Verification of Concurrent Programs, Part II: Temporal Proof Principles

by

Zohar Manna
Amir Pnueli

Department of Computer Science
Stanford University
Stanford, CA 94305

Approved for public release; distribution unlimited.
In this paper, the second of a series on the application of temporal logic to concurrent programs, we present proof methods for establishing invariance (safety) and eventuality (liveness) properties. The proof principle for establishing invariance properties is based on computational induction, and is a generalization of the inductive assertion method. For a restricted class of concurrent programs we present an algorithm for the automatic derivation of invariant assertions.
In order to establish eventuality properties we present several proof principles that translate the structure of the program into basic temporal statements about its behavior. These principles can be viewed as providing the temporal semantics of the program. The basic statements thus derived are then combined into temporal proofs for the establishment of eventuality properties. This method generalizes the intermittent assertion method.

The proof principles are amply illustrated by examples.
VERIFICATION OF CONCURRENT PROGRAMS:
TEMPORAL PROOF PRINCIPLES

by

ZOHAR MANNA
Computer Science Department
Stanford University
Stanford, CA
and
Applied Mathematics Department
The Weizmann Institute
Rehovot, Israel

AMIR PNUELI
Applied Mathematics Department
The Weizmann Institute
Rehovot, Israel

Abstract

In this paper, the second of a series on the application of temporal logic to concurrent programs, we present proof methods for establishing invariance (safety) and eventuality (liveness) properties.

The proof principle for establishing invariance properties is based on computational induction, and is a generalization of the inductive assertion method. For a restricted class of concurrent programs we present an algorithm for the automatic derivation of invariant assertions.

In order to establish eventuality properties we present several proof principles that translate the structure of the program into basic temporal statements about its behavior. These principles can be viewed as providing the temporal semantics of the program. The basic statements thus derived are then combined into temporal proofs for the establishment of eventuality properties. This method generalizes the intermittent assertion method.

The proof principles are amply illustrated by examples.


This paper appears in the Proceedings of the Workshop on Logics of Programs (Yorktown Heights, NY), Springer-Verlag Lecture Notes in Computer Science, 1981.

This research was supported in part by the National Science Foundation under grants MCS79-09495 and MCS80-06930, by the Office of Naval Research under Contract N00014-76-C-0687, and by the United States Air Force Office of Scientific Research under Grant AFOSR-81-0014.
INTRODUCTION

In a previous report [MP2] we introduced the temporal framework for reasoning about concurrent programs. We described the model of concurrent programs that we study which is based on interaction via shared variables and defined the concept of fair execution of such programs. We then demonstrated the application of the temporal logic formalism to the expression of properties of concurrent programs. Program properties of interest can be classified according to the syntactic form of the temporal formula expressing them; we studied three classes of properties: invariance properties, eventuality properties and precedence properties. We have shown that almost all of the program properties that were ever considered or studied for either sequential or concurrent programs fall into one of these three categories. These include properties such as partial correctness, clean behavior, global invariants, mutual exclusion, safety, deadlock absence, output integrity in the invariance category; total correctness, intermittent assertion realization, accessibility, liveness, responsiveness in the eventuality category; and safe liveness, absence of unsolicited response, FIFO responsiveness and general precedence in the precedence category.

In this paper, a sequel to [MP2], we concentrate on the application of the temporal logic formalism to proving these properties. We would thus present methods for establishing that a given program indeed possesses a certain property. In principle, once a property has been expressed within the temporal logic formalism, and an appropriate temporal characterization of the behavior of the given program derived ([MAN1], [MP1], [PNU1], [PNU2]), the task of proving that the property holds for this program reduces to proving the validity of a certain temporal implication. This implication states that every sequence of states, if it is a fair computation of the given program, has the desired property.

These principles can be justified by the general temporal formalism, and once justified, provide direct, simple, and intuitive rules for the establishment of these properties. They usually replace long but repetitively similar chains of primitive steps in more detailed proofs, and help us focus on the higher level overview of the proof while retaining the necessary standard of rigor.

Previous attempts to develop proof techniques for concurrent programs include [KEL], [LAM] and [OG].

In our exposition, we assume that the reader is familiar with the concepts and definitions introduced in our first paper of this series - [MP2].

THE INVARiance PRINCIPLE

Consider a typical concurrent program $P$ of form

$$ (\bar{y} := f_0(x)); [P_1] \ldots [P_m] $$

with input parameters $\bar{x} = (x_1, \ldots, x_k)$ and shared program variables $\bar{y} = (y_1, \ldots, y_n)$ over a domain $D$. Let $\psi$ be a classical formula, i.e., a formula with no modal operators. The basic idea in proving that the formula $\psi$ is an invariant of the program $P$, i.e.

$$ \varphi(\bar{x}) \supset \Box \psi, $$

is to show that:
(a) the precondition \( \varphi(x) \) implies that \( \psi \) is true initially.

(b) \( \psi \) is preserved by any possible transition of the program \( P \); that is, if it were true before the transition then it also will be true after the transition.

We can then infer the invariance of \( \psi \) under the precondition \( \varphi(x) \).

To state the result more precisely, let \( Q(\pi; \gamma) \) be a "state property", i.e., it is expressed by a classical formula with no temporal operators, which may refer to the location variables \( \pi \), the program variables \( \gamma \), and possibly some global variables.

Let

\[
\begin{array}{c}
\circ \quad \alpha \\
\ell \quad c_{\alpha}(\gamma) \rightarrow [\gamma := f_{\alpha}(\gamma)] \\
\alpha \quad \alpha'
\end{array}
\]

be a transition in process \( P_j \) for some \( j = 1, \ldots, m \). With each such transition we associate the location transformation function \( r_{\alpha} \) given by:

\[
r_{\alpha}(\pi_1, \ldots, \pi_j, \ldots, \pi_m) = (\pi_1, \ldots, \ell', \ldots, \pi_m),
\]

i.e., the value of \( \pi_j \) is replaced by \( \ell' \), while the value of each \( \pi_i, i \neq j \), is unchanged. This transformation denotes the change in the vector \( \pi \) when transition \( \alpha \) is taken, much in the same way that \( f_{\alpha} \) denotes the change in \( \gamma \) when \( \alpha \) is taken.

The notation we use to express the location change as a transformation underlines the similarity between the location and program variables. This leads to the possible description of a transition as:

\[
\begin{array}{c}
\circ \\
[\alