rightarrow v \). Interpreted in our ordering this is representable as \( w \supset (u \rightarrow v) \). Thus \( x = u \rightarrow v \) is the maximal solution to (x).

An intuitive explanation as to why \( u \rightarrow v \) is indeed the maximal solution of (x) can be given as follows:
Let \( w \) be any proposition satisfying (t) everywhere in a sequence \( \sigma = s_0, s_1, \ldots \). We note that (s) may have many solutions. In particular \( x = \text{false} \) is a trivial solution. However, an obvious property of every solution \( w \) is that if \( w \) is true in some state \( s_i \), this state must satisfy \( u \) and the next state \( s_{i+1} \) must also satisfy \( w \) unless \( s_i \) satisfies \( v \). Thus once \( w \) is true it can stop being true only in a v-state. In view of the second implication such a v-state is guaranteed. Consequently whenever \( w \) is true in a state, \( u \cup v \) must also be true in that state.

\[
\text{Left Until Introduction} \quad \vdash \quad (u \cup v) \supset w
\]

Proof:

1. \( \vdash [v \supset (u \circ w)] \supset w \) \hspace{1cm} \text{given}
2. \( \vdash u \cup v \supset [v \supset (u \circ (u \cup v))] \) \hspace{1cm} by A9 and PR
3. \( \vdash \sim w \supset [\sim v \supset (\sim v \circ 0 \sim w)] \) \hspace{1cm} by 1, A4 and PR
4. \( \vdash [u \cup v \circ \sim w] \supset [\sim v \supset (u \circ (u \cup v)) \circ \circ w] \) \hspace{1cm} by 2, 3 and PR
5. \( \vdash [u \cup v \circ \sim w] \supset [0 (u \cup v) \circ \circ w] \) \hspace{1cm} by PR
6. \( \vdash [u \cup v \circ \sim w] \supset [0 (u \cup v) \circ \circ w] \) \hspace{1cm} by T12 and PR
7. \( \vdash [u \cup v \circ \sim w] \supset 0 (u \cup v) \circ \circ w \) \hspace{1cm} by CI
8. \( \vdash [u \cup v \circ \sim w] \supset \sim v \) \hspace{1cm} by 3 and PR
9. \( \vdash 0 (u \cup v \circ \sim w) \supset \circ \circ \sim v \) \hspace{1cm} by \( 0 0 \)
10. \( \vdash [u \cup v \circ \sim w] \supset \sim \circ v \) \hspace{1cm} by 7, 9, A1 and PR
11. \( \vdash [u \cup v \circ \sim w] \supset \circ v \) \hspace{1cm} by A10 and PR
12. \( \vdash u \cup v \supset w \) \hspace{1cm} by 10, 11 and PR

The \( u \cup v \) rule, together with axiom A9, can be viewed as a characterization of the \( u \cup v \) construct as the minimal solution of the implication:

\[
(*) \quad [v \supset (u \circ x)] \supset x
\]

Axiom A9 implies that \( x = u \cup v \) is a solution of \((*)\). The \( u \cup v \) rule states that any other solution of \((*)\), \( x = w \), is implied by \( u \cup v \). This means that whenever \( u \cup v \) is true so is \( w \), which is interpretable in our ordering as \( u \cup v \supset w \). Thus \( u \cup v \) is the minimal of all possible solutions.

Note that \((*)\) possesses many solutions. In particular \( x = \text{true} \) is a trivial solution. However, the minimal solution is unique and is given by \( u \cup v \).

22
Rules

\[ \vdash u_1 \supset u_2 \quad \vdash u_1 \equiv u_2 \]
\[ (a) \quad \vdash v_1 \supset v_2 \quad (b) \quad \vdash v_1 \equiv v_2 \]
\[ \vdash u_1 \cup v_1 \supset u_2 \cup v_2 \quad \vdash u_1 \cup v_1 \equiv u_2 \cup v_2 \]

Proof of (a):

1. \[ \vdash u_1 \supset u_2 \] given
2. \[ \vdash v_1 \supset v_2 \] given
3. \[ \vdash [v_2 \lor (u_2 \land \Box (u_2 \cup v_2))] \supset u_2 \cup v_2 \] by A9
4. \[ \vdash [v_1 \lor (u_1 \land \Box (u_2 \cup v_2))] \supset u_2 \cup v_2 \] by 1, 2, 3 and PR
5. \[ \vdash u_1 \cup v_1 \supset u_2 \cup v_2 \] by \( \Box \cup I \)

The proof of part (b) follows from (a) by propositional reasoning and the symmetric application of (a).

This rule together with the \( \Box \), \( \Diamond \), \( \Box \), \( \Diamond \), \( \Box \), \( \Diamond \), rules show that all the temporal operators are monotonic in all their arguments.

T23. \[ \vdash (\sim w) \cup w \equiv \Diamond w \]

Proof:

1. \[ \vdash (\sim w) \cup w \supset o w \] by A10
2. \[ \vdash \Diamond w \supset [w \lor \Box \Diamond w] \] by T21 and PR
3. \[ \vdash o w \supset [w \lor (\sim w \land \Box \Diamond w)] \] by PR
4. \[ \vdash \Diamond w \supset o w \] by PT
5. \[ \vdash o w \supset (\sim w) \cup w \] by 3, 4 and RUI
6. \[ \vdash (\sim w) \cup w \equiv o w \] by 1, 5 and PR

T24. \[ \vdash (\Box w_1 \land \Diamond w_2) \supset (w_1 \cup w_2) \]

Proof:

1. \[ \vdash [\Box w_1 \land \Diamond w_2] \supset \Diamond w_2 \] by PR
2. \( \vdash [\Box w_1 A \Diamond w_2] \supset [(w_1 A \Box w_1) \land (w_2 \lor \Diamond w_2)] \) by PR, T20 and T21

3. \( \vdash (\Box w_1 A \Diamond w_2) \supset [w_2 \lor (w_1 A \Box w_1 A \Diamond w_2)] \) by PR

4. \( \vdash (\Box w_1 A \Diamond w_2) \supset [w_2 \lor (w_1 \land \Box (\Box w_1 \lor \Diamond w_2))] \) by T12 and PR

5. \( \vdash [\Box w_1 A \Diamond w_2] \supset w_1 \lor w_2 \) by 1, 4 and RUI, taking \( w \) to be \( w_1 A 0 w_2 \), \( u \) to be \( w_1 \), and \( v \) to be \( w_2 \)

T25. \( \vdash (w_1 \lor w_2) \lor w_2 \equiv w_1 \lor w_2 \)

**Proof:**

1. \( \vdash (w_1 \lor w_2) \lor w_2 \supset [w_2 \lor w_1 \lor w_2] \) by A9 and PR

2. \( \vdash w_2 \supset w_1 \lor w_2 \) by A9 and PR

3. \( \vdash (w_1 \lor w_2) \lor w_2 \supset w_1 \lor w_2 \) by 1, 2 and PR

4. \( \vdash w_1 \lor w_2 \supset \Box w_2 \) by A10

5. \( \vdash w_1 \lor w_2 \supset [w_2 \lor (w_1 A \Box (w_1 \lor w_2))] \) by A9 and PR

6. \( \vdash w_1 \lor w_2 \supset [w_2 \lor (w_1 \lor w_2 A \Box (w_1 \lor w_2))] \) by PR

7. \( \vdash w_1 \lor w_2 \supset (w_1 \lor w_2) \lor w_2 \) by 4, 6 and RUI

8. \( \vdash (w_1 \lor w_2) \lor w_2 \equiv w_1 \lor w_2 \) by 3, 7 and PR

T26. \( \vdash w_1 \lor w_2 \equiv w_1 \lor (w_1 \lor w_2) \)

**Proof:**

1. \( \vdash w_2 \supset w_1 \lor w_2 \) by A9 and PR

2. \( \vdash w_1 \lor w_2 \supset w_1 \lor (w_1 \lor w_2) \) by UU

3. \( \vdash w_1 \lor (w_1 \lor w_2) \supset [w_1 \lor w_2 \lor [(w_1 A \Box (w_1 \lor w_2))]] \) by A9 and PR

4. \( \vdash w_1 \lor (w_1 \lor w_2) \supset [w_2 \lor [w_1 \land \Box (w_1 \lor w_2)] \lor [w_1 \land \Box (w_1 \lor w_2)]] \) by A9 and PR

5. \( \vdash w_1 \lor (w_1 \lor w_2) \supset [w_2 \lor [w_1 A \Box (w_1 \lor w_2) \lor w_1 \lor (w_1 \lor w_2)]] \) by T13 and PR

6. \( \vdash [w_1 \lor w_2 \lor w_1 \lor (w_1 \lor w_2)] \supset w_1 \lor (w_1 \lor w_2) \) by 2 and PR
7. \( \vdash w_1 \cup (w_1 \cup w_2) \supset \{ w_2 \lor [w_1 A \circ (w_1 \cup (w_1 \cup w_2))] \} \) by 6 with 0, 0, and PR

8. \( \vdash w_1 \cup (w_1 \cup w_2) \supset \diamond (w_1 \cup w_2) \) by A10

9. \( \vdash w_1 \cup w_2 \supset \diamond w_2 \) by A10

10. \( \lor \diamond (w_1 \cup w_2) \supset 0 \diamond w_2 \) by 0, 0

11. \( \vdash w_1 \cup (w_1 \cup w_2) \supset \diamond w_2 \) by 8, 10, T4 and PR

12. \( \vdash w_1 \cup (w_1 \cup w_2) \supset w_1 \cup w_2 \) by 11, 7 and RUI, taking \( w \) to be \( w_1 \cup (w_1 \cup w_2) \), \( u \) to be \( w_1 \), and \( v \) to be \( w_2 \)

15. \( \vdash w_1 \cup w_2 \equiv w_1 \cup (w_1 \cup w_2) \) by 2, 12 and PR

\[
\begin{array}{c|c|c}
\hline
\text{U} & \text{Insertion -- UI} & \text{U} \\
\hline
(a) & \frac{\vdash v}{\vdash \cup v} & (b) \frac{\vdash u \cup v}{\vdash \cup u v} \\
& \text{for an arbitrary } u & \\
\hline
\end{array}
\]

Proof:

(a)
1. \( \vdash v \) given
2. \( \vdash v \lor u v \) by A9 and PR
3. \( \vdash u v \) by 1, 2 and PR

(b)
1. \( \vdash u \) given
2. \( \vdash \cup v \) given
3. \( \vdash \downarrow \) by 1 and \( \downarrow \)
4. \( \vdash (\square u A \circ v) \supset u v \) by T24
5. \( \vdash u v \) by 2, 3, 4 and PR

\[
\begin{array}{c|c|c|c}
\hline
\text{U} & \text{Concatenation -- UC} & \text{U} \\
\hline
\vdash v_1 \lor v_2 & \vdash v_1 \lor v_2 & \vdash v_1 \lor v_2 \\
\hline
\end{array}
\]

25
Proof:

1. $\vdash v_1 \cup u \cup v_2$
2. $\vdash v_2 \cup u \cup v_3$
3. $\vdash u \cup v_2 \cup u \cup (u \cup v_3)$
4. $\vdash v_1 \cup u \cup (u \cup v_3)$
5. $\vdash v_1 \cup u \cup v_3$

given

by $UU$

by 1, 3 and PR

by T26 and PR

$T27. \vdash [\square w_1 A w_2 \cup w_3] \supset (w_1 A w_2) \cup (w_1 \wedge w_3)$

Proof:

1. $\vdash w_2 \cup w_3 \supset \bigodot w_3$
2. $\vdash [\square w_1 A w_2 \cup w_3] \supset (\square w_1 A \bigodot w_3)$
3. $\vdash [\square w_1 A w_2 \cup w_3] \supset \bigodot (w_1 A w_3)$
4. $\vdash w_2 \cup w_3 \supset [w_3 \vee (w_2 A \bigcirc (w_2 \cup w_3))]$
5. $\vdash [\square w_1 A w_2 \cup w_3] \supset [[\square w_1 A w_3] \vee (\square w_1 A w_2 \bigcirc (w_2 \cup w_3))]$
6. $\vdash (\square w_1 A w_3) \supset (w_1 A w_3)$
7. $\vdash [\square w_1 A w_2 A \bigcirc (w_2 \cup w_3)] \supset [w_1 A w_2 A \bigcirc (w_2 \cup w_3)]$
8. $\vdash [\square w_1 A w_2 A \bigcirc (w_2 \cup w_3)] \supset [(w_1 A w_2) A \bigcirc (\square w_1 A w_2 \cup w_3)]$
9. $\vdash [\square w_1 A w_2 \cup w_3] \supset \{(w_1 A w_3) \vee [(w_1 A w_2) A \bigcirc (\square w_1 A w_2 \cup w_3)]\}$
10. $\vdash [\square w_1 A w_2 \cup w_3] \supset (w_1 A w_2) \cup (w_1 A w_3)$

by A10

by PR

by T11 and PR

by A9 and PR

by PR

by A3 and PR

by T20 and $PR$

by T12 and $PR$

by 5, 6, 8 and $PR$

by 3, 9 and $RUL$

The next theorem displays the commutation relation between the $\bigodot$ and the $\cup$ operators.

$T28. \vdash (\bigodot w_1) \cup (\bigodot w_2) \equiv \bigodot (w_1 \cup w_2)$

Proof:

1. $\vdash w_1 \cup w_2 \equiv [w_2 \vee (w_1 A \bigcirc (w_1 \cup w_2))]$

by A9
2. \( \vdash O(w_1 \cup w_2) \equiv \{ w \cup (O w_1 \cup O(w_1 \cup w_2)) \} \) by T12, T13, 0 0 and PR

3. \( \vdash [O w_2 \lor (O w_1 \cup O(w_1 \cup w_2))] \supset O(w_1 \cup w_2) \) by PR

4. \( \vdash (O w_1) \cup (O w_2) \supset O(w_1 \cup w_2) \) by L\( \cup \), taking \( w \) to be \( w_1 \cup w_2 \)

5. \( \vdash w_1 \cup w_2 \supset \Diamond w_2 \) by A10

6. \( \vdash O(w_1 \cup w_2) \supset O \Diamond w_2 \) by 0 0

7. \( \vdash O(w_1 \cup w_2) \supset \Diamond O w_2 \) by T17 and PR

8. \( \vdash O(w_1 \cup w_2) \supset \{ O w_2 \lor [O w_1 \land O(w_1 \cup w_2)] \} \) by 2 and PR

9. \( \vdash O(w_1 \cup w_2) \supset (O w_1) \cup (O w_2) \) by 7, 8 and RUT, taking \( w \) to be \( O(w_1 \cup w_2) \), \( u \) to be 0 \( w_1 \), and \( v \) to be 0 \( w_2 \)

10. \( \vdash (O w_1) \cup (O w_2) \equiv O(w_1 \cup w_2) \) by 4, 9 and PR

Having classified \( \Box \) as a universal operator, 0 as an existential operator and 0 as being both universal and existential, we observe that \( U \) is universal with respect to its first argument and existential with respect to its second argument. This yields the commutation properties listed in T29 and T30.

**T29.** \( \vdash (w_1 \land w_2) \cup w_3 \equiv [w_1 \cup w_3 \land w_2 \cup w_3] \)

**Proof:**

1. \( \vdash (w_1 \land w_2) \supset w_1 \) by PT

2. \( \vdash (w_1 \land w_2) \cup w_3 \supset w_1 \cup w_3 \) by UU

3. \( \vdash (w_1 \land w_2) \cup w_3 \supset w_2 \cup w_3 \) similarly

4. \( \vdash (w_1 \land w_2) \cup w_3 \supset [w_1 \cup w_3 \land w_2 \cup w_3] \) by 2, 3 and PR

5. \( \vdash w_1 \cup w_3 \supset \Diamond w_3 \) by A10

6. \( \vdash [w_1 \cup w_3 \land w_2 \cup w_3] \supset \Diamond w_3 \) by PR

7. \( \vdash w_1 \cup w_3 \supset \{ w_3 \lor [w_1 \land O(w_1 \cup w_3)] \} \) by A9 and 1'11

8. \( \vdash w_2 \cup w_3 \supset \{ w_3 \lor [w_2 \land O(w_2 \cup w_3)] \} \) by A9 and PR

9. \( \vdash [w_1 \cup w_3 \land w_2 \cup w_3] \supset \{ w_3 \lor [(w_1 \land w_2) \land O(w_1 \cup w_3 \land w_2 \cup w_3)] \} \) by 7, 8, T12 and PR

10. \( \vdash [w_1 \cup w_3 \land w_2 \cup w_3] \supset (w_1 \land w_2) \cup w_3 \) by 6, 9 and R\( \cup \), taking \( w \) to be \( (w_1 \cup w_3) \land (w_2 \cup w_3) \), \( u \) to be \( w_1 \land w_2 \), and \( v \) to be \( w_3 \)
11. $\vdash (w_1 \land w_2) \lor w_3 \equiv [w_1 \lor w_3 \land w_2 \lor w_3]$ by 4, LO and PR

T30. $\vdash w_1 \lor (w_2 \lor w_3) \equiv [w_1 \lor w_2 \lor w_1 \lor w_3]$

Proof:

1. $\vdash w_2 \lor (w_2 \lor w_3)$ by PT
2. $\vdash w_1 \lor w_2 \lor w_2 \lor w_3$ by UU
3. $\vdash w_1 \lor w_3 \lor w_2 \lor w_3$ similarly
4. $\vdash [w_1 \lor w_2 \lor w_1 \lor w_3] \lor w_1 \lor (w_2 \lor w_3)$ by 2, 3 and PR
5. $\vdash w_1 \lor (w_2 \lor w_3) \lor \{w_2 \lor w_3 \lor w_1 \lor ((w_2 \lor w_3) \lor w_3)\}$ by A9 and PR
6. $\vdash [w_2 \lor (w_1 \lor (w_1 \lor w_2))] \lor w_1 \lor w_2$ by A9 and PR
7. $\vdash \sim w_1 \lor w_2 \lor \{\sim w_2 \lor w_3 \lor \sim w_1 \lor (\sim w_1 \lor w_2)\}$ by A4 and PR
8. $\vdash \sim w_1 \lor w_2 \lor \{\sim w_3 \lor \sim w_1 \lor (\sim w_1 \lor w_3)\}$ similarly
9. $\vdash [w_1 \lor (w_2 \lor w_3) \lor w_1 \lor (w_2 \lor w_3) \lor w_1 \lor w_3] \lor \{\sim w_2 \lor w_3 \lor w_1 \lor (w_2 \lor w_3) \lor w_1 \lor w_3\}$ by 5, 7, 8 and PR
10. $\vdash [w_1 \ lor (w_2 \lor w_3) \lor (w_1 \lor w_2) \lor (w_1 \lor w_3)] \lor \{\sim (w_2 \lor w_3) \lor w_1 \lor (w_2 \lor w_3) \lor w_1 \lor w_3\}$ by T12 and PR
11. $\vdash [w_1 \lor (w_2 \lor w_3) \lor (w_1 \lor w_2) \lor (w_1 \lor w_3)] \lor \{\sim (w_2 \lor w_3) \lor w_1 \lor (w_2 \lor w_3) \lor w_1 \lor w_3\}$ by T12 and PR
12. $\vdash w_1 \lor (w_2 \lor w_3) \lor (w_2 \lor w_3)$ by 10
13. $\vdash w_1 \lor (w_2 \lor w_3) \lor (w_1 \lor w_2) \lor (w_1 \lor w_3)$ by 11, 12, A1 and PR
14. $\vdash w_1 \lor (w_2 \lor w_3) \lor (w_1 \lor w_2) \lor (w_1 \lor w_3)$ by PR
15. $\vdash w_1 \lor (w_2 \lor w_3) \lor (w_1 \lor w_2) \lor (w_1 \lor w_3)$ by 4, 14 and PR

T31. $\vdash [\xcirc w_1 \lor \xcirc w_2] \lor [(\sim w_1) \lor w_2 \lor (\sim w_2) \lor w_1]$

Proof:

1. $\vdash [\xcirc w_1 \lor \xcirc w_2] \lor [\xcirc w_1 \lor w_2]$ by T8 and PR
2. \( \vdash \diamond (w_1 \lor w_2) \supset (\sim (w_1 \lor w_2)) \cup (w_1 \lor w_2) \) by T23 and PR
3. \( \vdash \diamond (w_1 \lor w_2) \supset (\sim w_1 \land \sim w_2) \cup (w_1 \lor w_2) \) by UU and PR
4. \( \vdash \diamond (w_1 \lor w_2) \supset \{ [\lnot w_1 \lor \lnot w_2] \cup \lnot w_1 \lor \{ \lnot w_1 \lor \lnot w_2 \} \} \cup w_2 \) by T30 and PR
5. \( \vdash (\sim w_1 \land \sim w_2) \cup w_1 \supset (\sim w_2) \cup w_1 \) by \( \land \cup \) and PR
6. \( \vdash (\sim w_1 \land \sim w_2) \cup w_1 \supset (\sim w_1) \cup w_2 \) by \( \lor \cup \) and PR
7. \( \vdash (\sim w_1 \lor w_2) \supset \{ [\lnot w_1 \cup w_2] \lor \{ \lnot w_2 \cup w_1 \} \} \) by 4, 5, 6, and PR
8. \( \vdash (\diamond (w_1 \lor w_2)) \supset \{ [\lnot w_1 \cup w_2] \lor \{ \lnot w_2 \cup w_1 \} \} \) by 1, 7, and PR

The following two theorems display the one way implication resulting from the interchange of the U with a boolean operator of the opposite character.

1'32. \( \vdash w_1 \cup (w_2 \land w_3) \supset (w_1 \cup w_2) \land w_3 \)

Proof:
1. \( \vdash (w_2 \land w_3) \supset w_2 \) by PT
2. \( \vdash w_1 \cup (w_2 \land w_3) \supset w_1 \cup w_2 \) by \( \lor \cup \) and PR
3. \( \vdash w_1 \cup (w_2 \land w_3) \supset w_1 \cup w_3 \) similarly
4. \( \vdash w_1 \cup (w_2 \land w_3) \supset [w_1 \cup w_2 \land w_1 \cup w_3] \) by 2, 3, and PR

T33. \( \vdash [w_1 \cup w_3 \lor w_2 \lor w_3] \supset (w_1 \lor w_2) \lor w_3 \)

Proof:
1. \( \vdash w_1 \supset (w_1 \lor w_2) \) by IT
2. \( \vdash w_1 \cup w_3 \supset (w_1 \lor w_2) \cup w_3 \) by UU
3. \( \vdash w_2 \supset (w_1 \lor w_2) \) by PT
4. \( \vdash w_2 \cup w_3 \supset (w_1 \lor w_2) \cup w_3 \) by UU
5. \( \vdash [w_1 \cup w_3 \lor w_2 \cup w_3] \supset (w_1 \lor w_2) \lor w_3 \) by 2, 4, and PR

T34. \( \vdash (w_1 \supset w_2) \lor w_3 \supset [w_1 \lor w_3 \supset w_2 \lor w_3] \)
Proof:

1. \( \vdash (w_1 \lor w_2) \cup w_3 \supset \Diamond w_3 \) by \( \Lambda 10 \)

2. \( \vdash [(w_1 \lor w_2) \cup w_3] \supset \{w_3 \lor [(w_1 \supset w_2) \supset \circ((w_1 \lor w_2) \cup w_3)] \land w_1 \land \circ(w_1 \cup w_3)\} \) by \( \Lambda 9 \) and PR

3. \( \vdash [(w_1 \supset w_2) \cup w_3] \supset \{w_3 \lor [w_2 \land \circ((w_1 \lor w_2) \cup w_3)] \land \circ(w_1 \cup w_3)\} \) by PR

4. \( \vdash [(w_1 \lor w_2) \cup w_3] \supset \{w_3 \lor [w_2 \land \circ((w_1 \lor w_2) \cup w_3)] \land w_1 \cup w_3\} \) by \( T12 \) and PR

5. \( \vdash [(w_1 \supset w_2) \cup w_3] \supset \{w_3 \lor [w_2 \land \circ((w_1 \lor w_2) \cup w_3)] \land (w_1 \cup w_3), u \text{ to be } w_2, \text{ and } v \text{ to be } w_3\} \)

6. \( \vdash (w_1 \lor w_2) \cup w_3 \supset \{w_1 \cup w_3 \supset w_2 \cup w_3\} \) by PR

T35. \( \vdash [w_1 \cup w_2 \land (\neg w_2) \cup w_3] \supset w_1 \cup w_3 \)

Proof:

1. \( \vdash (\neg w_2) \cup w_3 \supset w_3 \) by \( \Lambda 10 \)

2. \( \vdash [w_1 \cup w_2 \land (\neg w_2) \cup w_3] \supset \Diamond w_3 \) by PR

3. \( \vdash w_1 \cup w_2 \supset \{w_2 \lor [w_1 \land \circ(w_1 \cup w_2)]\} \) by \( \Lambda 9 \) and PR

4. \( \vdash (\neg w_2) \cup w_3 \supset \{w_3 \lor [\neg w_2 \land \circ((\neg w_2) \cup w_3)]\} \) by \( \Lambda 9 \) and PR

5. \( \vdash [w_1 \cup w_2 \land (\neg w_2) \cup w_3] \supset \{w_3 \lor [w_1 \land \circ(w_1 \cup w_2)] \lor [(\neg w_2) \cup w_3] \land \circ((\neg w_2) \cup w_3)]\} \) by 3, 4 and PR

6. \( \vdash [w_1 \cup w_2 \land (\neg w_2) \cup w_3] \supset \{w_3 \lor [w_1 \land \circ(w_1 \cup w_2) \land (\neg w_2) \cup w_3)]\} \) by \( T12 \) and PR

7. \( \vdash [w_1 \cup w_2 \land (\neg w_2) \cup w_3] \supset w_1 \cup w_3 \) by 2, 6 and RUI

T36. \( \vdash w_1 \cup (w_2 \land w_3) \supset (w_1 \cup w_2) \cup w_3 \)

Proof:

1. \( \vdash w_1 \cup (w_2 \land w_3) \supset \Diamond (w_2 \land w_3) \) by \( \Lambda 10 \)
2. \( \vdash (w_2 \land w_3) \supset w_3 \) by PT
3. \( \vdash \Diamond (w_2 \land w_3) \supset \Diamond w_3 \) by 0 0
4. \( \vdash w_1 \cup (w_2 \land w_3) \supset \Diamond w_3 \) by 1, 3 and PR
5. \( \vdash (w_2 \land w_3) \supset \{(w_2 \land w_3) \lor (w_1 \land \Diamond (w_1 \cup (w_2 \land w_3)))\} \) by A9 and PR
6. \( \vdash (w_2 \land w_3) \supset w_2 \) by PT
7. \( \vdash w_1 \cup (w_2 \land w_3) \supset w_1 \cup w_2 \) by UU
8. \( \vdash w_1 \cup (w_2 \land w_3) \supset \{w_3 \lor [w_1 \cup w_2 \land \Diamond (w_1 \cup (w_2 \land w_3))]\} \) by 5, 7 and PR
9. \( \vdash w_1 \cup (w_2 \land w_3) \supset (w_1 \cup w_2) \cup w_3 \) by 4, 8 and RUI

The following two theorems are referred to as “collapsing” theorems, since they may be used to derive a consequence of smaller nesting depth from a nested until expression.

**T37.** \( \vdash (w_1 \cup w_2) \cup w_3 \supset (w_1 \lor w_2) \cup w_3 \)

**Proof:**

1. \( \vdash w_1 \cup w_2 \supset [w_2 \lor (w_1 \land \Diamond (w_1 \cup w_2))] \) by A9 and PR
2. \( \vdash w_1 \cup w_2 \supset (w_1 \lor w_2) \) by PR
3. \( \vdash (w_1 \cup w_2) \cup w_3 \supset (w_1 \lor w_2) \cup w_3 \) by UU

**T38.** \( \vdash w_1 \cup (w_2 \cup w_3) \supset (w_1 \lor w_2) \cup w_3 \)

**Proof:**

1. \( \vdash w_1 \cup (w_2 \cup w_3) \supset \Diamond (w_2 \lor w_3) \) by A10
2. \( \vdash w_2 \lor w_3 \supset 0 \lor w_3 \) by A10
3. \( \vdash w_1 \cup (w_2 \lor w_3) \supset 0 \lor w_3 \) by 1, 2 and \( \Diamond C \)
4. \( \vdash w_1 \cup (w_2 \lor w_3) \supset \{w_2 \lor w_3 \lor [w_1 \land \Diamond (w_1 \cup (w_2 \lor w_3))]\} \) by A9 and PR
5. \( \vdash w_1 \cup (w_2 \lor w_3) \supset \{w_3 \lor [w_2 \land \Diamond (w_2 \lor w_3)] \lor [w_1 \land \Diamond (w_1 \cup (w_2 \lor w_3))]\} \) by A9 and PR
6. \( \vdash w_2 \lor w_3 \supset w_1 \cup (w_2 \lor w_3) \) by A9 and PR
7. \( \vdash [w_2 \land \Diamond (w_2 \lor w_3)] \supset [(w_1 \lor w_2) \land \Diamond (w_1 \cup (w_2 \lor w_3))] \) by 0 0 and PR
A very useful derived operator is the unless operator \( u \uplus v \) being defined by

\[
u \uplus v \equiv [\square u \lor (u \uplus v)].\]

The unless operator does not insist on the fact that \( v \) actually happens but it requires that \( u \) holds until such an occurrence. If \( v \) never happens \( u \) must hold forever. This operator is related to the binary “as long as” operator \( p \square q \), reading “\( q \) as long as \( p \)” introduced by Lamport in [1,2]. The meaning of this construct is that \( q \) holds continuously as long as \( p \) is continuously maintained. We may express \( p \square q \) by:

\[
p \square q \equiv q \uplus (\neg p).
\]

Following is a rule for establishing the unless operator.

**Unless Introduction — \( \uplus \)**

\[
\begin{array}{c}
\vdash u \supset O(u \lor v) \\
\vdash u \supset (u \uplus v)
\end{array}
\]

**Proof:**

1. \( \vdash u \supset O(u \lor v) \) \( \quad \) given
2. \( \vdash u \supset [O u \lor O v] \) \( \quad \) by T13
3. \( \vdash \neg(u \uplus v) \supset \{ \neg v \quad \neg u \lor O \neg(u \uplus v) \} \) \( \quad \) by A9, T4 and PR
4. \( \vdash \neg(u \uplus v) \supset O \neg v \) \( \quad \) by 0 0 and PR
5. \( \vdash [u A \neg(u \uplus v)] \supset [u A O \neg(u \uplus v)] \) \( \quad \) by 3 and PR
6. \( \vdash [u A \neg(u \uplus v)] \supset [u A O(u \uplus v) A \sim O v] \) \( \quad \) by 4, 5, A4 and PR
7. \( \vdash [u A \neg(u \uplus v)] \supset [u A O u A O \sim (u \uplus v)] \) \( \quad \) by 2, 6 and PR
8. \( \vdash [u A \neg(u \uplus v)] \supset [u A O(u A \sim (u \uplus v))] \) \( \quad \) by T7 and PR
9. \( \vdash [u A \neg(u \uplus v)] \supset \square u \) \( \quad \) by DCI
10. \( \vdash u \supset (\square u \lor (u \uplus v)) \) \( \quad \) by PR
11. \( t \rightarrow u \supset (u \lor v) \) by definition of \( \supset \)

This concludes the description of the propositional section of general temporal logic. The axiomatic system presented for this section of the logic is known to be complete, and the validity problem decidable ([IS]). Consequently, there exists a procedure that tests each formula in PTL (Propositional Temporal Logic) for validity, and constructs a proof in the presented system if the statement is valid. The procedure given in [PS] takes exponential time in the size of the tested formula.

4. QUANTIFIERS

Since we intend to use terms and predicates in our reasoning we have to extend our system to admit individual variables, terms and quantification. Let us consider additional axioms involving quantifiers and their interaction with the temporal operators.

AXIOMS:

\[
\begin{align*}
A11. & \quad \vdash 3 \exists w \equiv \forall x. \sim w \\
A12. & \quad \vdash (\forall x. w(x)) \supset w(t) \\
& \quad \text{where } t \text{ is any term globally free for } x \text{ in } w \\
A13. & \quad \vdash (\forall x. 0 w) \supset (0 \forall x. w)
\end{align*}
\]

In these axioms, \( x \) is any global individual variable. Axioms \( A11 \) and \( A12 \) are the usual predicate calculus axioms: \( A11 \) defines \( 3 \) as the dual of \( \forall \) and \( A12 \) is the instantiation axiom. Axiom \( A13 \) is the Barcan formula for the \( 0 \) operator; it states that since both operators \( \forall \) and \( 0 \) have universal characteristics they commute. We use the substitution notation \( w(t) \) replaced by \( w(t) \) to denote the substitution of the term \( t \) for all free occurrences of \( x \) in \( w \).

A-term \( t \) is said to be globally free for \( x \) in \( w \) if substitution of \( t \) for all free occurrences of \( x \) in \( w \); (a) does not create new bound occurrences of (global) variables, and (b) does not create new occurrences of local variables in the scope of a temporal operator. A trivial case: if \( t \) is \( x \) itself, then \( t \) is free for \( x \). Condition (a) is the one stipulated in classical predicate logic. Condition (b) is special to modal and temporal logics with quantification. Condition (b) is essential for \( A12 \), because without it we could derive the formula

\[
(\forall x. \Diamond (x < y)) \supset \Diamond (y < y),
\]

which is not valid for a local variable \( y \).

An additional rule of inference is:
**INFERENCE RULE:**

\[
\text{R4. } \forall \text{ Insertion} \quad \forall I
\]

\[
\frac{\vdash u \supset v}{\vdash u \supset \forall x.v}
\]

where \(x\) is not free in \(u\).

**DERIVED RULES AND THEOREMS:**

From R4 we can obtain the derived rule

\[
\text{Instantiation Rule} \quad \text{INST}
\]

\[
\begin{align*}
\vdash w(x) \\
\vdash w(t)
\end{align*}
\]

where \(t\) is any term globally free for \(x\) in \(w\).

**Proof:**

1. \(\vdash w(x)\) \hspace{1cm} \text{given}
2. \(t. \forall x. w(x)\) \hspace{1cm} \text{by } \forall I \text{ (taking } u \text{ to be true)}
3. \(\vdash (\forall x. w(x)) \supset w(t)\) \hspace{1cm} \text{by A12}
4. \(\vdash w(t)\) \hspace{1cm} \text{by 2, 3 and MI'}

The following are the duals of A12 and R4 for the existential quantifier \(\exists\):

**T39.**

\(\vdash w(t) \supset \exists x. w(x)\)

where \(t\) is any term globally free for \(x\) in \(w\).

**Proof:**

1. \(t. (\forall x. \sim w(x)) \supset \sim w(t)\) \hspace{1cm} \text{by A12}
2. \(\vdash (\sim \exists x. w(x)) \supset \sim w(t)\) \hspace{1cm} \text{by A11 and PR}
3. \(\vdash w(t) \supset \exists x. w(x)\) \hspace{1cm} \text{by 1R}

Note again that we need here the additional condition (b) ensuring that the substitution of \(t\) for \(x\) in \(w\) does not create new occurrences of local variables in the scope of a modal operator.
3 Insertion - \exists I
\[
\frac{l \cdot u \subseteq v}{t \cdot 3x. u \subseteq v}
\]
where x is not free in v

Proof:

1. \(\vdash u \subseteq v\) given
2. \(\vdash \neg v \subseteq \neg u\) by PR
3. \(\vdash \neg v \subseteq \forall x. \neg u\) by \(\forall I\)
4. \(\vdash \neg v \subseteq \neg \exists x. u\) by \(\Lambda 11\) and PR
5. \(\vdash \exists x. u \subseteq v\) by PR

\(\forall\) Rules

\[
\begin{array}{c}
\text{(b) (a) } \\
\hline
\vdash u \subseteq v & \vdash u \equiv v \\
\vdash \forall x. u \subseteq \forall x. v & \vdash \forall x. u \equiv \forall x. v
\end{array}
\]

Proof of (a):

1. \(\vdash \forall x. u \subseteq u\) by \(\Lambda 12\)
2. \(l \cdot u \subseteq v\) given
3. \(l \cdot \forall x. u \subseteq v\) by PR
4. \(l \cdot \forall x. u \subseteq \forall x. v\) by \(\forall I\), since \(\forall x. u\) contains no free occurrences of x.

Rule (b) then follows by propositional reasoning.

\(\exists\) Rules

\[
\begin{array}{c}
\text{(a) (b) } \\
\hline
\vdash t \cdot u \subseteq 3x. v & \vdash t \cdot u \equiv v \\
\vdash \exists x. u \subseteq 3x. v & \vdash 3x. u \equiv \exists x. v
\end{array}
\]

Proof of (a):

1. \(l \cdot u \subseteq v\) given
2. \(t \cdot (\neg v) \subseteq (\neg u)\) by PR
3. \(t \cdot (\forall x. \neg v) \subseteq (\forall x. \neg u)\) by \(\forall I\)
4. \(\vdash (4x.2) \subseteq (\neg \exists x. u)\) by \(\Lambda 11\) and PR
Rule (b) then follows by propositional reasoning.

From the axiom Al,

\[ \vdash \Diamond w \equiv \neg o \cdot w, \]

we can clearly deduce the formula

\[ \vdash \neg(w \lor \square \neg w) \equiv \neg(w \lor \neg w) \]

by propositional reasoning (PR). However, we cannot deduce by PR the formula

\[ \square \square \neg w \equiv \neg o \cdot w \]

or

\[ \forall x. \square \neg w \equiv \forall x. \neg o \cdot w. \]

Here, the replacement of \( \square \neg w \) by \( \neg o \cdot w \) is under the scope of the operator \( \square \) and the quantifier \( \forall x \), respectively, and thus cannot be justified by propositional reasoning alone. For this reason we need the following equivalence rule.

<table>
<thead>
<tr>
<th>Equivalence Rule ---- ER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let ( w' ) be the result of replacing an occurrence of a subformula ( v_1 ) in ( w ) by ( v_2 ). Then</td>
</tr>
<tr>
<td>[ \vdash v_1 \equiv v_2 ]</td>
</tr>
<tr>
<td>[ \vdash w \equiv w' ]</td>
</tr>
</tbody>
</table>

**Proof:**

By induction on the structure of \( w \).

Case: \( w \) is \( v_1 \). Then \( w' \) is \( v_2 \) and \( \vdash v_1 \equiv v_2 \) implies \( \vdash w \equiv w' \).

Case: \( w \) is of the form \( \neg u \). We assume that \( \vdash v_1 \equiv v_2 \) implies \( \vdash u \equiv u' \). Then by propositional reasoning \( \vdash \neg u \equiv \neg u' \), i.e., \( \vdash w \equiv w' \).

Case: \( w \) is of the form \( u_1 \lor u_2 \). We assume that if \( \vdash v_1 \equiv v_2 \), then \( \vdash u_1 \equiv u'_1 \) and \( \vdash u_2 \equiv u'_2 \). Then by propositional reasoning \( \vdash (u_1 \lor u_2) \equiv (u'_1 \lor u'_2) \), i.e., \( \vdash w \equiv w' \).

The cases where \( w \) is of forms \( u_1 \land u_2, u_1 \supset u_2 \), etc., are similar.

Case: \( w \) is of the form \( \square \ l u \). We assume that if \( \vdash v_1 \equiv v_2 \), then \( \vdash u \equiv u' \). By the Cl-rule, \( \vdash \square u \equiv \square u' \), i.e., \( \vdash w \equiv w' \).
The cases in which \( w \) is of forms 0 u, 0 u, and \( u_1 \cup u_2 \) are treated similarly, using the 0 \( \Diamond \)-rule, the 0 0-rule, and the UU-rule, respectively.

Case: \( w \) is of the form \( \forall x. u \). We assume that if \( t \vdash v_1 \equiv v_2 \), then \( t \vdash u \equiv u' \). Then by the \( \forall \forall \)-rule, \( t \vdash \forall x. u \equiv \forall x. u' \), i.e., \( t \vdash w \equiv w' \).

The case where \( w \) is of form \( \exists z. u \) is proved similarly by the \( \exists \exists \)-rule.

---

**Deduction Rule -- DED**

\[
\frac{w_1 \vdash w_2}{t \vdash (\Diamond w_1) \supset w_2}
\]

where the \( \forall \) rule (Rule \( R4 \)) is never applied to a free variable of \( w_1 \) in the derivation of \( w_1 \vdash w_2 \).

That is, if under the assumption \( w_1 \) we can derive \( t \vdash w_2 \), where rule \( R4 \) is never applied to a free variable of \( w_1 \), then there exists a proof establishing \( t \vdash (\Diamond w_1) \supset w_2 \). We clearly must also be careful in using any theorem or derived rule such as the \( \forall \forall \) or \( \forall \forall \) rule that was established using the \( \forall \) rule.

The additional \( \Diamond \) operator in the conclusion is obviously necessary since in general \( w_1 \vdash w_2 \) does not imply \( t \vdash \neg w_1 \supset w_2 \). For example, obviously \( w \vdash \neg \Diamond w \) is true (an immediate application of rule \( R3 \): \( t \vdash w \) by assumption and therefore \( t \vdash \Diamond w \) by \( \Diamond \) ); but \( \neg \Diamond \Diamond \neg \) \( w \) is not, a theorem.

**Proof:**

The proof of the temporal Deduction Rule follows the same arguments used in the proof of the classical deduction theorem of Predicate Calculus. By the given \( w_1 \vdash w_2 \), there exists a proof of the form:

\[
\vdash u_1 \\
\vdash u_2 \\
\vdots \\
\vdash u_m
\]

such that \( u_1 = w_1 \) is the hypothesis on which the proof relies, and \( u_m = w_2 \) is the consequence of the proof. We replace each line \( \vdash u_i \) in the proof of \( w_1 \vdash w_2 \) by the line \( \vdash \Diamond u_i \), and show that this transformation preserves soundness. That is

\[
\begin{array}{ll}
given & show \\
\vdash u_1 & \vdash (\Diamond w_1) \supset u_1 \\
\vdash u_2 & \vdash (\Diamond w_1) \supset u_2 \\
\vdots & \vdots \\
\vdots & \vdots \\
\end{array}
\]
where each \( u_i \) is either the assumption \( w_1 \), an axiom, or derived from previous \( u_j \)'s by some rule of inference.

The proof is by a complete induction on \( i \). We assume that for all \( k < i \), \( \vdash (\Box w_1) \supset u_k \), and prove that \( \vdash (\Box w_1) \supset u_i \).

Case: \( u_i \) is an axiom.

1. \( \vdash u_i \) \hspace{1cm} \text{axiom}
2. \( \vdash (\Box w_1) \supset u_i \) \hspace{1cm} \text{by PR}

Note that \( \vdash w' \) implies \( \vdash w \supset w' \) for any \( w \), by propositional reasoning.

Case: \( u_i \) is \( w_1 \).

1. \( \vdash (\Box w_1) \supset w_1 \) \hspace{1cm} \text{by A3}

Case: \( u_i \) is obtained by rule R1, i.e., \( u_i \) is an instance of a tautology.

1. \( \vdash u_i \) \hspace{1cm} \text{by PT}
2. \( \vdash (\Box w_1) \supset u_i \) \hspace{1cm} \text{by PR}

Case: \( u_i \) is obtained by rule R2 (using previous \( \vdash u_k \) and \( \vdash u_k \supset u_i \)).

1. \( \vdash (\Box w_1) \supset u_k \) \hspace{1cm} \text{induction hypothesis}
2. \( \vdash (\Box w_1) \supset (u_k \supset u_i) \) \hspace{1cm} \text{induction hypothesis}
3. \( \vdash (\Box w_1) \supset u_i \) \hspace{1cm} \text{by 1, 2 and PR}

Case: \( u_i \) is obtained by rule R3 (using previous \( \vdash u_k \)), i.e., \( u_i \) is \( \Box u_k \).

1. \( \vdash (\Box w_1) \supset u_k \) \hspace{1cm} \text{induction hypothesis}
2. \( \vdash (\Box \Box w_1) \supset \Box u_k \) \hspace{1cm} \text{by \( \Box \) ICI}
3. \( \vdash (\Box w_1) \supset \Box u_k \) \hspace{1cm} \text{by T3 and PR}
4. \( \vdash (\Box w_1) \supset \Box u_i \) \hspace{1cm} \text{by 2, 3 and PR}
Case: \( u_i \) is obtained by rule R4 (using previous \( \vdash u \supset v \), i.e. \( u_k \), Lo get \( \vdash u \supset \forall x. v \), i.e. \( u_i \), where \( x \) is not free in \( u \)).

By our deduction rule assumption, we know that \( x \) is also not free in \( w_1 \).

1. \( \vdash (\Box w_1) \supset (u \supset v) \)  
   induction hypothesis
2. \( \vdash ((\Box w_1) A u) \supset v \)  
   by PR
3. \( \vdash ((\Box w_1) A u) \supset \forall x. v \)  
   by R4
   (since \( x \) is not free in \( u \) or \( w_1 \))
4. \( \vdash (\Box w_1) \supset (u \supset \forall x. v) \)  
   by PR

A different approach to coping with the application of the CI insertion rule (rule R3) is Lo forbid it altogether. We then get the following restricted deduction rule:

**Restricted Deduction Rule -- RDED**

\[
\vdash w_1 \vdash w_2 \\
\vdash w_1 \supset w_2
\]

where \( \Box I \) (rule R3) is never applied and \( \forall I \) (rule R4) is never applied to a free variable of \( w_1 \) in the derivation of \( w_1 \vdash w_2 \).

Here, we are not allowed to use rule \( \Box I \) or any theorem or derived rule in whose proof \( \Box I \) was used.

The proof of RDED follows exactly that of DED except that the case in which rule R3 is applied does not arise.

**QUANTIFIER THEOREMS:**

**T40.** \( \vdash (4x.w) \equiv (3x. \sim w) \)

**Proof:**

1. \( \vdash (\sim \sim w) \equiv w \)  
   by PT
2. \( \vdash (\forall x. \sim \sim w) \equiv \forall x. w \)  
   by \( \forall \forall \)
3. \( \vdash (\sim \exists x. \sim w) \equiv \exists x. w \)  
   by Al 1 and PR
4. \( \vdash \sim \forall x. w \equiv 3x. \sim w \)  
   by PR
\[ \text{T41. } \vdash \forall x.(w_1 \land w_2) \equiv (\forall x.w_1 \land \forall x.w_2) \]

**Proof:**

1. \( \vdash \forall x.w_1 \quad 3 \ w_1 \)
2. \( \vdash \forall x.w_2 \quad 3 \ w_2 \)
3. \( \vdash (\forall x.w_1 \land \forall x.w_2) \supset (w_1 \land w_2) \)
4. \( \vdash (\forall x.w_1 \land \forall x.w_2) \supset (\forall x.(w_1 \land w_2)) \)
5. \( \vdash (w_1 \land w_2) \supset w_1 \)
6. \( \vdash \forall x.(w_1 \land w_2) \supset \forall x.w_1 \)
7. \( \vdash (w_1 \land w_2) \supset w_2 \)
8. \( \vdash \forall x.(w_1 \land w_2) \supset \forall x.w_2 \)
9. \( \vdash \forall x.(w_1 \land w_2) \supset (\forall x.w_1 \land \forall x.w_2) \)
10. \( \vdash \forall x.(w_1 \land w_2) \equiv (\forall x.w_1 \land \forall x.w_2) \)

\[ \text{T42. } \vdash \exists x.(w_1 \lor w_2) \equiv (\exists x.w_1 \lor \exists x.w_2) \]

**Proof:**

1. \( \vdash \forall x.(\sim w_1 \lor \sim w_2) \equiv (\forall x.\sim w_1 \lor \forall x.\sim w_2) \) by T41
2. \( \vdash \forall x.\sim (w_1 \lor w_2) \equiv (\forall x.\sim w_1 \lor \forall x.\sim w_2) \) by ER
3. \( \vdash \sim \exists x.(w_1 \lor w_2) \equiv (\sim \exists x.w_1 \lor \sim \exists x.w_2) \)
4. \( \vdash \exists x.(w_1 \lor w_2) \equiv (\exists x.w_1 \lor \exists x.w_2) \) by PR

\[ \text{T43. } \vdash \forall x.(w_1 \lor w_2) \equiv [w_1 \lor \forall x.w_2] \text{ where } x \text{ is not free in } w_1. \]

**Proof:**

1. \( \vdash \forall x.(w_1 \lor w_2) \supset [w_1 \lor w_2] \) by A12
2. \( \vdash [\forall x.(w_1 \lor w_2) \lor \sim w_1] \quad 3 \ w_2 \) by PR

40
3. \( \vdash \forall x. (w_1 \lor w_2) \land \sim w_1 \supset \forall x. w_2 \)
   by \( \forall I \),
   since \( x \) is not free in \( \forall x. (w_1 \lor w_2) \land \sim w_1 \)

4. \( \vdash \forall x. (w_1 \lor w_2) \supset [w_1 \lor \forall x. w_2] \)
   by PR

5. \( \vdash w_1 \supset [w_1 \lor w_2] \)
   by PT

6. \( \vdash \forall x. w_2 \supset w_2 \)
   by \( \land I_2 \)

7. \( \vdash \forall x. w_2 \supset [w_1 \lor w_2] \)
   by PR

8. \( \vdash [w_1 \lor \forall x. w_2] \supset [w_1 \lor w_2] \)
   by 5, 7 and PR

9. \( \vdash [w_1 \lor \forall x. w_2] \supset \forall x. (w_1 \lor w_2) \)
   by \( \forall I \),
   since \( x \) is not free in \( w_1 \lor \forall x. w_2 \)

10. \( \vdash \forall x. (w_1 \lor w_2) \equiv [w_1 \lor \forall x. w_2] \)
    by 4, 9 and PR

T44. \( \vdash \exists x. (w_1 \land w_2) \equiv [w_1 \land \exists x. w_2] \) where \( x \) is not free in \( w_1 \)

Proof: By duality on the previous theorem.

The following two theorems show that the 0 operator also commutes with the quantifiers.

T45. \( \vdash (\forall x. 0 \ w) \equiv (0 \ \forall x. w) \)

Proof:

1. \( \vdash (\forall x. 0 \ w) \equiv (0 \ \forall x. w) \)
   by \( \land I_3 \)

2. \( \vdash \forall x. w \supset w \)
   by \( \land I_2 \)

3. \( \vdash (0 \ \forall x. w) \equiv 0 \ w \)
   by \( O O \)

4. \( \vdash (0 \ \forall x. w) \equiv (\forall x. 0 \ w) \)
   by \( \forall I \)

5. \( \vdash (\forall x. 0 \ w) \equiv (0 \ \forall x. w) \)
   by 1, 4 and PR

T46. \( \vdash (3 x. 0 w) \equiv (0 \ \exists x. w) \)

Proof:

1. \( \vdash (\forall x. 0 \sim w) \equiv (0 \ \forall x. \sim w) \)
   by T45

41
2. \( \vdash (\forall x. \sim 0 \ w) \equiv (0 \sim 3x. \ w) \)

3. \( \vdash (\exists x. \ O w) \equiv (\sim 0 \ 3x. \ w) \)

4. \( \vdash (3x. \ O w) \equiv (O \exists x. \ w) \)

The following two theorems show that each temporal operator commutes with the quantifier that has similar character (universal, or existential).

**T47.** \( \vdash (\forall x. \ Box w) \equiv (\Box \forall x. w) \)

**Proof:**

1. \( \vdash \Box w \vdash [w \ A \ Box w] \) by T20 and PR
2. \( \vdash (\forall x. \ Box w) \vdash \forall x.(w \ A \ Box w) \) by VV
3. \( \vdash (\forall x. \ Box w) \vdash [(\forall x. w) \ A (\forall x. \ Box w)] \) by T41 and PR
4. \( \vdash (\forall x. \ Box w) \vdash [(\forall x. \ Box w) (\Box \forall x. \ Box w)] \) by T45 and PR
5. \( \vdash (\forall x. \ Box w) \vdash (\Box \forall x. w) \) by DCI, taking u to be \( \forall x. \ C l \ w \) and v to be \( \forall x. w \)
6. \( \vdash (\forall x. w) \vdash w \) by A12
7. \( \vdash (\Box \forall x. w) \vdash \Box w \) by Box
8. \( \vdash (\Box \forall x. w) \vdash (\forall x. C l \ w) \) by VI
9. \( \vdash (\forall x. \ Box w) \equiv (\Box \forall x. w) \) by 5, 8 and PR

**T48.** \( \vdash (3x. \ O w) \equiv (\Box \exists x. w) \)

**Proof:**

1. \( \vdash (\forall x. \ Box \sim w) \equiv (\Box \forall x. \sim w) \) by T47
2. \( \vdash (\forall x. \ ~ 0 \ w) \equiv (\Box \sim 3x. \ w) \) by A1, A 11 and ER (twice)
3. \( \vdash (\sim \exists x. \ O w) \equiv (\sim \Box \exists x. w) \) by A1, A1 1 and PR
4. \( \vdash (3x. \ O w) \equiv (\Box \exists x. w) \) by PR

Theorem T47 implies the commutativity of \( \forall \) with \( \Box \): Both have a universal character, with one quantifying over individuals and the other quantifying over states. Similarly, theorem T48
implies the commutativity of 3 with 0. The first two theorems (T45 and 1'46) imply the commutativity of V and 3 with 0.

The next two theorems are consistent with the interpretation that the U operator is universal with respect to its first argument and existential with respect to the second.

T49. \( \vdash \forall x. (w_1 U w_2) \equiv (\forall x. w_1) U w_2 \) where x is not free in \( w_2 \)

**Proof:**

1. \( \vdash w_1 U w_2 \supset [w_2 \lor (w_1 A \circ (w_1 U w_2))] \) by A9 and PR
2. \( \vdash \forall x. (w_1 U w_2) \supset \forall x. [w_2 \lor (w_1 A \circ (w_1 U w_2))] \)
3. \( \vdash \forall x. (w_1 U w_2) \supset [w_2 \lor \forall x. (w_1 A \circ (w_1 U w_2))] \)
4. \( \vdash \forall x. (w_1 U w_2) \supset [w_2 \lor (\forall x. w_1 \land \forall x. \circ (w_1 U w_2))] \) by T41 and PR
5. \( \vdash \forall x. (w_1 U w_2) \supset [w_2 \lor (\forall x. w_1 \land \forall x. (w_1 U w_2))] \) by T45 and PR
6. \( \vdash \forall x. (w_1 U w_2) \supset [w_2 \lor \forall x. (w_1 U w_2)] \) by A12, A10 and PR
7. \( \vdash \forall x. (w_1 U w_2) \supset (\forall x. w_1) U w_2 \)

taking w to be \( \forall x. (w_1 U w_2) \), u to be \( \forall x. w_1 \), and v to be \( w_2 \)

8. \( \vdash (\forall x. w_1) U w_1 \) by A12
9. \( \vdash (\forall x. w_1) U w_2 \supset w_1 U w_2 \) by \( UU \)
10. \( \vdash (\forall x. w_1) U w_2 \supset \forall x. (w_1 U w_2) \) by \( \forall \lor \), since x is not free in \( w_2 \)

11. \( \vdash \forall x. (w_1 U w_2) \equiv (\forall x. w_1) U w_2 \)

T50. \( \vdash \exists x. (w_1 U w_2) \equiv w_1 U (\exists x. w_2) \) where x is not free in \( w_1 \)

**Proof:**

1. \( \vdash w_1 U w_2 \supset \lor w_2 \) by A10
2. \( \vdash \exists x. (w_1 U w_2) \supset (\exists x. \lor w_2) \) by 33
3. \( \vdash \exists x. (w_1 U w_2) \supset (\lor \exists x. w_2) \) by T48 and PR
4. \( \vdash w_1 U w_2 \supset [w_2 \lor (w_1 A \circ (w_1 U w_2))] \) by A9 and PR
5. \( \vdash \exists x. (w_1 U w_2) \supset [(\exists x. w_2) \lor \exists x. (w_1 A \circ (w_1 U w_2))] \) by T42, 33 and PR
While operators of similar character, i.e., both universal or both existential, commute to yield equivalent formulas, operators of opposite character usually admit implication in one direction only. Thus we have:

**T51.** \( \vdash \exists x. \Box w \supset \Box 3x.w \)**

**T52.** \( \vdash \Diamond \forall x. w \supset \forall x. \Diamond w \)**

**T53(a).** \( \vdash \exists x.(w_1 \cup w_2) \supset (\exists x.w_1) \cup w_2 \) where \( x \) is not free in \( w_2 \)

(b). \( \vdash w_1 \cup (\forall x.w_2) \supset \forall x.(w_1 \cup w_2) \) where \( x \) is not free in \( w_1 \)

Theorems of similar character are:

**T54(a).** \( \vdash \exists x.(u \cup v) \supset (\exists x.u) \cup (\exists x.v) \)

(b). \( \vdash (\forall x.u) \cup (\forall x.v) \supset \forall x.(u \cup v) \)**

**THE NEXT OPERATOR APPLIED TO TERMS:**

The use of the next operator \( \Diamond \) applied to terms is governed by the axioms:
These axioms are consistent with the evaluation rules that we gave which stated that in order to evaluate an expression \( \xi(t_1, \ldots, t) \), we can evaluate \( \& (O t_1, \ldots, O t_n) \) whether \( \xi \) is a function or a predicate.

5. **EQUAILITY**

Equality is handled by the following axioms:

**AXIOMS:**

A16. **Reflexivity of Equality**

\[1 \cdot t = t \text{ for any term } t\]

A17. **Substitutivity of Equality**

\[1 \cdot (t_1 = t_2) \supset [w(t_1, t_1) \equiv w(t_1, t_2)]\]

where \( t_2 \) is any term globally free for \( t_1 \) in \( w \)

and where \( w \) does not contain temporal operators

A18. \[1 \cdot O(t_1 = t_2) \equiv (O t_1 = O t_2)\]

We use \( w(t_1, t_2) \) to indicate that \( t_2 \) replaces some of the occurrences of \( t_1 \) in \( w \).

* The axiom A18 is a special case of A15 when the predicate \( p \) is the equality predicate.

Recall that a term \( t_2 \) is said to be globally free for \( t_1 \) in \( w \) if substitution of \( t_2 \) for all free occurrences of \( t_1 \) in \( w \): (a) does not create new bound occurrences of (global) variables, (i.e., \( t_2 \) is free for \( t_1 \) in \( w \)), and (b) does not create new occurrences of local variables in the scope of a modal operator.

Note that the classical axiom for substitutivity of equality A17

\[1 \cdot (t_1 = t_2) \supset [w(t_1, t_1) \equiv w(t_1, t_2)]\]

(where \( t_2 \) is free for \( t_1 \) in \( w \)) is not correct if \( w \) contains temporal operators. We could take \( w(t_1, t_2) \) \( \supset [\Box(t_1 = t_1) \equiv \Box(t_1 = t_2)] \) and deduce from A17

\[1 \cdot (t_1 = t_2) \supset [\Box(t_1 = t_1) \equiv \Box(t_1 = t_2)]\]
i.e.,

\[ \vdash (t_1 = t_2) \supset (t_1 = t_2), \]

which is not a valid statement (since \( t_1 = t_2 \) may contain local variables).

**T55. Commutativity of Equality**

\[ \vdash \left( t_1 = t_2 \right) \supset \left( t_2 = t_1 \right) \]

**Proof:**

1. \( \vdash (t_1 = t_2) \supset [(t_1 = t_1) \equiv (t_2 = t_1)] \) \hspace{1cm} \text{by A17}
2. \( \vdash t_1 = t_1 \) \hspace{1cm} \text{by A16}
3. \( \vdash (t_1 = t_2) \supset (t_2 = t_1) \) \hspace{1cm} \text{by 1, 2 and PR}

**T56. Transitivity of Equality**

\[ \vdash [(t_1 = t_2) \land (t_2 = t_3)] \supset (t_1 = t_3) \]

**Proof:**

1. \( \vdash (t_1 = t_2) \supset [(t_1 = t_3) \equiv (t_2 = t_3)] \) \hspace{1cm} \text{by A17}
2. \( \vdash [(t_1 = t_2) \land (t_2 = t_3)] \supset (t_1 = t_3) \) \hspace{1cm} \text{by PR}

**T57. Term Equality**

(a) \[ \vdash \left( t_1 = t_2 \right) \supset [\tau(t_1, t_1) = \tau(t_1, t_2)] \] for any term \( \tau \)

(b) \[ \vdash \left( t_1 = t_2 \right) \supset [\tau(t_1, t_1) = \tau(t_1, t_2)] \] provided \( \tau \) does not contain the next operator.

**Proof of (a):**

By induction on the structure of \( \tau \).

**Case:** \( \tau(t_1, t_1) = t_1 \) and \( \tau(t_1, t_2) = t_1 \). Then

1. \( \vdash t_1 = t_1 \) \hspace{1cm} \text{by A16}
2. \( \vdash \Box(t_1 = t_2) \supset \left[ \tau(t_1, t_1) = \tau(t_1, t_2) \right] \) \hspace{1cm} \text{by PR and definition of } \tau(t_1, t_1) \text{ and } \tau(t_1, t_2)
Case: \( \tau(t_1, t_1) = t_1 \) and \( \tau(t_1, t_2) = t_2 \). Then

1. \( \Box (t_1 = t_2) \supset [\tau(t_1, t_1) = \tau(t_1, t_2)] \)
   by A3

2. \( \Box (t_1 = t_2) \supset [\tau(t_1, t_1) = \tau(t_1, t_2)] \)
   by the definition of \( \tau(t_1, t_1) \) and \( \tau(t_1, t_2) \)

Case: \( \tau(t_1, t_1) = f(\tau_1(t_1, t_1), \ldots, \tau_k(t_1, t_1)) \) and \( \tau(t_1, t_2) = f(\tau_1(t_1, t_2), \ldots, \tau_k(t_1, t_2)) \). Then

1. \( \Box (t_1 = t_2) \supset [\tau_i(t_1, t_1) = \tau_i(t_1, t_2)], \) for \( i = 1, \ldots, k \)
   by the induction assumption.

2. \( \Box \bigwedge_{i=1}^k [\tau_i(t_1, t_1) = \tau_i(t_1, t_2)] \supset
   \bigwedge_{i=1}^k [f(\tau_1(t_1, t_1), \ldots, \tau_k(t_1, t_1)) = f(\tau_1(t_1, t_2), \ldots, \tau_k(t_1, t_2))] \)
   by repeated application of A17 and using T56 for transitivity of equality.

A typical step in this repeated application is:

\[ \Box [\tau_i(t_1, t_1) = \tau_i(t_1, t_2)] \supset [f(\tau_1(t_1, t_1), \ldots, \tau_k(t_1, t_1)) = f(\tau_1(t_1, t_2), \ldots, \tau_k(t_1, t_2))] \]
justified by A17 and the fact that \( \tau_i(t_1, t_2) \) is free for \( \tau_i(t_1, t_1) \) in \( f(...) \) since \( f \) does not contain any temporal operators.

3. \( \Box (t_1 = t_2) \supset [\tau(t_1, t_1) = \tau(t_1, t_2)] \)
   by 1, 2, PR and the definition of \( \tau(t_1, t_1) \) and \( \tau(t_1, t_2) \).

Case: \( \tau(t_1, t_1) = \varnothing \tau'(t_1, t_1) \) and \( \tau(t_1, t_2) = \varnothing \tau'(t_1, t_2) \). Then

1. \( \Box (t_1 = t_2) \supset [\tau'(t_1, t_1) = \tau'(t_1, t_2)] \)
   by the induction hypothesis

2. \( \Box (t_1 = t_2) \supset [\tau'(t_1, t_1) = \tau'(t_1, t_2)] \)
   by \( \Box \)

3. \( \Box (t_1 = t_2) \supset [\varnothing \tau'(t_1, t_1) = \varnothing \tau'(t_1, t_2)] \)
   by A18 and PR

4. \( \Box (t_1 = t_2) \supset [\varnothing \tau'(t_1, t_1) = \varnothing \tau'(t_1, t_2)] \)
   by A7

5. \( \Box (t_1 = t_2) \supset [\tau(t_1, t_1) \supset \tau(t_1, t_2)] \)
   by 4, 2, 3 and PR

6. \( \Box (t_1 = t_2) \supset [\tau(t_1, t_1) \supset \tau(t_1, t_2)] \)
   by the definition of \( \tau(t_1, t_1) \), \( \tau(t_1, t_2) \).

Proof of (b):

1. \( \Box (t_1 = t_2) \supset [(\tau(t_1) = \tau(t_2)) \equiv (\tau(t_2) = \tau(t_2))] \)
   by A17 (no 0 in \( \tau \))

2. \( \Box (t_1 = t_2) \supset [\tau(t_2) = \tau(t_2)] \)
   by A16
The following theorem generalizes A17 to arbitrary formulas.

T58. **Substitutivity of Equality**

\[ \vdash \Box (t_1 = t_2) \supset [w(t_1, t_1) \equiv w(t_1, t_2)] \]

where \( t_2 \) is free for \( t_1 \) in \( w \).

**Proof:**

By induction on the structure of \( w \).

**Case:** \( w \) contains no temporal operators. Then

1. \( \vdash (t_1 = t_2) \supset [w(t_1, t_1) \equiv w(t_1, t_2)] \) \qquad by A17
2. \( \vdash \Box (t_1 = t_2) \supset (t_1 = t_2) \) \quad by A3
3. \( \vdash \Box (t_1 = t_2) \supset [w(t_1, t_1) \equiv w(t_1, t_2)] \) \qquad by MP

**Case:** \( w(t_1, t_2) \) is of the form \( \tau_1(t_1, t_2) = \tau_2(t_1, t_2) \). Then

1. \( \vdash \Box (t_1 = t_2) \supset [\tau_1(t_1, t_1) = \tau_1(t_1, t_2)] \) \qquad by T57
2. \( \vdash \Box \neg (t_1 = t_2) \supset [\tau_2(t_1, t_1) = \tau_2(t_1, t_2)] \) \quad by T57
3. \( \vdash [\tau_1(t_1, t_1) = \tau_1(t_1, t_2)] \supset [(\tau_1(t_1, t_1) = \tau_2(t_1, t_1)) \equiv (\tau_1(t_1, t_2) = \tau_2(t_1, t_1))] \)
   \[ \text{by A17 of the form } (\theta_1 = \theta_2) \supset [(\theta_1 = \tau_2(t_1, t_1)) \equiv (\theta_2 = \tau_2(t_1, t_1))] \]
   with \( \theta_1 = \tau_1(t_1, t_1) \) and \( \theta_2 = \tau_1(t_1, t_2) \)
4. \( \vdash \Box (t_1 = t_2) \supset [(\tau_1(t_1, t_1) = \tau_2(t_1, t_1)) \equiv (\tau_1(t_1, t_2) = \tau_2(t_1, t_1))] \)
   \[ \text{by 1, 3 and PR} \]
5. \( \vdash \Box (t_1 = t_2) \supset [(\tau_1(t_1, t_2) = \tau_2(t_1, t_1)) \equiv (\tau_1(t_1, t_2) = \tau_2(t_1, t_2))] \)
   \[ \text{similarly by A17, using 2} \]
6. \( \vdash \Box (t_1 = t_2) \supset [(\tau_1(t_1, t_1) = \tau_2(t_1, t_1)) \equiv (\tau_1(t_1, t_2) = \tau_2(t_1, t_2))] \)
   \[ \text{by 4, 5 and PR} \]
7. \( \vdash \Box (t_1 = t_2) \supset [w(t_1, t_1) \equiv w(t_1, t_2)] \) \qquad by the definition of \( w(t_1, t_2) \)

**Case:** \( w \) is of the form \( Cl \ u \). Then

1. \( \vdash \Box (t_1 = t_2) \supset [u(t_1, t_1) \equiv u(t_1, t_2)] \) \qquad induction hypothesis
2. \( \vdash \Box (t_1 = t_2) \) \qquad assumption
3. \[ I - u(t_1, t_1) \equiv u(t_1, t_2) \] by MP
4. \[ \Box (t_1) \equiv u(t_1, t_2) \] by \( \Box \Box \)

Thus, \( \Box (t_1) = t_2 \) \[ I - (u(t_1, t_1) \equiv u(t_1, t_2)) \] by DED
5. \[ \Box (t_1 = t_2) \equiv (u(t_1, t_1) \equiv u(t_1, t_2)) \] by T3 and PR

The cases in which \( w \) is of the form \( 0 u, 0 u, \forall x u \) and \( \exists x u \) are treated similarly, using the 0 O-rule, the 0 O-rule, the W-rule and the \( \exists \exists \)-rule, respectively.

Case: \( w \) is of the form \( u \sqcup v \).
1. \[ \Box (t_1 = t_2) \equiv (u(t_1, t_1) \equiv u(t_1, t_2)) \] induction hypothesis
2. \[ \Box (t_1 = t_2) \equiv (v(t_1, t_1) \equiv v(t_1, t_2)) \] induction hypothesis
3. \( (u(t_1, t_1) \equiv u(t_1, t_2)) \) assumption
4. \( I - u(t_1, t_1) \equiv u(t_1, t_2) \) by 1, 3 and MP
5. \( I - v(t_1, t_1) \equiv v(t_1, t_2) \) by 2, 3 and MP
6. \( I - (u(t_1, t_1) \sqcup v(t_1, t_1)) \equiv (u(t_1, t_2) \sqcup v(t_1, t_2)) \) by 4, 5 and ER

Thus, \( \Box (t_1 = t_2) \equiv (u(t_1, t_1) \sqcup v(t_1, t_1)) \equiv (u(t_1, t_2) \sqcup v(t_1, t_2)) \) by DED
7. \[ \Box (t_1 = t_2) \equiv (u(t_1, t_1) \sqcup v(t_1, t_1)) \equiv (u(t_1, t_2) \sqcup v(t_1, t_2)) \] by T3 and PR

6. FRAME AXIOMS AND RULES

In this section we consider the consequences of the partition of the set of all variables into local and global variables. By the semantic definition, global variables are given their value by the global assignment \( a \), and these values do not vary from state to state. Consequently, for a global variable \( u \) it must be universally true that \( u = 0 u \), i.e., the value of \( u \) at any state is identical to its value in the next state (see A19 below). The following axioms are called frame axioms in reference to the “frame axiom” in Hoare’s deductive system for program verification ([ILL]).

Recall that we split the set of our symbols into two subsets: global and local symbols. The logical consequence of this convention is the following frame axiom:

\[ A19. \quad \text{Frame Axiom} \]
\[ \vdash x = \Box x \quad \text{for every global variable } x \]
We can therefore prove by induction on the structure of the term \( t \) and the formula \( w \) the following frame theorems:

**T59.** For a term \( t \) and formula \( w \)

(a) \( \vdash t = Ot \)
where \( t \) is global, i.e., does not contain local symbols

(b) \( \vdash w \equiv \Box w \)
where \( w \) is global, i.e., does not contain local symbols.

(c) \( \vdash w(\bigcirc y_1, \bullet \ldots, \bigcirc y_n) \equiv w(y_1, \bullet \ldots, y_n) \)
where \( y_1, \ldots, y_n \) are all the local variables in \( w \).

We present several frame theorems that facilitate moving global formulas in and out of the scope of temporal operators.

**T60.** \( \vdash \Box (w_1 \lor w_2) \equiv (w_1 \lor \Box w_2) \)
where \( w_1 \) is global, i.e., contains no local symbols.

**Proof:**

1. \( \vdash \Box (w_1 \lor w_2) \lor \Box \sim w_1 \)
by T59b
2. \( \vdash \Box (w_1 \lor w_2) \lor \Box \sim w_1 \lor \Box w_2 \)
by T7 and PR
3. \( \vdash \Box (w_1 \lor w_2) \lor \Box \sim w_1 \lor \Box w_2 \)
by PT
4. \( \vdash \Box (w_1 \lor w_2) \lor \Box \sim w_1 \lor \Box w_2 \)
by 2, 3, \( \Box \) and PR
5. \( \vdash \Box (w_1 \lor w_2) \lor \Box \sim w_1 \lor \Box w_2 \)
by 1, 4 and PR
6. \( \vdash \Box (w_1 \lor w_2) \lor \Box \sim w_1 \lor \Box w_2 \)
by PR
7. \( \vdash \Box (w_1 \lor w_2) \lor \Box \sim w_1 \lor \Box w_2 \)
by T59b
8. \( \vdash \Box (w_1 \lor w_2) \lor \Box \sim w_1 \lor \Box w_2 \)
by PR
9. \( \vdash \Box (w_1 \lor w_2) \lor \Box \sim w_1 \lor \Box w_2 \)
by T9
10. \( \vdash \Box (w_1 \lor w_2) \lor \Box \sim w_1 \lor \Box w_2 \)
by 8, 9 and PR
11. \( \vdash \Box (w_1 \lor w_2) \lor \Box \sim w_1 \lor \Box w_2 \)
by 6, 10 and PR

**T61.** \( \vdash \Diamond (w_1 \land w_2) \equiv (w_1 \land \Diamond w_2) \) where \( w_1 \) is global.

**Proof:** The proof follows from T60 by duality.
A derived frame rule that we will be using is

\[
\text{Frame Rule} \quad \text{FR} \\
\Gamma \vdash u \supset \Diamond v \\
\Gamma \vdash (w \land u) \supset (w \land v)
\]

where \( w \) is global

**Proof:**

1. \( \Gamma \vdash u \supset \Diamond v \) given by PR
2. \( \Gamma \vdash (w \land u) \supset (w \land v) \) by T61 and PR
3. \( \Gamma \vdash (w \land \Diamond v) \supset (w \land v) \) by 2, 3 and PR
C. DOMAIN PART

The next part of the system contains domain axioms that specify the necessary properties of the domain of interest. Thus, to reason about programs manipulating natural numbers, we need the set of Peano Axioms, and to reason about trees we need a set of axioms giving the basic properties of trees and the basic operations defined on them.

7. INDUCTION AXIOMS AND RULES

An essential axiom schema for many domains is the induction axiom schema. This (and all other schemas) should be formulable to admit temporal instances as subformulas. Thus the induction principle for natural numbers can be stated as follows:

\[ \forall n [R(n) \rightarrow R(n + 1)] \rightarrow R(k) \]

for any statement \( R \).

One instance of this axiom, which will be used later, is obtained by taking \( R(n) \) to be \( Q(n) \supset \Diamond \psi \):

T62. Induction Theorem:

\[ \vdash \{ \Box (Q(0) \supset \Diamond \psi) \wedge \forall n [\Box (Q(n) \supset \Diamond \psi) \rightarrow \Box (Q(n + 1) \supset \Diamond \psi)] \} \]

\[ \supset \Box (Q(k) \supset \Diamond \psi) \].

Using this induction theorem we can derive the following useful induction rule:

\[ \Diamond \text{Induction Rule} \rightarrow \Diamond \text{IND} \]

\[ \vdash Q(0) \supset \Diamond \psi \]

\[ \vdash Q(n + 1) \supset [\Diamond \psi \lor \Diamond Q(n)] \]

\[ \vdash Q(k) \supset 0 \psi \]

\( \Diamond \text{IND} \) is useful for proving convergence of a loop: show that \( Q(0) \) guarantees \( 0 \psi \) and that for each \( n \), either \( Q(n + 1) \) implies \( Q(n) \) across the loop or it already establishes \( 0 \psi \) and no further execution is necessary. Then for any \( k \), \( Q(k) \) ensures that \( 0 \psi \) is established.

Proof:

1. \( \vdash Q(0) \supset \Diamond \psi \) given

2. \( \Box (Q(0) \supset \Diamond \psi) \) by \( \Box I \)
While induction over the natural numbers is usually sufficient in order to prove properties of sequential programs, we need induction over more general orderings in order to reason about concurrent programs ([LPS]). Thus we have to formulate a more general induction principle over arbitrary well-founded orderings.

Let $(A, \prec)$ be a partially ordered set. We call the ordering $\prec$ a well-founded ordering if there exists no infinitely decreasing sequence of elements in $A$:

$$\alpha_1 > \alpha_2 > \alpha_3 > \ldots$$

For each well-founded ordering $(A, \prec)$, the following is a valid induction rule:

**R5. Well-Founded Induction Rule — WIND**

$$\vdash \forall \beta (\beta \prec \alpha \vdash w(\beta)) \vdash w(\alpha)$$

This rule should hold for an arbitrary temporal formula $w(\alpha)$ dependent on a global variable $\alpha \in A$, and we adopt it as a primitive inference rule.

To justify the rule semantically we may argue as follows:

Assume that the premise holds but the conclusion is not. Then there must exist a model $M$ and an $\alpha_1$ such that $w(\alpha_1)$ is false under $M$. By the premise there must exist some $\alpha_2$ such that $\alpha_2 \prec \alpha_1$ and $w(\alpha_2)$ is false under $M$. Arguing in a similar way we obtain an infinitely decreasing sequence:

$$\alpha_1 > \alpha_2 > \alpha_3 > \ldots$$

such that for each $i$, $w(\alpha_i)$ is false under $M$. This of course contradicts the well foundedness of $(A, \prec)$.

Note that the induction axiom and rules can be derived from WIND by taking $(A, \prec)$ to be $(N, \prec)$. 

53
In order to use the WIND rule, one has to establish that the ordering < is indeed a well-founded ordering. Several specific orderings are known to be well-founded (such as lexicographic ordering over tuples of integers, multisets, etc.), and may be freely used. However, the general statement that an ordering ‘<’ is well-founded is a second order statement which may require second order reasoning for its establishment.

By substitution of a special form of a temporal formula we can obtain the following induction principle for 0 formulas:

\[
\text{Well-Founded 0 Induction Rule -- OWIND}
\]

\[
\vdash w(\alpha) \models \Diamond (\psi \lor \exists \beta ((\beta < \alpha) \land w(\beta)))
\]

\[
\vdash w(\alpha) \models \Diamond \psi
\]

We show that \(\Diamond\) WIND follows from WIND.

**Proof:**

1. \(\vdash w(\alpha) \models \Diamond (\psi \lor \exists \beta ((\beta < \alpha) \land w(\beta)))\) given
2. \(\vdash w(\alpha) \models (\Diamond \psi \lor \Diamond \exists \beta ((\beta < \alpha) \land w(\beta)))\) by T8 and PR
3. \(\vdash \Box (\exists \beta ((\beta < \alpha) \land w(\beta)) \lor \Diamond \psi) \lor (\Diamond \exists \beta ((\beta < \alpha) \land w(\beta)) \lor \Diamond \psi)\) by T6, T4 and PR
4. \(\vdash \{w(\alpha) \land \Diamond \exists \beta ((\beta < \alpha) \land w(\beta)) \lor \Diamond \psi\} \models \Diamond \psi\) by 2, 3 and PR
5. \(\vdash \Box (\exists \beta ((\beta < \alpha) \land w(\beta)) \lor \Diamond \psi) \lor \Box (w(\alpha) \lor \Diamond \psi)\) by PR
6. \(\vdash (\exists \beta ((\beta < \alpha) \land w(\beta)) \lor \Diamond \psi) \equiv (\exists \beta ((\beta < \alpha) \land w(\beta)) \lor \Diamond \psi)\) by PT
7. \(\vdash (\exists \beta ((\beta < \alpha) \land w(\beta)) \lor \Diamond \psi) \equiv (\exists \beta ((\beta < \alpha) \land \Diamond w(\beta)) \lor \Diamond \psi)\) by T43, PR and PR, since \(\Diamond \psi\) does not depend on \(\beta\)
8. \(\vdash (\forall \beta [\sim (\beta < \alpha) \lor \sim w(\beta)] \lor \Diamond \psi) \equiv \forall \beta [\exists \beta ((\beta < \alpha) \lor \Diamond \psi)]\) by T43, PR and PR, since \(\Diamond \psi\) does not depend on \(\beta\)
9. \(\vdash (\exists \beta ((\beta < \alpha) \land w(\beta)) \lor \Diamond \psi) \equiv \forall \beta [\exists \beta ((\beta < \alpha) \lor \Diamond \psi)]\) by 6, 7, 8 and PR
10. \(\vdash \Box \forall \beta [\exists \beta ((\beta < \alpha) \lor \Diamond \psi)] \lor (w(\beta) \lor \Diamond \psi)] \lor \Box (w(\alpha) \lor \Diamond \psi)\) by 9, 5 and PR
11. \(\vdash \Box \exists \beta ((\beta < \alpha) \lor \Diamond \psi)] \lor (w(\beta) \lor \Diamond \psi)] \lor \Box (w(\alpha) \lor \Diamond \psi)\) by T3, \(\Box \Box \) and PR
12. \(\vdash \Box \exists \beta ((\beta < \alpha) \lor \Diamond \psi)] \lor (w(\beta) \lor \Diamond \psi)] \lor \Box (w(\alpha) \lor \Diamond \psi)\) by T47 and PR
13. \(\vdash \Box \forall \beta [\exists \beta ((\beta < \alpha) \lor \Diamond \psi)] \lor (w(\beta) \lor \Diamond \psi)] \lor \Box (w(\alpha) \lor \Diamond \psi)\) by T60, PR and PR, since \((\beta < a)\) is global
14. \(\vdash \Box (w(\alpha) \lor \Diamond \psi)\) by WIND, taking \(w(\alpha)\) to be \(\Box (w(\alpha) \lor \Diamond \psi)\)
15. \(\vdash w(\alpha) \lor \Diamond \psi\) by A3 and PR
Our proof system must be augmented by additional axioms that reflect the structure of the program under consideration. The additional axioms constrain the state sequences to be exactly the set of execution sequences of the program under study. This relieves us from the need to include program text explicitly in the system; all the necessary information is captured by the additional axioms.

8. PROGRAMS AND COMPUTATIONS

In our model a concurrent program consists of m parallel processes:

\[ P : \ y := g(x); [P_1 \parallel \ldots \parallel P_m]. \]

Each process \( P_i \) is represented as a transition graph with locations (nodes) \( L_i = \{ \ell_0^i, \ldots, \ell_r^i \} \). The edges in the graph are labelled by guarded commands of the form \( c(y) \rightarrow [y := f(y)] \) whose meaning is that if \( c(y) \) is true the edge may be traversed while replacing \( y \) by \( f(y) \).

Let \( \ell, \ell_1, \ell_2, \ldots, \ell_k \in L_i \) be locations in process \( P_i \):

\[
\begin{align*}
\ell & \quad \quad \quad \quad c_1(y) \rightarrow [y := f_1(y)] \\
\ell_1 & \quad \quad \quad \quad \alpha_1 \\
& \quad \quad \quad \quad \quad \vdots \\
& \quad \quad \quad \quad \quad \vdots \\
& \quad \quad \quad \quad \quad \vdots \\
& \quad \quad \quad \quad c_k(y) \rightarrow [y := f_k(y)] \\
\ell_k & \quad \quad \quad \quad \alpha_k
\end{align*}
\]

The variables \( y = (y_1, \ldots, y_n) \) are shared by all processes. We define \( E_\ell(y) = c_1(y) \lor \ldots \lor c_k(y) \) to be the exit condition at node \( \ell \). We do not require that the conditions \( c_i \) be either exclusive or exhaustive.

The advantage of the transition graph representation is that programs are represented in a uniform way and that we have only to deal with one type of instruction. We show first that programs represented in a linear text form can easily be translated into graph form.

Assume that a linear text program allows the following types of instructions:

Assignment: \( \overline{y} := f(y) \)
Conditional Branch: if \( p(y) \) then go to \( \ell_1 \) else go to \( \ell_2 \)

Halt: halt

Waiting loop: loop until \( p(y) \)

loop while \( p(y) \)

and the semaphore instructions

Request: request(y)

Release: release(y)

A linear text program for each of the processes has the following form:

\[
\begin{align*}
\ell_0 &: I_0 \\
\ell_1 &: I_1 \\
& \vdots \\
\ell_t &: \text{halt or go to } \ell_j
\end{align*}
\]

where \( \ell_0, \ell_1, \ldots, \ell_t \) are labels and \( I_0, I_1, \ldots \) are instructions from the list above.

The graph representation of such a program for process \( P \) will be a labelled graph with \( L_i = \{ \ell_0, \ldots, \ell_t \} \) as the set of nodes. For each instruction \( I \) at label \( \ell \in L_i \) we construct edges as follows:

- for the instruction
  \[ \ell : \bar{y} := f(y) \]
  \[ \ell' : \]
  construct
  \[
  \begin{array}{c}
  \ell \\
  \text{true} \to [\bar{y} := f(y)] \\
  \ell'
  \end{array}
  \]

- for the instruction
  \[ \ell : \text{if } p(y) \text{ then go to } \ell' \text{ else go to } \ell'' \]
  \[ \ell' : \]
  construct

\[
\begin{array}{c}
\ell \\
p(y) \to [] \\
\ell' \\
\sim p(y) \to [] \\
\ell''
\end{array}
\]
• for the instruction
  \[ \ell : \text{ if } p(y) \text{ then go to } \ell' \]
  \[ \ell' : \]
  construct

\[ p(y) \rightarrow [] \]
\[ \ell' \]
\[ \sim p(y) \rightarrow [] \]
\[ \ell'' \]

• for the instruction
  \[ \ell : \text{ if } p(y) \text{ then } y := f(y) \]
  \[ \ell' : \]
  construct

\[ p(y) \rightarrow [y := f(y)] \]
\[ \ell' \]
\[ \sim p(y) \rightarrow [] \]

• for the instruction
  \[ \ell : \text{ loop until } p(y) \]
  \[ \ell' : \]
  construct

\[ \sim p(y) \rightarrow [] \]
\[ p(y) \rightarrow [] \]
\[ \ell' \]

• for the instruction
  \[ \ell : \text{ loop while } p(y) \]
  \[ \ell' : \]
  construct

\[ p(y) \rightarrow [] \]
\[ \sim p(y) \rightarrow [] \]
\[ \ell' \]

• for the instruction
\( \ell : \text{request}(y) \)

\( \ell' : \)

\[
\begin{array}{c}
\ell \quad y > 0 \rightarrow [y := y - 1] \quad \ell'
\end{array}
\]

\[\text{Construct}\]

\[\text{for the instruction} \]

\( \ell : \text{release}(y) \)

\( \ell' : \)

\[
\begin{array}{c}
\ell \quad \text{true} \rightarrow [y := y + 1] \quad \ell'
\end{array}
\]

For halt at label \( \ell \) we construct no edges out of \( \ell \).

The actual translation into graph form need not be carried out explicitly. Rather, the general axiomatic description of transition diagrams can be easily translated to axioms for each of the types of instructions in the linear text form.

A state of the program \( P \) is a tuple of the form \( s = (\vec{e}; \eta) \) with \( \vec{e} \in E_1 \times \ldots \times E_m \) and \( \eta \in D^n \), where \( D \) is the domain over which the program variables \( y_1, \ldots, y_n \) range. The vector \( \vec{e} = (e_1', \ldots, e_m') \) is the set of current locations which are next to be executed in each of the processes. The vector \( \eta \) is the set of current values assumed by the program variables \( \eta \) at state \( s \).

Let \( s = (\vec{e}_1, \ldots, \vec{e}_i, \ldots, \vec{e}_m; \eta) \) be a state. We say that process \( P_i \) is enabled on \( s \) if \( E_{P_i}(\eta) = \text{true} \). This implies that if we let \( P_i \) run at this point, there is at least one condition \( c_j \) among the edges departing from \( \vec{e}_i \) that is true. Otherwise, we say that \( P_i \) is disabled on \( s \). An example of a disabled process is the case where \( \vec{e}_i \) labels an instruction \( \text{request}(y) \) and \( y = 0 \). Another example is that of \( \vec{e}_i \) labeling a halt statement. A state is defined to be terminal if no \( P_i \) is enabled on it.

Given a program \( P \) we define the notion of a computation step of \( P \).

Let \( s = (\vec{e}_1, \ldots, \vec{e}_i, \ldots, \vec{e}_m; \eta) \) and \( \tilde{s} = (\vec{e}_1, \ldots, \vec{e}_m; \bar{\eta}) \) be two states of \( P \). Let \( \tau \) be a transition in \( P_i \) of the form:

\[
\begin{array}{c}
\vec{e}_i \quad c(\vec{y}) \rightarrow [y := f(\vec{y})] \quad \vec{e}_i'
\end{array}
\]

such that \( c(\eta) = \text{true} \), \( \bar{\eta} = f(\eta) \), and for every \( j \neq i \), \( \vec{e}_j = \vec{e}_j' \). Then we say that \( \tilde{s} \) can be obtained from \( s \) by a Pi-step (a single computation step), and write

\[
\begin{array}{c}
s \xrightarrow{P_i} \tilde{s}
\end{array}
\]

An initialized admissible computation of a program \( P \) for an input \( \vec{x} = \vec{\xi} \) is a labelled maximal sequence of states of \( P \):

\[
\sigma : s_0 \xrightarrow{P_{i_1}} s_1 \xrightarrow{P_{i_2}} s_2 \xrightarrow{P_{i_3}} s_3 \xrightarrow{\ldots}
\]

58
which satisfies the following three conditions. (The sequence \( \sigma \) is considered \textit{maximal} if it cannot be extended, i.e., it is either infinite or ends with a state \( s_k \) which is terminal.)

\section*{A. Initialization:}

The first state \( s_0 \) has the form:

\[
s_0 = \langle \ell_0; g(\xi) \rangle
\]

where \( \ell_0 = (\ell_0^1, \ldots, \ell_0^m) \) is the vector of initial locations. The values \( g(\xi) \) are the initial values assigned to the \( y \) variables for the input \( \xi \).

\section*{B. State to State Sequencing:}

Every step in the computation \( s \xrightarrow{P_i} \tilde{s} \), is justified by a Pi-step.

\section*{C. Fairness:}

Every \( P_i \) which is enabled on infinitely many states in \( \sigma \) must be activated infinitely many times in \( \sigma \), i.e., there must be an infinite number of \( P_i \)-steps in \( \sigma \).

We define an \textit{admissible computation} of \( P \) for input \( \xi \) to be either an initialized admissible computation or a suffix of an initialized admissible computation.

Thus the class of admissible computations is closed under the operation of taking the suffix. This is needed in order to ensure soundness of the inference rule \( \square I \) (123). We denote the class of all \( \xi \)-admissible computations of a program \( P \) by \( A(P, \xi) \).

An admissible computation is said to be \textit{convergent} if it is finite:

\[
\sigma : s_0 \xrightarrow{P_{i_1}} s_1 \xrightarrow{P_{i_2}} \ldots \xrightarrow{P_{i_f}} s_f.
\]

If the terminal state \( s_f \) in a convergent computation is of the form \( s_f = \langle \ell^1, \ldots, \ell^m; \eta \rangle \), where each \( \ell^i \) labels a halt instruction, we say that the \textit{computation has terminated}. Otherwise, we say that the \textit{computation has blocked} or is \textit{deadlocked}.

In order to describe properties of states we introduce a vector of \textit{locution variables} \( \pi = (\pi_1, \ldots, \pi_m) \). Each \( \pi_i \) ranges over \( L_i \), and assumes the location value \( \ell^i \) in a state

\[
s = \langle \ell^1, \ldots, \ell^i, \ldots, \ell^m; \eta \rangle.
\]

Thus we may describe a state \( s = \langle \ell; \eta \rangle \) by saying that in this state \( \pi = \ell \) and \( y = \eta \).

A \textit{state formula} \( Q = Q(\pi; \eta) \) is any formula which contains no temporal operators. It is built up of terms and predicates over the location and program variables \( (\pi; \eta) \) and may also refer to global variables.

We frequently abbreviate the statement \( \pi_i = \ell \) to \( \ell \). Since the \( L_i \)'s are disjoint, there is no difficulty in identifying the particular \( \pi_i \) which assumes the value \( \ell \).
Let us consider a program $P$ over a domain $D$ with fixed interpretation $I$ for all the predicate, function and individual constant symbols. A model $M$ is said to be admissible for $P$ if it has the form:

$$M = (I, \alpha, \hat{\sigma})$$

where $\alpha$ and $\hat{\sigma}$ satisfy the following condition:

There exists an $\alpha[\bar{z}]$-admissible computation $\sigma \in A(P, \alpha[\bar{z}])$ such that either

$$\sigma \text{ is infinite: } \sigma = s_0 \xrightarrow{P_{i_1}} s_1 \xrightarrow{P_{i_2}} s_2 \rightarrow \cdots$$

and

$$\hat{\sigma} = s_0, s_1, s_2, \ldots$$

or

$$\sigma \text{ is finite: } \sigma = s_0 \xrightarrow{P_{i_1}} s_1 \xrightarrow{P_{i_2}} s_2 \rightarrow \cdots \rightarrow s_f$$

and then

$$\hat{\sigma} = s_0, s_1, s_2, \ldots, s_f, s_f, \ldots$$

Thus we force $\hat{\sigma}$ to be always infinite by indefinitely repeating the last state of $\sigma$ if it is finite. This corresponds to our intuition that while the computation may have terminated, time still marches on, but no further change in the program will ever occur.

Let us denote the class of all admissible models for a program $P$ by $C(P)$. Note that this class, differently from $A(P, \bar{\xi})$, contains computations corresponding to different inputs.

We define the state formula stating that a process $P_i$ is enabled as follows:

$$Enabled(P_i; \bar{x}; \bar{y}) = \bigwedge_{\ell \in L_i} \left( [\pi_i = \ell] \supset E_\ell(\bar{y}) \right).$$

For the complete program $P$ we defined

$$Enabled(P; \bar{x}; \bar{y}) = \bigvee_{i=1}^{m} Enabled(P_i; \bar{x}; \bar{y}).$$

Thus a state $s = (\bar{t}; \bar{\eta})$ is terminal iff

$$Enabled(P; \bar{t}; \bar{\eta}) = \text{false}$$

and we may define

$$\text{Terminal} (\bar{x}; \bar{y}) \equiv \neg Enabled(P; \bar{x}; \bar{y}).$$

60
Let the following be a transition \( \tau \) in process \( P_i \):

\[ e(\bar{y}) \rightarrow [\bar{y} := f(\bar{y})] \]

We define the transformation associated with the transition \( \tau \) by:

\[ r_\tau(\bar{x}; \bar{y}) = \left( \bar{x}[\bar{y} / \bar{y}]; f(\bar{y}) \right) \]

The transformation is obtained by replacing the current value \( \ell \) of \( \pi_i \) by \( \ell' \) and the values of \( \bar{y} \) by \( f(\bar{y}) \).

Let \( \varphi(\bar{x}; \bar{y}) \) and \( \psi(\bar{x}; \bar{y}) \) be two state formulas. We say:

- **The transition** \( \tau \) **leads from** \( \varphi \) **to** \( \psi \) if the following implication is valid:
  \[ [\varphi(\bar{x}; \bar{y}) \land at \ell \land e(\bar{y})] \supset \psi \left( r_\tau(\bar{x}; \bar{y}) \right) \] .

- **The process** \( P_i \) **leads from** \( \varphi \) **to** \( \psi \) if every transition \( \tau \) in \( P_i \) leads from \( \varphi \) to \( \psi \).

- **The program** \( P \) **leads from** \( \varphi \) **to** \( \psi \) if every \( P_i \) leads from \( \varphi \) to \( \psi \).

We are ready now to give a temporal axiomatization for the notion of computation under the program \( P \).

### 9. AXIOMS AND RULES FOR CONCURRENT PROGRAMS

The first axiom states that the location variable \( \pi_i \) may only assume values in \( L_i \).

**A21. Location Axiom -- LOC**

\[ \vdash \pi_i \in L_i \quad \text{for} \quad i = 1, \ldots, m. \]

This is an abbreviation for:

\[ \vdash (\pi_i = \ell_0) \lor (\pi_i = \ell_1) \lor \ldots \lor (\pi_i = \ell_m). \]

Since all the locations are disjoint, it also follows from the equality axioms that \( \pi_i \) may be equal to at most one \( \ell_j \) at a time.

For each of the three requirements defining an admissible computation we have a corresponding inference rule scheme:

**R6. Initialization -- INIT**

For an arbitrary temporal formula \( w \):

\[ \vdash [at \ell_0 \land \bar{y} = y(z)] \supset \square w \]

\[ \vdash \Box w \]
For let us assume that the premise to this rule holds. This implies that Cl \( w \) is true for all initialized computations. By the semantic definition of \( \square \), this implies that \( w \) is true for every suffix of an initialized computation, i.e., for every admissible computation. Thus, \( w \) is C(P)-valid, and by generalization(\( \Box \)) so is \( \square \) \( \Box w \).

R7. Transition -- TRNS

Let \( \varphi(\vec{x}; \vec{y}) \) and \( \psi(\vec{x}; \vec{y}) \) be two state formulas.

\[
\begin{align*}
\vdash P & \text{ leads from } \varphi \text{ to } \psi \\
\vdash [\varphi(\vec{x}; \vec{y}) \land \text{Terminal}(\vec{x}; \vec{y})] & \supset \psi(\vec{x}; \vec{y}) \\
\vdash \varphi & \supset \Box \psi
\end{align*}
\]

Indeed let \( s \) be a state in the sequence \( \sigma \) corresponding to an admissible computation \( \sigma \), and let \( s' \) be its successor in \( \sigma \). Assume that \( \varphi(s) \) is true. There are two cases to be considered. In the first case, \( s' \) is derived from \( s \) by a \( P_i \)-step for some \( i = 1, \ldots, m \). But then, by the first premise, \( P_i \) leads from \( \varphi \) to \( \psi \) and therefore \( \psi \) must be true for \( s' \). In the other case, \( s \) is terminal and \( s' = s \) the repetition of the terminal state of a finite computation. But then \( s \) is terminal and satisfies the antecedent of the second premise, leading to \( \psi(s) = \psi(s') = \text{true} \). Hence, in both cases \( \psi(s') \) must hold and the conclusion of the rule follows.

Note that the first premise to this rule requires establishing many conditions involving the individual transitions of each of the processes. However, by examining the definitions of “leading from \( \varphi \) to \( \psi \)” we see that they are all expressible as classical statements involving no temporal operators. Therefore this premise should be provable from the domain axioms plus the usual predicate calculus proof system. The second premise is also classical, and ensures the consequence after the sequence has reached a terminal state.

R8. Fairness -- FAIR

Let \( \varphi(\vec{x}; \vec{y}) \) and \( \psi(\vec{x}; \vec{y}) \) be two state formulas and \( P_k \) be one of the processes.

\[
\begin{align*}
A. & \quad \text{I- } P \text{ leads from } \varphi \text{ to } \varphi \lor \psi \\
B. & \quad \text{t- } P_k \text{ leads from } \varphi \text{ to } \psi \\
\vdash [\varphi \land \Box \Diamond \text{Enabled}(P_k)] & \supset \varphi \lor \psi
\end{align*}
\]

To give a semantic justification of this rule, consider a computation such that \( \varphi \) is true initially. By A, \( \varphi \) will hold until \( \psi \) is realized, if ever. By B, once \( P_k \) will be activated in a state satisfying \( \varphi \) it will achieve \( \psi \) in one step. Consider now a sequence \( \sigma \) such that \( \varphi \land \Box \Diamond \text{Enabled}(P_k) \) is true on \( \sigma \). This means that \( \varphi \) is initially true and \( P_k \) is enabled infinitely many times in \( \sigma \). By fairness, \( P_k \) will eventually be activated, which, if \( \psi \) has not been realized before, will achieve \( \psi \) in one step.

Since \( (\varphi \lor \psi) \supset 0 \psi \), we often use the FAIR rule in order to derive the consequence

\[ [\varphi \land \Box \Diamond \text{Enabled}(P_k)] \supset 0 \psi. \]

There are several derived rules that can be obtained from the above axiomatization.
Invariance Rule -- INV
\[ \vdash P \text{ leads from } \varphi \text{ to } \varphi \]
\[ \vdash \varphi \subset \Box \varphi \]

Proof:

1. \( \vdash P \text{ leads from } \varphi \text{ to } \varphi \) given
2. \( \vdash [\varphi \land \text{Terminal}] \subset \varphi \) by PT
3. \( \vdash \varphi \subset \Box \varphi \) by TRNS
4. \( \vdash \varphi \subset \Box \varphi \) by CI

Initialized Invariance Rule -- IINV

Let \( \varphi \) be a state formula
\[ t- [at \bar{t}_0 A \; \bar{y} = g(F)] \subset \varphi \]
\[ \vdash P \text{ leads from } \varphi \text{ to } \varphi \]
\[ \vdash \Box \varphi \]

Proof:

1. \( \vdash [at \bar{t}_0 A \; \bar{y} = g(F)] \subset \varphi \) given
2. \( t- P \text{ leads from } \varphi \text{ to } \varphi \) given
3. \( \vdash \varphi \subset \Box \varphi \) by 2 and IINV
4. \( \vdash [at \bar{t}_0 A \; \bar{y} = g(F)] \subset \Box \varphi \) by 1, 3 and PR
5. \( \vdash \Box \varphi \) by INIT

The IINV rule is the rule most often used in order to establish invariance properties of programs.

Unless Establishment Rule -- UER

Let \( \varphi \) be a state formula
\[ \vdash P \text{ leads from } \varphi \text{ to } \varphi \lor \psi \]
\[ \vdash \varphi \subset (\varphi \lor \psi) \]

Proof:

1. \( \vdash P \text{ leads from } \varphi \lor \psi \) given
2. \( \vdash \varphi \supset (\varphi \lor \psi) \)  
3. \( \vdash [\varphi \land \text{Terminal}] \supset (\varphi \lor \psi) \)  
4. \( \vdash \varphi \supset \Box(\varphi \lor \psi) \)  
5. \( \vdash \varphi \supset (\varphi \lor \psi) \)

The following rule is a consequence of the FAIR rule.

**Eventuality Rule—— EVNT**

Let \( \varphi(\overline{x}; \overline{y}) \) and \( \psi(\overline{x}; \overline{y}) \) be two state formulas and \( P_k \) one of the processes.

\[ \begin{align*} 
A. & \quad I-P \text{ leads from } \varphi \text{ to } \varphi \lor \psi \\
B. & \quad I-P_k \text{ leads from } \varphi \text{ to } \psi \\
C. & \quad \vdash \varphi \supset (\psi \lor \text{Enabled}(P_k)) \\
\end{align*} \]

\[ \vdash \varphi \supset \varphi \lor \psi \]

**Proof:**

1. \( I-P \text{ leads from } \varphi \text{ to } \varphi \lor \psi \) given
2. \( t-P_k \text{ leads from } \varphi \text{ to } \psi \) given
3. \( t- \varphi \supset \Box (\psi \lor \text{Enabled}(P_k)) \) given
4. \( t-[\varphi A \Box \Box \text{Enabled}(P_k)] \supset \varphi \lor \psi \) by 1, 2 and FAIR
5. \( \vdash [\varphi A \Box \Box \text{Enabled}(P_k)] \supset \varphi \lor \psi \) by 1 and CINV
6. \( t-[\varphi A \Box \Box \text{Enabled}(P_k)] \supset \varphi \lor \psi \) by 3, T8, A1 and PR
7. \( \Diamond \Box \Box (\varphi \Box \Box \text{Enabled}(P_k)) \) by \( \Box \Box \)
8. \( t-[\Box \varphi A \Box \Box \text{Enabled}(P_k)] \supset \Box \Box \psi \) by T3, T7 and PR
9. \( I-[\Box \varphi A \Box \Box \text{Enabled}(P_k)] \supset \Box \psi \) by A1 and PR
10. \( I- \Box \varphi \supset \Box \psi \) by 4, 9, A3, A10 and PR
11. \( I- \Box \varphi \supset \varphi \lor \psi \) by 10, T24 and PR
12. \( I- \varphi \supset \varphi \lor \psi \) by 5, 11 and PR

In contrast with earlier rules, premise C of EVNT is not purely classical since it contains the temporal operator \( \Box \). Since C has a form similar to the conclusion of the EVNT rule, it is to be expected that its derivation will require once more the application of the EVNT rule. This seems
to imply circular reasoning. *However*, note that at each nested application of the EVNT rule, another $P_k$ is taken out of consideration. This is because in trying to establish $\emptyset \text{ Enabled}(P_k)$ we need not consider any $P_k$-steps at all, since when they are possible, $P_k$ is already enabled.

A useful special case of $C$ that frequently suffices for the application of the EVNT rule is:

$$C' : \vdash \varphi \models [\psi \lor \text{ Enabled}(P_k)].$$

Note that the EVNT rule can also be used to establish properties of the form

$$\varphi \models \Diamond \psi,$$

since $\varphi \cup \psi \models \Diamond \psi$.

The EVNT rule is the one most often used in order to establish both eventuality (liveness) properties and precedence properties.
E. EXAMPLES

In this section we present several examples of proofs of properties of programs using the proof system described above.

10. EXAMPLE 1: DISTRIBUTED GCD

Let us consider the following example of a program computing the greatest common divisor of two positive integers in a distributed manner.

\[
(y_1, y_2) := (x_1, x_2)
\]

\[
\ell_0 : \text{if } y_1 > y_2 \text{ then } y_1 := y_1 - y_2 \quad m_0 : \text{if } y_1 < y_2 \text{ then } y_2 := y_2 - y_1
\]

\[
\ell_1 : \text{if } y_1 \neq y_2 \text{ then go to } \ell_0 \quad m_1 : \text{if } y_1 \neq y_2 \text{ then go to } m_0
\]

\[
\ell_2 : \text{halt} \quad m_2 : \text{halt}
\]

We wish to prove total correctness for this program, i.e.,

**Theorem:**

\[
\vdash [\text{at}(\ell_0, m_0) \land (y_1, y_2) = (x_1, x_2)] \supset \text{at}(\ell_2, m_2) \land y_1 = \text{gcd}(x_1, x_2)]
\]

We will split the proof into two parts, proving separately invariance and termination.

**Lemma A:**

\[
\vdash \Box[gcd(y_1, y_2) = gcd(x_1, x_2)]
\]

**Proof of Lemma A:**

Let us denote \(gcd(y_1, y_2) = gcd(x_1, x_2)\) by \(\bar{\phi}(x_1, x_2, y_1, y_2)\).

It is easy to check that every transition in \(P\) leads from \(\bar{\phi}\) to \(\bar{\phi}\). Also

\[
\vdash [(y_1, y_2) = (x_1, x_2)] \supset \bar{\phi}(x_1, x_2, y_1, y_2).
\]
Thus we have the two premises to the IINV rule, which yields the desired result.

Lemma B:

\[ \vdash [\text{at}_{l_0,1} \land \text{atm}_{0,1} \land (y_1, y_2) > 0 \land (y_1 + y_2) \leq n + 1] \land y_1 \neq y_2 \]

\[ \supset \lozenge [\text{at}_{l_0,1} \land \text{atm}_{0,1} \land (y_1, y_2) > 0 \land (y_1 + y_2) \leq n] \]

Here we use \text{at}_{l_0,1} as an abbreviation for \text{at}_{l_0} \lor \text{at}_{l_1}, \text{atm}_{0,1} \lor \text{atm}_1 \lor (y_1, y_2) > 0 \lor (y_1 > 0) \land (y_2 > 0).

Proof of Lemma B:

Let us define

\[ \varphi(y_1, y_2, n) : \text{at}_{l_0,1} \land \text{atm}_{0,1} \land (y_1, y_2) > 0 \land (y_1 + y_2) \leq n. \]

Thus we have to prove:

\[ \vdash [\varphi(y_1, y_2, n + 1) \land (y_1 \neq y_2)] \supset \lozenge \varphi(y_1, y_2, n). \]

We will split the proof into two cases:

B1. \[ \vdash [\varphi(y_1, y_2, n + 1) \land (y_1 > y_2)] \supset \lozenge \varphi(y_1, y_2, n) \]

B2. \[ \vdash [\varphi(y_1, y_2, n + 1) \land (y_1 < y_2)] \supset \lozenge \varphi(y_1, y_2, n) \]

The lemma obviously follows from these two statements.

To prove B1 we first observe that by PR:

1. \[ \vdash \varphi(y_1, y_2, n + 1) \supset (\text{at}_{l_0} \lor \text{at}_{l_1}) \]

Consider therefore first the case that \( P_1 \) is at \( l_0 \). We take

\[ \varphi' : \varphi(y_1, y_2, n + 1) \supset (y_1 > y_2) \land \text{at}_{l_0} \]

\[ \psi' : \varphi(y_1, y_2, n). \]

We claim that \( \varphi' \) and \( \psi' \) satisfy the premises of EVNT with \( P_k = P_1 \).

To see this, consider requirement A of EVNT that states that every transition in \( P \) leads from \( \varphi' \) to \( \varphi' \lor \psi' \).

Consider transitions in \( P_2 \). The only relevant ones are \( m_0 \to m_1 \) and transitions leading out of \( m_1 \). The transition \( m_1 \to m_1 \) under \( y_1 > y_2 \) leaves \( \varphi' \) invariant. Again, under \( y_1 > y_2 \) the only transition out of \( m_1 \) goes to \( m_0 \) leaving \( \varphi' \) invariant.
The only transition enabled in $P_1$ is $\ell_0 \rightarrow \ell_1$ which replaces $(y_1, y_2)$ by $(y_1 - y_2, y_2)$. If $y_1 + y_2 \leq n + 1$ and $y_1 > 0$, $y_2 > 0$ then certainly $(y_1 - y_2) + y_2 \leq n$ and $(y_1 - y_2) > 0, y_2 > 0$. Thus $\ell_0 \rightarrow \ell_1$ leads from $\varphi'$ to $\psi'$. This also establishes requirement $B$ with $P_k = P_1$.

Since $E_{\ell_0} = \text{true}$, condition $C$ is trivially fulfilled. Consequently we conclude by the EVNT rule that $\vdash \varphi' \supset \psi'$, i.e.,

2. $\vdash [\varphi(y_1, y_2, n + 1) \land (y_1 > y_2) \land at \ell_0] \supset \varphi(y_1, y_2, n)$.

Consider next the case where $P_1$ is at $\ell_1$. By taking

$\varphi'' = \varphi(y_1, y_2, n + 1) \land (y_1 > y_2) \land at \ell_1$

$\psi'' = \varphi': \varphi(y_1, y_2, n + 1) \land (y_1 > y_2) \land at \ell_0$.

We can show that the premises of the EVNT rule are satisfied with respect to $\varphi''$, $\psi''$. Consequently we have $\vdash \varphi'' \supset \psi''$, i.e.,

3. $\vdash [\varphi(y_1, y_2, n + 1) \land (y_1 > y_2) \land at \ell_1] \supset [\varphi(y_1, y_2, n + 1) \land (y_1 > y_2) \land at \ell_0]$

4. $\vdash [\varphi(y_1, y_2, n + 1) \land (y_1 > y_2) \land at \ell_1] \supset \varphi(y_1, y_2, n)$ by 2, 3 and OC

5. $\vdash [\varphi(y_1, y_2, n + 1) \land (y_1 > y_2)] \supset \varphi(y_1, y_2, n)$ by 1, 2, 4 and PR

This establishes B1.

By a symmetric argument we can establish B2. By propositional reasoning B1 and 132 lead to Lemma B.

\[ \blacksquare \]

**Proof of theorem:**

We will now proceed with the proof of the main theorem.

6. $\vdash [\varphi(y_1, y_2, n + 1) \land (y_1 \neq y_2)] \supset \varphi(y_1, y_2, n)$ Lemma B

7. $\vdash \varphi(y_1, y_2, n + 1) \supset [(y_1 = y_2) \lor \varphi(y_1, y_2, n)]$ by PR

8. $\vdash \varphi(y_1, y_2, n + 1) \supset [\varphi(y_1 = y_2) \lor \varphi(y_1, y_2, n)]$ by T1 and PR

9. $\vdash \lnot \varphi(y_1, y_2, 0)$ by PR, using the domain property that the conjunction $(y_1 > 0) \land (y_2 > 0) \land (y_1 + y_2 \leq 0)$ is impossible

10. $\vdash \varphi(y_1, y_2, 0) \supset \varphi(y_1 = y_2)$ by PR

11. $\vdash \varphi(y_1, y_2, n) \supset \varphi(y_1 = y_2)$ by 8, 10 and OIND

12. $\vdash \exists n. \varphi(y_1, y_2, n) \supset \varphi(y_1 = y_2)$ by EI

13. $\vdash [at(\ell_0, m_0) \land (y_1, y_2) = (x_1, x_2) > 0] \exists n. \varphi(y_1, y_2, n)$
By considering the different locations of $P_1$ and $P_2$ under the assumption that $y_1 = y_2$ it is easy (though long if carried out in full detail) to establish

14. $\vdash (y_1 = y_2) \supset \lozenge [at(\ell_2, m_2) \land (y_1 = y_2)]$.

By combining 12, 13 and 14 using OC we obtain:

15. $\vdash [at(\ell_0, m_0) \land (y_1, y_2) = (x_1, x_2) > 0] \supset \lozenge [at(\ell_2, m_2) \land (y_1 = y_2)]$.

Together with lemma A and T10 this gives

16. $\vdash [at(\ell_0, m_0) \land (y_1, y_2) = (x_1, x_2) > 0] \supset \lozenge [at(\ell_2, m_2) \land y_1 = gcd(x_1, x_2)]$

since $(y_1 = y_2) \supset y_1 = gcd(y_1, y_2)$

Note that theorem T10 enables us to infer from a previously established invariant $I:\ 3 \varphi$ and an implication $\vdash w_1 \supset w_2$ the implication $\vdash w_1 \supset (w_2 \land \varphi)$.

II. EXAMPLE 2: SEMAPHORES

For our next example we will present a very simple program with semaphores:

<table>
<thead>
<tr>
<th>$\gamma' := 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell_0 : request(y)$</td>
</tr>
<tr>
<td>$\ell_1 : release(y)$</td>
</tr>
<tr>
<td>$\ell_2 : go \ to \ \ell_0$</td>
</tr>
</tbody>
</table>

This example models a solution to the mutual exclusion problem using semaphores.

There are two properties that we wish to prove for this program. The first is that of mutual exclusion, namely:

Lemma A:

$\vdash \Box [\sim at(\ell_1) \lor \sim at(m_1)]$

Proof:

Take

$\varphi(\pi_1, \pi_2; y): \ (at(\ell_1 + at(m_1 + y = 1) \land (y \geq 0)$.
In expressions such as the above we interpret propositions as having the numerical value 1 when true and 0 otherwise.

We can easily show that \( \varphi \) is preserved under every transition. For example, consider the transition \( \ell_0 \rightarrow \ell_1 \). When it is enabled, we have \( y > 0 \), and the transition assigns to the variable \( y \) the value \( y - 1 \) which is nonnegative. Considering the value of the sum

\[
\ell_1 + m_1 + y,
\]

\( \ell_1 \) changes from 0 to 1 on this transition but \( y \) is decremented by 1. Consequently the value of the sum remains invariant.

Initially, \( \ell_1 + m_1 + y = 0 + 0 + 1 = 1 \) and \( y = 1 \geq 0 \).

Hence \( \varphi \) satisfies the two premises of the \( \text{INV} \) rule, from which we conclude

\[
I_1 : \quad \square [\ell_1 + m_1 + y = 1] \land (y \geq 0)].
\]

This implies

\[
\square [m_1 \leq 1]
\]

which is equivalent to Lemma A.

The second property is that of accessibility. It states that each process will eventually be admitted to its critical section. This is established by:

**Lemma B:**

\[
\vdash \ell_0 \supset \Diamond \ell_1
\]

and

\[
\vdash m_0 \supset 0 m_1
\]

**Proof:**

Let us define

\[
\varphi_1 : \quad \ell_0 \land m_1 \land y = 0
\]

\[
\psi_1 : \quad y > 0
\]

We show that \( \varphi_1 \) and \( \psi_1 \) satisfy the conditions of the \( \text{EVNT} \) rule with \( k = 2 \).

In fact the only enabled transition is \( m_1 \rightarrow m_2 \) which does lead from \( \varphi_1 \) to \( \psi_1 \). While at \( m_1 \), \( P_2 \) is always enabled. Thus we conclude:

1. \( \vdash \ell_0 \land m_1 \land y = 0 \supset \Diamond(y > 0) \)

by \( \text{EVNT} \) with \( k = 2 \)
2. \( \varphi_2 \vdash at\ell_0 \land atm_1 \supset (y > 0) \) by \( I_1 \) above, 1 and PR

3. \( \varphi_2 \vdash at\ell_0 \land atm_{2,3} \supset (y > 0) \) also by \( I_1 \) and PR

4. \( \varphi_2 \vdash y > 0 \) by T1, 2, 3, LOC and PR

Take now

\( \varphi_2 : at\ell_0 \)

\( \psi_2 : at\ell_1 \)

We check premises A to C in the EVNT rule with respect to the pair \( \{ \varphi_2, \psi_2 \} \) taking \( k = 1 \). Clearly \( P \) always leads from \( \varphi_2 \) to \( \varphi_2 \lor \psi_2 \). The process \( P_1 \) always leads (when enabled) from \( \varphi_2 \) to \( \psi_2 \). Condition C is guaranteed by 4 above. We therefore conclude

5. \( \varphi_2 \vdash at\ell_0 \supset 0 at\ell_1 \).

By a completely symmetric argument we can show that:

\( \varphi_2 \vdash atm_0 \supset 0 atm_1 \).

### 12. EXAMPLE 3: MUTUAL EXCLUSION

As a third example we consider a program that solves the mutual exclusion problem without semaphores:

\[
(y_1, y_2, t) := (false, false, 1)
\]

\( \ell_0 : \) Noncritical Section
\( \ell_1 : y_1 := true \)
\( \ell_2 : t := 1 \)
\( \ell_3 : \) if \( y_2 = false \) then go to \( \ell_5 \)
\( \ell_4 : \) if \( t = 1 \) then go to \( \ell_3 \)
\( \ell_5 : \) Critical Section
\( \ell_6 : y_1 := false \)
\( \ell_7 \) go to \( \ell_0 \)
\( \ell_8 \) go to \( \ell_9 \)

\( \ell_0 \) to \( \ell_9 \)

\( m_0 : \) Noncritical Section
\( m_1 : y_2 := true \)
\( m_2 : t := 2 \)
\( m_3 : \) if \( y_1 = false \) then go to \( m_5 \)
\( m_4 : \) if \( t = 2 \) then go to \( m_3 \)
\( m_5 : \) Critical Section
\( m_6 : y_2 := false \)
\( m_7 : \) go to \( m_0 \)

\( m_8 \) go to \( m_9 \)

\( m_9 \) go to \( m_0 \)

\( m_0 \) to \( m_9 \)

For convenience we will abbreviate formulas \( at\ell_i \) to \( \ell_i \).
The principle of operation of this program is that each process $P_i$ has a variable $y_i$, $i = 1, 2$, which expresses the process’s wish to enter its critical section. The variable $y_i$ is set to true at $l_1$ and $m_1$ and reset to false at $l_6$ and $m_6$, respectively. In addition, each process leaves a signature in the common variable $t$. The process $P_1$ sets it to 1 at $l_2$ and $P_2$ sets it to 2 at $m_2$. A process $P_i$ may enter its critical section only if either $y_j = false$ (meaning that the other process is not interested) or if $t = j$, for $j \neq i$. The latter case corresponds to both processes being interested in entering the critical section but $P_j$ being the last to pass through the signing instructions at $(l_2, m_2)$.

To formally prove that this program is correct we first prove several invariance properties.

**Lemma A:**

$\vdash y_1 \equiv l_{2..6}$

Here $l_{2..6}$ stands for at $l_{2..6}$. Thus the lemma states that

$y_1 = true$ if and only if $\pi_1 \in \{l_2, l_3, l_4, l_5, l_6\}$.

**Proof:**

To prove the lemma we take

$\varphi_1 : (y_1 \equiv l_{2..6})$

and show that it is invariant under every transition, i.e., every transition leads from $\varphi_1$ to $\varphi_1$.

The only transitions that can affect the truth of $\varphi_1$ are $l_1 \rightarrow l_2$ and $l_6 \rightarrow l_7$.

In $l_1 \rightarrow l_2$ both $y_1$ and at $l_{2..6}$ become simultaneously true. Similarly in $l_6 \rightarrow l_7$ both $y_1$ and at $l_{2..6}$ become simultaneously false. Thus

1. $\vdash (y_1 \equiv l_{2..6}) \supset O(y_1 \equiv l_{2..6})$ by TRNS

2. $\vdash \{at(l_6, m_0) \land [(y_1, y_2, t) = (false, false, 1)]\} \supset (y_1 \equiv l_{2..6})$

3. $\vdash \Box (y_1 \equiv l_{2..6})$ by 1, 2 and TINV

**Lemma B:**

$\vdash y_2 \equiv m_{2..6}$

The lemma is proved by a symmetric argument.

**Lemma C:**

$\vdash (t = 1) \lor (t = 2)$
This lemma states that the only possible values of the variable \( t \) are 1 or 2.

**Proof:**

The Lemma is clearly provable by the IIIV principle. Obviously, it is true initially since \( t = 1 \). The only transitions that modify the value of \( t \) set it either to 1 or to 2. Thus \( P \) always leads to a state satisfying \((t = 1) \lor (t = 2)\).

**Lemma D:**

\[
\vdash \ell_{5,6} \supset [(\neg y_2) \lor (t = 2) \lor m_2]
\]

**Proof:**

Let \( \varphi_2 \) stand for \( \ell_{5,6} \supset [(\neg y_2) \lor (t = 2) \lor m_2] \).

It is clearly true initially since \( \vdash \ell_0 \supset \neg \ell_{5,6} \). To show that every transition leads from \( \varphi_2 \) to \( \varphi_2 \), consider the only transitions that may falsify \( \varphi_2 \), i.e., that may possibly lead from \( \varphi_2 \) to \( \neg \varphi_2 \). Potentially they are:

- \( \ell_3 \rightarrow \ell_5 \). This transition is possible only under \( \neg y_2 \) which makes \((\neg y_2) \lor (t = 2) \lor m_2 \) true.

- \( \ell_4 \rightarrow \ell_5 \). This is possible only when \( t \neq 1 \) which by Lemma C makes \((\neg y_2) \lor (t = 2) \lor m_2 \) again true.

The other transitions we should consider are transitions of \( P_2 \) while \( P_1 \) is already at \( \ell_{5,6} \). The only ones to be considered are those which affect any of the variables in \( \neg y_2 \lor (t = 2) \lor m_2 \).

- \( m_1 \rightarrow m_2 \). Causes \( m_2 \) to become true.

- \( m_2 \rightarrow m_3 \). Causes \( t \) to be set to 2.

- \( m_6 \rightarrow m_7 \). Sets \( y_2 \) to false, making \( \neg y_2 \) true.

The lemma follows by the IIIV principle.

**Lemma E:**

\[
\vdash m_{5,6} \supset [(\neg y_1) \lor (t = 1) \lor \ell_2]
\]

The lemma is proved by a completely symmetric argument.
Theorem:

\[ \vdash (\neg \ell_{5,6}) \lor (\neg m_{5,6}) \]

This theorem proves the mutual exclusion of the processes.

Proof:

1. \[ \vdash (\ell_{5,6} \land m_{5,6}) \supset [((\neg y_2) \lor (t = 2) \lor m_2) \land ((\neg y_1) \lor (t = 1) \lor \ell_2)] \]
   by lemmas C, D and PR

2. \[ \vdash (\ell_{5,6} \land m_{5,6}) \supset [y_1 \land y_2 \land \neg \ell_2 \land \neg m_2] \]
   by lemmas A, B, LOC and PR

3. \[ \vdash (\ell_{5,6} \land m_{5,6}) \supset [(t = 1) \land (t = 2)] \]
   by 1, 2 and PR

4. \[ \vdash (\neg \ell_{5,6} \land m_{5,6}) \]
   by the equality axioms and PR, using the domain fact that \(1 \neq 2\)

5. \[ \vdash (\neg \ell_{5,6}) \lor (\neg m_{5,6}) \]
   by PR

Next we will prove accessibility. We will only prove:

Theorem:

\[ \vdash at\ell_1 \supset \Diamond at\ell_5 \]

The result for \(P_2\) is completely symmetric.

Proof:

The proof will proceed by a sequence of statements most of which are proved by the EVNT rule in the version whose conclusion is \(\varphi \supset 0 \psi\). Simple passages justified by propositional temporal reasoning will not be fully presented and their omission is denoted by mentioning PTR in the justification clause.

1. \[ \vdash (\ell_4 \land m_{3,4} \land t = 2) \supset 0 \ell_5 \]
   by EVNT with \(k = 1\), using lemma A

2. \[ \vdash (\ell_3 \land m_{3,4} \land t = 2) \supset \Diamond (\ell_4 \land m_{3,4} \land t = 2) \]
   by EVNT with \(k = 2\), using lemmas A, B

3. \[ \vdash (\ell_3 \land m_{3,4} \land t = 2) \supset 0 \ell_5 \]
   by 2, 1 and OC

4. \[ \vdash (\ell_3 \land m_{3,4} \land t = 2) \supset 0 \ell_5 \]
   by 1, 3 and PR

5. \[ \vdash (\ell_3 \land m_2) \supset \Diamond [\ell_5 \lor (\ell_3 \land m_{3,4} \land t = 2)] \]
   by EVNT with \(k = 2\)
6. \( \vdash (\ell_{3,4} \land m_2) \supset \Diamond \ell_5 \)
   \[ \text{by 4, 5 and PTR} \]

7. \( \vdash (\ell_{3,4} \land m_1) \supset \Diamond [\ell_5 \lor (\ell_{3,4} \land m_2)] \)
   \[ \text{by EVNT with } k = 2 \]

8. \( \vdash (\ell_{3,4} \land m_1) \supset \Diamond \ell_5 \)
   \[ \text{by 7, 6 and PTR} \]

9. \( \vdash (\ell_3 \land m_0) \supset \Diamond [\ell_5 \lor (\ell_{3,4} \land m_1)] \)
   \[ \text{by EVNT with } k = 1 \]

10. \( \vdash (\ell_3 \land m_0) \supset 0 \ell_5 \)
    \[ \text{by 9, 8 and PTR} \]

11. \( \vdash (\ell_4 \land m_0) \supset \Diamond [\ell_5 \lor (\ell_{3,4} \land m_1) \lor (\ell_3 \land m_0)] \)
    \[ \text{by EVNT with } k = 1 \]

12. \( \vdash (\ell_4 \land m_0) \supset 0 \ell_5 \)
    \[ \text{by 11, 8, 10 and PTR} \]

13. \( \vdash (\ell_{3,4} \land m_0) \supset \Diamond \ell_5 \)
    \[ \text{by 10, 12 and PR} \]

14. \( \vdash (\ell_{3,4} \land m_0) \supset \Diamond [\ell_5 \lor (\ell_{3,4} \land m_0)] \)
    \[ \text{by EVNT with } k = 2 \]

15. \( \vdash (\ell_{3,4} \land m_7) \supset \Diamond \ell_5 \)
    \[ \text{by 14, 13 and PTR} \]

16. \( \vdash (\ell_{3,4} \land m_6) \supset \Diamond (\ell_{3,4} \land m_7) \)
    \[ \text{by EVNT with } k = 2 \text{ and lemma E} \]

17. \( \vdash (\ell_{3,4} \land m_6) \supset \Diamond \ell_5 \)
    \[ \text{by 16, 15 and PTR} \]

18. \( \vdash (\ell_{3,4} \land m_5) \supset \Diamond (\ell_{3,4} \land m_6) \)
    \[ \text{by EVNT with } k = 2 \text{ and lemma E} \]

19. \( \vdash (\ell_{3,4} \land m_5) \supset \Diamond \ell_5 \)
    \[ \text{by 18, 17 and PTR} \]

20. \( \vdash (\ell_{3,4} \land m_4 \land t = 1) \supset \Diamond (\ell_{3,4} \land m_5) \)
    \[ \text{by EVNT with } k = 2 \text{ and lemma A} \]

21. \( \vdash (\ell_{3,4} \land m_4 \land t = 1) \supset 0 \ell_5 \)
    \[ \text{by 20, 19 and PTR} \]

22. \( \vdash (\ell_{3,4} \land m_3 \land t = 1) \supset \Diamond (\ell_{3,4} \land m_4 \land t = 1) \)
    \[ \text{by EVNT with } k = 2 \text{ and lemma A} \]

23. \( \vdash (\ell_{3,4} \land m_3 \land t = 1) \supset 0 \ell_5 \)
    \[ \text{by 22, 21 and PTR} \]

24. \( \vdash (\ell_{3,4} \land m_3 \land t = 1) \supset 0 \ell_5 \)
    \[ \text{by 21, 23 and PR} \]

25. \( \vdash (\ell_{3,4} \land m_3 \land t = 1) \supset \Diamond \ell_5 \)
    \[ \text{by 4, 24, lemma C and PR} \]

We may summarize now as follows:

26. \( \vdash \ell_{3,4} \supset [\ell_{3,4} \land (m_0 \lor m_1 \lor m_2 \lor m_3 \lor m_4 \lor m_5 \lor m_6 \lor m_7)] \)
    \[ \text{by LOC} \]

27. \( \vdash \ell_{3,4} \supset 0 \ell_5 \)
    \[ \text{by 26, 13, 8, 6, 25, 19, 17, 15 and PTR} \]

28. \( \vdash \ell_2 \supset \Diamond \ell_{3,4} \)
    \[ \text{by EVNT with } k = 1 \]

29. \( \vdash \ell_2 \supset 0 \ell_5 \)
    \[ \text{by 27, 28 and OC} \]

30. \( \vdash \ell_1 \supset \Diamond \ell_2 \)
    \[ \text{by EVNT with } k = 1 \]

31. \( \vdash \ell_1 \supset 0 \ell_5 \)
    \[ \text{by 29, 30 and } \Diamond C \]
F. COMPACT PROOF PRINCIPLES

In the preceding sections we introduced a comprehensive proof system for proving arbitrary temporal properties of concurrent programs. However, as demonstrated in the last examples a fully formal proof tends to be rather lengthy and sometimes tedious to follow. Consequently we will next discuss shorter and more compact representations of proofs and corresponding compact proof principles. All these principles can be derived in the basic proof system presented above. Consequently, a proof according to these principles can always be mechanically expanded into a more detailed proof using just the basic axioms. We will discuss the three main classes of properties one may wish to prove about programs, namely: invariance, liveness and precedence properties.

13. THE INVARIANCE PRINCIPLE

The IINV principle does not significantly simplify formal proofs. Most of the needed work in applying the IINV principle is in establishing the premise that the program $P$ leads from $\varphi$ to $\varphi$. Several heuristics or meta-rules can be suggested in order to reduce the number of transitions that have to be checked, which in the worst case is proportional to the size of the program. For example:

a) Only transitions that modify variables on which $\varphi$ depends should be checked.

b) Assume that $\varphi$ has the form $\varphi = \varphi_1 \lor \varphi_2$ (similarly for implication), and that some variables $y_1, \ldots, y_m$ appear only in $\varphi_1$. Then, in checking transitions that only modify those variables, it is sufficient to check transitions that may falsify $\varphi$ and one may assume in checking them that $\varphi_2 = false$.

c) Assume that an invariance $\chi$ has already been established before. Let

$$[\varphi \land \chi] \supset (\sim \ell)$$

for some location $\ell$. Then no transitions of the form $\ell \rightarrow \ell'$ need ever be considered in showing that $P$ leads from $\varphi$ to $\psi$.

A simple generalization of the IINV rule is given by:

\[
\text{Generalized Invariance Rule -- GINV}
\]

\[
\begin{align*}
A. & \quad \vdash \varphi \supset \psi \\
B. & \quad \vdash [at & A \bar{y} = g(C)] \supset \varphi \\
C. & \quad \vdash P \text{ leads from } \varphi \text{ to } \varphi \\
& \quad \vdash \Box \Box \psi
\end{align*}
\]

Certainly premises B and C establish $\vdash \Box \varphi$ according to IINV, from which by premise A and the $\Box \Box \Box$ rule, $\vdash \Box \psi$ follows.
The advantage of the GINV principle is that no additional temporal reasoning is required and the rule can be proved complete by itself. By this we mean that, given a program $P$, any state property $\psi$ which is invariant for all executions of $P$ can be proven invariant by a single application of the GINV rule and no additional temporal reasoning.

**Theorem:**

The GINV rule is complete for proving invariance properties.

**Proof:**

Let $\psi = \psi(\overline{x}; \overline{z}; \overline{y})$ be a state property, possibly dependent on the input variables $\overline{z}$. We define a state $s = (\overline{e}; \overline{q})$ to be $\overline{z}$-accessible in $P$ if there exists a segment of some computation initialized with $\overline{z} = \overline{\xi}$ that reaches $s$, i.e.,

$$\langle \overline{e}_0; g(\overline{z}) \rangle \rightarrow \cdots \rightarrow \langle \overline{e}; \overline{q} \rangle.$$

Define the predicate $\varphi = \varphi(\overline{x}; \overline{z}; \overline{y})$ by:

$$\varphi(\overline{\xi}; \overline{e}; \overline{q}) = \text{true} \iff \langle \overline{e}; \overline{q} \rangle \text{ is } T\text{-accessible}.$$

Thus, $\varphi$ characterizes all the states that are $\overline{z}$-accessible. We will show that the predicate $\varphi$ so defined satisfies, together with $\psi$, all the premises required by the rule GINV.

Consider premise A. Since $\psi$ is invariantly true in all computations of $P$ it must be true for every accessible state $\langle \overline{e}; \overline{q} \rangle$. Consequently

$$\varphi(\overline{\xi}; \overline{e}; \overline{q}) \supset \psi(\overline{\xi}; \overline{e}; \overline{q});$$

when generalized to arbitrary $\overline{\xi}$, $\overline{e}$ and $\overline{q}$. This implies

$$\vdash \varphi \supset \psi.$$

Since we assume that the underlying domain theory is adequate for proving all classically sound formulas this implies

$$\vdash \varphi \supset \psi.$$

Consider now premise B. Since every initial state is by definition accessible we certainly have

$$\vdash \varphi(\overline{x}; \overline{e}_0; g(\overline{z})).$$

Again by completeness of our domain part with respect to classical formulas, this leads to

$$\vdash [at \overline{e}_0 A \ y = g(\overline{x})] \supset \varphi(\overline{x}; \overline{z}; \overline{y}).$$

Finally, consider premise C. Clearly every transition in $P$ leads from an $\overline{z}$-accessible state to another $\overline{z}$-accessible state. Consequently

$$\vdash P \text{ leads from } \psi \text{ to } \varphi.$$
From this premise $C$ follows by completeness of the domain part.

In the preceding theorem we have only shown the existence of an appropriate state predicate $\varphi$. We have not discussed the question of the exact formal language in which such a predicate can be expressed. However, assuming that our domain contains the integers or some isomorphic structure, and using a first-order language, it is not difficult to show that the statement:

"There exists a finite computation of $P$ leading from $\langle t_0, g(t) \rangle$ to $\langle t, \eta \rangle"\)

can be Gödel-encoded into a first-order statement over the integers.

14. LIVENESS PRINCIPLES

As a typical example of a detailed proof of liveness properties we may examine the proof of accessibility for the mutual exclusion program (Example 3). The structure of such a proof proceeds through a chain of events characterized by state assertions. Let the eventuality to be proved be $\varphi \supset \Diamond \psi$ where both $\varphi$ and $\psi$ are state properties. We may regard $\psi = \varphi_0$ as being the last assertion in the chain. Then we identify an assertion $\varphi_1$ such that by a single application of the EVNT principle we can prove

$$\vdash \varphi_1 \supset \Diamond \psi.$$ 

In the example considered we have

$$\psi : \ell_5$$

$$\varphi_1 : \ell_4 \land m_{3,4} \land (t = 2).$$

Next, we identify an assertion $\varphi_2$ such that by a single application of the EVNT principle we can prove

$$\vdash \varphi_2 \supset \Diamond(\varphi_1 \lor \psi).$$

In the general step, we identify an assertion $\varphi_i$ such that by a single application of the EVNT principle we can prove

$$\vdash \varphi_i \supset \Diamond(\bigvee_{j < i} \varphi_j).$$

Finally we have to prove $\varphi \supset (\bigvee_{i=0}^r \varphi_i)$ where $\varphi_1, \ldots, \varphi_r$ is the chain of assertions constructed. We may summarize this proof pattern by the following proof principle:
The Huin Reasoning Proof Principle --- CHAIN

Let $\varphi_0, \varphi_1, \ldots, \varphi_r$ be a sequence of state properties satisfying the following requirements:

A. $\vdash P$ leads from $\varphi_i \lor \bigvee_{j \leq i} \varphi_j$ for $i = 1, \ldots, r$.

B. For every $i > 0$ there exists a $k = k_i$ such that:

$\vdash P_k$ leads from $\varphi_i$ to $\bigvee_{j < i} \varphi_j$.

C. For $i > 0$ and $k = k_i$ as above:

$\vdash \forall i > 0 \left( \forall j < i \left( \varphi_i \supset \emptyset \left( \bigvee_{j < i} \varphi_j \lor \text{Enabled}(P_k) \right) \right) \right) \supset \bigvee_{i=0}^r \bigcup_{i=1}^r \varphi_i$

Proof:

To justify this principle we will prove by induction on $n$, $n = 0, 1, \ldots, r$, that

$\vdash \bigvee_{i=0}^n \varphi_i \supset \bigcup_{i=1}^n \varphi_i$.

For $n = 0$ we have $\vdash \varphi_0 \supset \varphi_0$ from which trivially follows by axiom $A_9$

$\vdash \varphi_0 \supset (\text{false} \lor \varphi_0)$.

Note that we interpret an empty disjunction as false.

We assume that the statement above has been proved for certain $n$ and we attempt to prove it for $n + 1$.

Consider the EVNT rule with $\varphi = \varphi_{n+1}$, $\psi = (\bigvee_{i=0}^n \varphi_i)$. By premise $A$ of CHAIN we obtain that $P$ leads from $\varphi_{n+1} = \varphi$ to

$\left( \bigvee_{j \leq n+1} \varphi_j \right) = (\varphi_{n+1} \lor \bigvee_{j \leq n} \varphi_j) = (\varphi \lor \psi)$.

This provides premise $A$ of EVNT. Let $k = k_{n+1}$. Then by premise $B$ of CHAIN, $P_k$ leads from $\varphi_{n+1} = \varphi$ to $\left( \bigvee_{j < n+1} \varphi_j \right) = \psi$. Similarly, premise $C$ of CHAIN yields that

1. $\vdash \varphi \supset \Diamond (\psi \lor \text{Enabled}(P_k))$. 

79
By the EVNT rule it follows that

2. \( \vdash \varphi \supset \varphi \cup \psi \)

or

3. \( \vdash \varphi_{n+1} \supset \varphi_{n+1} \cup (\bigvee_{i=0}^{n} \varphi_i) \).

By the induction hypothesis and the \( \text{UU} \) rule this yields

4. \( \vdash \varphi_{n+1} \supset \varphi_{n+1} \cup (\bigvee_{i=1}^{n} \varphi_i) \cup \varphi_0 \).

Again by the induction hypothesis using part of \( \text{Ag} \), \( w_2 \supset w_1 \cup w_2 \), wc can obtain

5. \( \vdash (\bigvee_{i=0}^{n} \varphi_i) \supset \varphi_{n+1} \cup (\bigvee_{i=1}^{n} \varphi_i) \cup \varphi_0 \).

Combining this with 4 above yields

6. \( \vdash (\bigvee_{i=0}^{n+1} \varphi_i) \supset \varphi_{n+1} \cup (\bigvee_{i=1}^{n} \varphi_i) \cup \varphi_0 \).

By \( \text{T38} \), \( p \cup (q \cup r) \supset (p \lor q) \cup r \), wc can reduce the nesting depth of the \( \cup \) operator to get:

7. \( \vdash (\bigvee_{i=0}^{n+1} \varphi_i) \supset (\bigvee_{i=0}^{n+1} \varphi_i) \cup \varphi_0 \)

as needed.

Taking \( n = r \) concludes the proof of the principle.

In presenting a proof according to the chain-reasoning principle it is usually sufficient to identify \( \varphi_0, \varphi_1, \ldots, \varphi_r \) and for each \( i \) to point out the "helpful" process \( \varphi_k = \varphi_{k_i} \). It can be left to the reader to verify that premises A to C are satisfied for each \( i = 1, 2, \ldots, r \).

We prefer to present such proofs in the form of a diagram. Consider a diagram consisting of nodes that correspond to the assertions \( \varphi_0, \varphi_1, \ldots, \varphi_r \). For each transition affected by some process \( \varphi_j \), that leads from a state \( s \) satisfying \( \varphi_i \) to a state \( s' \) satisfying \( \varphi_{\ell} \), \( \ell < i \), wc draw an edge from the node \( \varphi_i \) to the node \( \varphi_{\ell} \) and label it by \( \varphi_j \), the name of the responsible process. All edges corresponding to the helpful process \( \varphi_k = \varphi_{k_i} \) are drawn as double arrows. We do not explicitly draw edges corresponding to transitions from \( \varphi_i \) back to itself. However it is assumed that such edges may exist for all but the helpful process for \( \varphi_i \).

As an example wc present a diagram form of the proof of accessibility for the Mutual Exclusion program. It is given in Fig. 1. in constructing such a proof wc may freely use any invariants previously derived.

80
Fig. 1. Proof Diagram for the Mutual Exclusion Program
In this program, and typically in all non-terminating programs that have no semaphore instructions, we do not have to check premise C of the CHAIN or EVNT rule. This is because in non-terminating programs without semaphores every process is continuously enabled and therefore condition C is automatically satisfied.

In contrast let us consider the proof of accessibility for example 2—a program with semaphores. Here we want to prove $\ell_0 \supset \lozenge \ell_1$. The main diagram here is very simple:

\[ \ell_0 \xrightarrow{P_1} \ell_1 \]

It denote: a single application of the EVNT rule with $\varphi : \text{at } \ell_0$ and $\psi : \text{at } \ell_1$ with $P_k = P_1$ being the helpful process.

However, in order to justify premise C, which is not trivial in this case, we have to prove

$\vdash \ell_0 \supset \lozenge (\ell_1 \vee y > 0)$. 

For this we have to consider $P_2$'s position. If $P_2$ is at $m_0$ or $m_2$ then $y = 1$ by the invariant $I_1$ proved above. The only other case is when $P_2$ is at $m_1$ where by a single application of the EVNT rule it will eventually move to $m_2$ producing a positive value of $y$. This may be represented by a secondary diagram:

\[ \ell_{0, m_1} \xrightarrow{P_2} \ell_{0, y > 0} \]

The diagram representation of a proof according to the CHAIN principle is very similar to the proof lattices introduced in [OL] as a concise presentation of a proof of a liveness property. A superficial difference is that they choose to represent as edges the consequences of the EVNT rule, while in our representation edges stand for the premises of the EVNT rule which are also the premises to the CHAIN rule. To illustrate this difference, consider the following trivial program:

\[ \ell_0 : y := y \quad m_0 : \text{go to } m_0 \]
\[ \ell_1 : \]
\[ -P_1 - \quad -P_2 - \]

The liveness property to be proved is $\ell_0 \supset 0 \ell_1$. Below are diagram representations of the CHAIN principle and a proof lattice according to [OL].

As we see, the CHAIN diagram contains a self-edge, labelled by $P_2$ (this time drawn explicitly) and a helpful edge labelled by $P_1$. The process $P_1$ is guaranteed to get us to $\ell_1$. As a consequence
of this, by the EVNT rule, \( \ell_0 \supset \varnothing \ell_1 \). This conclusion is represented in the proof lattice by a single edge from \( \ell_0 \) to \( \ell_1 \). Thus, the different choices of representation lead to the following minor syntactical differences between CHAIN diagrams and proof lattices:

(a) Proof lattices are acyclic, whereas CHAIN diagrams are only weakly acyclic, i.e., may contain self-loops.

(b) In CHAIN diagrams, edges are labelled by the processes responsible for the transition. Special identification is provided for edges traversed by the helpful process. In proof lattices, we no longer care about the identities of the processes since progress along the lattice has already been established.

However these differences are minor and a simple procedure for translation between CHAIN diagrams and proof lattices exists. The important part in both is the identification of the intermediate assertions that are represented as nodes. In constructing a proof, this is usually the creative and most demanding process. Both graph presentations provide a natural and intuitive representation of these assertions and the precedence relations between them.

The chain-reasoning principle assumed a finite number of links in the chain. It is quite adequate for finite-state programs, i.e., programs whose variables range over finite domains. However, once we consider programs over the integers it is no longer sufficient to consider only finitely many assertions. In fact, sets of assertions of quite high cardinality are needed. The obvious generalization of a finite set of assertions \( \{\varphi_i | i = 0, \ldots, r\} \) is to consider a single assertion \( \varphi(\alpha) \), parametrized by a parameter \( \alpha \) taken from a well-founded ordered set \( (A, \prec) \). Obviously, the most important property of our chain of assertions is that program transitions eventually lead from \( \varphi_i \) to \( \varphi_j \) with \( j < i \). This property can also be stated for an arbitrary well-founded ordering. Thus a natural generalization of the chain reasoning rule is the following:

The Well Founded Liveness Principle — WELL

Let \( (A, \prec) \) be a well-founded set. Let \( \varphi(\alpha) = \varphi(\alpha; \overline{x}; \overline{y}) \) be a parametrized state formula. Let \( h : A \rightarrow \{1 \ldots k\} \) be a helpfulness function identifying for each \( \alpha \in A \) the helpful process \( P_{h(\alpha)} \) for states in \( \varphi(\alpha) \).

\[
\begin{align*}
A. \quad \vdash P \text{ lends } \varphi(\alpha) & \text{ to } \psi \lor \left( \exists \beta \preceq \alpha . \varphi(\beta) \right) \\
B. \quad \vdash P_{h(\alpha)} \text{ leads from } \varphi(\alpha) & \text{ to } \psi \lor \left( \exists \beta < \alpha . \varphi(\beta) \right) \\
c. \quad \vdash P_{h(\alpha)} \supset \Diamond \left[ \psi \lor \left( \exists \beta < \alpha . \varphi(\beta) \right) \lor \text{Enabled}(P_{h(\alpha)}) \right] \\
\vdash \left( \exists \alpha . \varphi(\alpha) \right) \supset \left( \exists \alpha . \varphi(\alpha) \right) \cup \psi
\end{align*}
\]

A justification of this rule can again be conducted, based on induction. Now, however, induction over arbitrary well-founded sets is required.
15. EXAMPLE 4: BINOMIAL COEFFICIENT

As an example for the application of the WELL principle, we consider the following program that computes the binomial coefficient \( \binom{n}{k} \) for inputs \( 0 \leq k \leq n \).

\[
(y_1, y_2, y_3, y_4) := (n, 0, 1, 1)
\]

\[
\ell_7 : \text{if } y_1 = (n - k) \text{ then go to } \ell_1
\]
\[
\ell_6 : \text{request}(y_4)
\]
\[
\ell_5 : t_1 := y_3 \cdot y_1
\]
\[
\ell_4 : y_3 := t_1
\]
\[
\ell_3 : \text{release}(y_4)
\]
\[
\ell_2 : y_1 := y_1 - 1
\]
\[
\ell_8 : \text{go to } \ell_7
\]
\[
\ell_1 : \text{halt}
\]

\[
m_3 : \text{if } y_2 = k \text{ then go to } m_1
\]
\[
m_2 : y_2 := y_2 + 1
\]
\[
m_9 : \text{loop until } y_1 + y_2 \leq n
\]
\[
m_8 : \text{request}(y_4)
\]
\[
m_7 : t_2 := y_3 / y_2
\]
\[
m_6 : y_3 := t_2
\]
\[
m_5 : \text{release}(y_4)
\]
\[
m_4 : \text{go to } m_3
\]
\[
m_1 : \text{halt}
\]

The labelling scheme of the program has been constructed in a way that simplifies the expression of the assertion \( \varphi(\alpha) \).

The computation of this program is based on the formula:

\[
\binom{n}{k} = \frac{n \cdot (n - 1) \cdots (n - k + 1)}{1 \cdot 2 \cdots k}.
\]

The values of \( y_1 \), i.e., \( n, n - 1, \ldots, n - k + 1 \), are used to compute the numerator in \( P_1 \), and the values of \( y_2 \), i.e., \( 1, 2, \ldots, k \), are used to compute the denominator. The process \( P_1 \) multiplies \( n \cdot (n - 1) \cdots (n - k + 1) \) into \( y_3 \) while \( P_2 \) divides \( y_3 \) by \( 1 \cdot 2 \cdots k \).

The instruction

\[
m_9 : \text{loop until } y_1 + y_2 \leq n
\]

guarantees even divisibility of \( y_3 \) by \( y_2 \). It synchronizes \( P_2 \)'s operation with that of \( P_1 \) to ensure that \( y_3 \) is divided by \( i \) only after \( (n - i + 1) \) has already been multiplied into it. We rely here on the mathematical theorem that the product of \( i \) consecutive integers \( n \cdot (n - 1) \cdots (n - i + 1) \) is always divisible by \( i! \) (the quotient actually being the integer \( \binom{n}{i} \)).

The critical sections \( \ell_{3,5} \) and \( m_{5,7} \) are mutually protected by the semaphore variable \( y_4 \). This protection ensures that \( y_3 \) is not updated by \( P_2 \) between, say, the computation of \( y_3 \cdot y_1 \) and the assignment of this value to \( y_3 \). Without this protection, the updated value might have been overwritten by \( P_1 \).
We start by establishing some invariant properties of this program.

\[ I_1: \vdash (at_{3..5} + \text{at}_{m_{5..7}} + y_4 = 1) \land (y_4 \geq 0). \]

This is the usual semaphore invariant. It can be proven by observing that initially this sum equals 1, and then by considering all possible transitions. For example, the \( \ell_6 \rightarrow \ell_5 \) transition changes \( at_{3..5} \) from 0 (false) to 1 (true), and also decrements \( y_4 \) by 1, leaving however the sum constant. From \( I_1 \) we can deduce mutual exclusion of the critical sections, i.e.,

\[ \vdash (\sim l_{3..5}) \lor (\sim m_{5..7}). \]

As a consequence of this we can establish:

\[ I_2: \vdash (\ell_4 \supset t_1 = y_3 \cdot y_1) \land (m_6 \supset t_2 = y_3/y_2). \]

This holds due to the impossibility of interference by \( P_2 \) while \( P_1 \) is at \( \ell_4 \).

\[ I_3: \vdash (n-k + at_{2..6}) \leq y_1 \leq n. \]

This invariance states that \( y_1 \) always lies between \( n-k \) and \( n \). When \( P_1 \) is at \( \ell_{2..6} \), \( y_1 > n-k \), whereas \( P_1 \) is at other locations, \( y_1 \geq n-k \). To verify \( I_4 \) we need only consider the transitions:

- \( \ell_7 \rightarrow \ell_6 \) which maintains \( n-k < y_1 \leq n \), assuming it was previously known that \( n-k \leq y_1 \leq n \).
- \( \ell_2 \rightarrow \ell_8 \) which results in \( n-k \leq y_1 - 1 \leq n \) from \( n-k < y_1 \leq n \).

\[ I_4: \vdash 0 \leq y_2 \leq (k - \text{at}_{m_2}). \]

This invariance bounds the range of \( y_2 \). We need consider the transitions \( m_3 \rightarrow m_2 \) and \( m_2 \rightarrow m_4 \) which can be shown to maintain \( I_4 \).

\[ I_5: \vdash \text{at}_{m_{7..8}} \supset (y_1 + y_2) \leq n. \]

Here we should consider two transitions:

- \( m_9 \rightarrow m_8 \) which is possible only if currently \( y_1 + y_2 \leq n \).
- \( \ell_2 \rightarrow \ell_8 \) is the only transition modifying \( y_1 \). However since it decrements \( y_1 \) it certainly preserves \( y_1 + y_2 \leq n \).

Let us define the following virtual variables:

\[
\begin{align*}
y_1^* &= \text{if at}_{\ell_{2,3}} \text{ then } y_1 - 1 \text{ else } y_1 \\
y_2^* &= \text{if at}_{m_{6..9}} \text{ then } y_2 - 1 \text{ else } y_2
\end{align*}
\]
These variables are roughly equal to \( y_1 \) and \( y_2 \) respectively and differ from them by 1 in certain ranges.

\[
I_6: \quad I \cdot y_3 = [n \cdot (n - 1) \ldots (y_1^* + 1)]/[1 \cdot 2 \cdot \ldots y_2^*].
\]

To verify this invariant we have to check the transitions \( \ell_4 \rightarrow \ell_3, m_6 \rightarrow m_5 \). Making use of \( I_2 \), they can be shown to maintain \( I_6 \).

\[
I_7: \quad I \cdot [at \ell_1 \supset y_1 = (n - k)] \land [at m_1 \supset (y_2 = k)].
\]

Using \( I_6, I_7 \) and the definition of \( y_1^*, y_2^* \) we obtain partial correctness of this program, namely

\[
I \cdot (at \ell_1 \land at m_1) \supset [y_3 = \binom{k}{n}].
\]

To prove termination we will use the \textsc{WELL} rule in order to establish

\[
I \cdot 0( at \ell_1 \land at m_1).
\]

As the well-founded domain we take

\[
(A, \prec) = (\mathbb{N} \times \mathbb{N} \times \mathbb{N}, \prec_{lex}).
\]

That is, the set of \textit{triplets} of nonnegative integers ordered by lexicographic ordering. This ordering defines \( (m_1, m_2, m_3) \prec (n_1, n_2, n_3) \) iff for the lowest \( i, i = 1, 2, 3 \) such that \( m_i \neq n_i; m_i < n_i \).

For our goal assertion we take \( \psi : at \ell_1 \land at m_1 \). The parameterized assertion is given by:

\[
\varphi(\alpha; \ell_i, m_i; y_1, y_2) : (y_1 + k - y_2, j, i) = \alpha.
\]

The helpfulness function is given by:

\[
h(a) = h(r, j, i) = (\text{if } i = 1 \text{ then } 2 \text{ else } 1).
\]

Thus as long as the first process \( P_1 \) has not terminated we rely on \( P_1 \) to be the helpful process. Once it has terminated, we take \( P_2 \) to be the helpful process.

We have to show that all the three premises of the \textsc{WELL} rule are satisfied.

Consider first premise A. We have to show that every transition of \( P \) leads to \( \varphi(\beta) \) with \( \beta \preceq \alpha \) if \( \psi \) is not already satisfied. By simple inspection of all the possible transitions we find that they all lead from \( (\ell_i, m_j) \) to \( (\ell_{i'}, m_{j'}) \) such that either \( i' < i \) or \( j' < j \) except for the following transitions:

- \( \ell_2 \rightarrow \ell_8 \). But this transition decrements \( y_1 \) producing a strict decrease in \( y_1 + k \cdot y_2 \) which is the first component in \( \alpha \).

- \( m_2 \rightarrow m_9 \). In a similar way this transition increments \( y_2 \), leading to a decrease in \( y_1 + k \cdot y_2 \).

- \( m_9 \rightarrow m_9 \). This transition leaves \( \alpha \) at the same value.

Consider now premise B. As we have shown above, all transitions provide a strict decrease in \( \alpha \). The only exception is \( m_9 \rightarrow m_9 \). However this is a \&transition which is considered helpful only when \( P_1 \) is at \( \ell_1 \). By \( I_7 \), at this point \( y_1 = (n - k) \) so that in view of \( I_4 \), \( y_1 + y_2 \leq k \) and hence the only transition possible from \( m_9 \) is \( m_9 \rightarrow m_8 \).
To show premise C we have to prove that $P_h$ is always eventually enabled. Consider first the case that $h = 1$. The only location in which it is not immediately enabled is when $P_1$ is at $t_6$ while $P_2$ is at $m_{5.7}$ (in view of $I_1$). However by simple chain reasoning it is obvious that in such a case, $P_2$ will certainly reach $m_4$ in which $y_4$ becomes positive and $P_1$ enabled.

The case $h = 2$ is even simpler because it is only considered when $P_1$ is at $t_1$. Consequently, even when $P_2$ is at $m_8$, which may potentially raise some problems, we have in view of $I_1$ and at $t_1$ that $y_4 > 0$ and $P_2$ is enabled.

Thus we conclude that $\psi : at_{t_1} \land at_{m_1}$ must eventually be realized and therefore the program must terminate.

16. PRECEDENCE PROPERTIES

The next class of properties we will consider and provide proof principles for is that of precedence properties. These are properties, usually needing the $\cup$ operator for their expression, which ensure that some event precedes another event, or that a certain event will not happen until another event happens first. In view of the fact that the basic FAIR and EVNT rules did actually provide a conclusion containing the $\cup$ operator, they may be naturally utilized to form precedence proof principles which are generalizations of the corresponding liveness principles.

In the following we will often consider nested until expressions in which the nesting always occurs in the second argument. We therefore adopt the convention of representing the nested formula:

$$\varphi_n \cup (\varphi_{n-1} \cup (\ldots \varphi_1 \cup \varphi_0)\ldots))$$

by:

$$\varphi_n \cup \varphi_{n-1} \cup \ldots \varphi_1 \cup \varphi_0.$$
The Chain Rule for Precedence Properties — P-CHAIN

Let \( \varphi_0, \varphi_1, \ldots, \varphi_r \) be a sequence of state assertions, and \( 0 = p_0 < p_1 < p_2 < \ldots < p_s = r \) a partition of \([1 \ldots r] \).

A. \( \vdash P \) leads from \( \varphi_i \) to \( \left( \bigvee_{j \leq i} \varphi_j \right) \) for \( i = 1, \ldots, r \).

B. For every \( i > 0 \) there exists a \( k = k_i \) such that:

\[ \vdash P_k \text{ leads from } \varphi_i \text{ to } \left( \bigvee_{j < i} \varphi_j \right) \]

C. For \( i > 0 \) and \( k = k_i \) as above:

\[ \vdash \varphi \supset \bigcup_{j < \ell} (\psi_j \cup \psi_{j+1} \ldots \psi_{\ell}) \]

where

\[ \psi_{\ell} \text{ is } \bigvee_{p_{\ell-1} < j \leq p_\ell} \varphi_j \text{ for } \ell = 1, \ldots, s. \]

The conclusion states that starting at a state that satisfies one of the \( \varphi_i, i = 0, \ldots, r \), we are guaranteed to have a period in which \( \left( \bigvee_{j = p_{\ell-1} + 1}^{p_\ell} \varphi_j \right) \) continuously holds, followed by a period in which \( \left( \bigvee_{j = p_{\ell-2} + 1}^{p_{\ell-1} + 1} \varphi_j \right) \) continuously holds, etc., until \( \varphi_0 \) is finally realized. Any of these periods may be empty.

Proof:

To justify the soundness of this conclusion we will first prove it for the most refined partition possible, namely:

\[ \left( \bigvee_{i = 0}^r \varphi_i \right) \supset (\varphi_r \cup \varphi_{r-1} \cup \varphi_{r-2} \cup \ldots \varphi_1 \cup \varphi_0). \]

This is proved in a way similar to the justification of the corresponding liveness principle. We show, by induction on \( n, n = 0, 1, \ldots, r \), that

\[ \vdash \left( \bigvee_{i = 0}^n \varphi_i \right) \supset (\varphi_n \cup \varphi_{n-1} \cup \ldots \varphi_1 \cup \varphi_0). \]

For \( n = 0 \) we have \( \vdash \varphi_0 \supset \varphi_0 \) which is the induction statement for \( n = 0 \).
Assume that the statement above has been proved for a certain \( n \) and consider its proof for \( n + 1 \).

Consider the EVNT rule with \( \varphi = \varphi_{n+1} \), \( \psi = (\bigvee_{i=0}^{n} \varphi_i) \). As shown in the proof of the liveness case, all the premises of the EVNT rule are satisfied. Consequently we may conclude:

\[
\vdash \varphi_{n+1} \supset \varphi_{n+1} \cup (\bigvee_{i=0}^{n} \varphi_i).
\]

By the induction hypothesis and the \( \cup \cup \) rule this yields

\[
\vdash \varphi_{n+1} \supset \varphi_{n+1} \cup (\varphi_n \cup \ldots \cup \varphi_1 \cup \varphi_0).
\]

Due to \( \vdash \psi \supset (\psi \cup \psi) \) which is a consequence of axiom A9, the induction hypothesis can also be written as

\[
\vdash (\bigvee_{i=0}^{n} \varphi_i) \supset \varphi_{n+1} \cup (\varphi_n \cup \ldots \cup \varphi_1 \cup \varphi_0).
\]

Taking the disjunction of the last two gives

\[
\vdash (\bigvee_{i=0}^{n+1} \varphi_i) \supset \varphi_{n+1} \cup (\varphi_n \cup \ldots \cup \varphi_1 \cup \varphi_0),
\]

which is the required statement for \( n + 1 \).

Consider now a coarser partition:

\[
0 = p_0 < p_1 < p_2 < \ldots < p_s = r.
\]

By consecutively merging any two contiguous assertions that fall into the same partition cell, using theorem T38:

\[
\vdash (\varphi_{i+1} \cup (P_i \cup \varphi)) \supset ((\varphi_{i+1} \vee P_i) \cup \varphi),
\]

we obtain the coarser conclusion:

\[
\vdash (\bigvee_{i=0}^{n+1} \varphi_i) \supset \bigvee_{p_{s-1} < j \leq p_s} \varphi_j \cup \bigvee_{p_{s-2} < j \leq p_{s-1}} \varphi_j \cup \ldots \cup (\bigvee_{0 < j \leq p_1} \varphi_j) \cup \varphi_0).
\]

**Examples:**

As our first example, let us consider the Mutual Exclusion program analyzed above. We have already proven that mutual exclusion is maintained by this program. We have also proven the liveness property that if \( P_1 \) wishes to enter its critical section it will eventually gain access to it. A more discriminating question is that, of how fair is our algorithm. That is, if \( P_1 \) wishes to enter
its critical section, how many times will \( P_2 \) be able to enter its own critical section before \( P_1 \)? Is that number bounded? We refer to this question as the problem of bounded overtaking. Namely, how many times can \( P_2 \) overtake \( P_1 \) before \( P_1 \) enters his critical section.

Our first analysis makes use of Fig. 1 without any modifications. We only read from it the stronger conclusion according to the stronger P-CHAIN rule. As a partition we choose \( p_1 = 7, p_2 = 9, p_3 = r = 11 \). Consequently, from the diagram of Fig. 1 we conclude by the P-CHAIN rule:

\[
\vdash \left( \bigvee_{i=1}^{11} \varphi_i \right) \supset \left( \bigvee_{i=8}^{11} \varphi_i \right) \cup \left( \bigvee_{i=1}^{9} \varphi_i \right) \cup \left( \bigvee_{i=1}^{7} \varphi_i \right) \cup \varphi_0
\]

Replacing each of the right hand side disjunctions by a weaker property and the left hand side disjunction by a stronger statement we obtain:

\[
\vdash \ell_{3,4} \supset \left( \sim m_{5,6} \right) \cup m_{5,6} \cup \left( \sim m_{5,6} \right) \cup \ell_5
\]

This implies that if \( P_1 \) is at the waiting loop in \( \ell_{3,4} \), there will be a period in which \( P_2 \) is not in the critical section \( m_{5,6} \), followed by a period in which \( P_2 \) is inside the critical section \( m_{5,6} \), followed by a period in which \( P_2 \) is outside the critical section which terminates by \( P_1 \) entering his critical section. Since any of these periods may be empty this is a worst-case analysis. But it certainly assures 1-bounded overtaking, i.e., once \( P_1 \) is in \( \ell_{3,4} \), \( P_2 \) may overtake it at most once.

Having successfully analyzed the situation from \( \ell_{3,4} \) on we may attempt to obtain a similar analysis from the moment that \( P_1 \) enters \( \ell_2 \).

This analysis calls for a refinement of the diagram of Fig. 1. The following is a subdiagram that should replace the node corresponding to \( \varphi_{12} \) in Fig. 1. It consists of three nodes labeled respectively \( \varphi_{7.5} \), \( \varphi_{9.5} \) and \( \varphi_{11.5} \). The fractional indexing indicates that \( \varphi_{7.5} \) should be inserted between \( \varphi_7 \) and \( \varphi_8 \) in Fig. 1. The edges out of \( \varphi_{13} \) should enter one of these three nodes. Edges out of \( \varphi_{7.5} \) lead to some of \( \varphi_{11}, \ldots, \varphi_7 \).

Similarly for edges out of \( \varphi_{9.5} \) and \( \varphi_{11.5} \). Considering the updated diagram composed of Fig. 1 and the above subdiagram we obtain the following conclusion:

\[
\vdash \ell_{2,4} \supset \left( \bigvee_{i=8}^{11.5} \varphi_i \right) \cup \left( \bigvee_{i=8}^{9.5} \varphi_i \right) \cup \left( \bigvee_{i=1}^{7.5} \varphi_i \right) \cup \varphi_0
\]
This again leads to

\[ \vdash \ell_{2.4} \supset ((\sim m_{5,6}) \cup m_{5,6} \cup (\sim m_{5,6}) \cup \ell_5), \]

which ensures \( l \)-bounded overtaking even from \( \ell_2 \). Encouraged by this, we may next ask whether a similar result can be obtained from \( \ell_1 \). Unfortunately this is not the case. \( P_2 \) may enter its critical section an arbitrary number of times while \( P_1 \) is at \( \ell_1 \). This is obvious since while being at \( \ell_1 \), \( P_1 \) has not yet modified any variable in a way that will show that it is not still in \( \ell_0 \). And naturally while \( P_1 \) is at \( \ell_0 \), \( P_2 \) may enter the critical section any number of times if the algorithm is correct.

**THE WELL-FOUNDED PRINCIPLE FOR PRECEDENCE PROPERTIES**

A natural extension of the P-CHAIN rule to programs that require infinite chains of assertions again uses well founded ordered sets.

Let \( (A, \prec) \) be a well founded ordered set, we require however that the ordering is total (or linear). That is, for every two distinct elements \( \alpha_1, \alpha_2 \in A \) either \( \alpha_1 \prec \alpha_2 \) or \( \alpha_2 \prec \alpha_1 \).

**Well Founded Precedence Rule --- P-WELL**

Let \( \varphi(\alpha) = \varphi(\alpha; \pi; \psi) \) be a parametrized state assertion with \( \alpha \in A \).

Let \( h : A \rightarrow [1 \ldots k] \) be a helpfulness function.

Let \( \alpha_1 \prec \alpha_2 \prec \ldots \prec \alpha_s \) be a sequence of elements of \( A \).

\( t \)-**P** leads from \( \varphi(\alpha) \) to \( \psi \lor (\exists \beta \leq \alpha . \varphi(\beta)) \)

\( t \)-**P** leads from \( \varphi(\alpha) \) to \( \psi \lor (\exists \beta < \alpha . \varphi(\beta)) \)

\( t \)-**P** leads from \( \varphi(\alpha) \) to \( \psi \lor (\exists \beta < \alpha . \varphi(\beta)) \lor Enabled(P_{h(\alpha)}) \)

\[ \vdash (\exists \alpha \leq \alpha_s . \varphi(\alpha)) \supset (\psi_s \cup \psi_{s-1} \cup \ldots \psi_1 \cup \psi) \]

where

\( \psi_{\ell} \) is \( \exists \beta(\alpha_{\ell-1} \prec \beta \leq \alpha_{\ell}) . \varphi(\beta) \) for \( \ell = 2, \ldots, s \), and

\( \psi_1 \) is \( \exists \beta(\beta \leq \alpha_1) . \varphi(\beta) \)

Note that while the range of the parameter in the assertions is infinite, the partition is still finite.

**Acknowledgement:**

We thankfully acknowledge the help extended to us by Yoni Malachi, Ben Moszkowski, Stuart Russell, and Francine Yellin in reading the manuscript. Special thanks are due Lo Evelyn Eldridge-Diaz for TEXing the manuscript and Lo Carol Weintraub for typing its first draft.
REFERENCES


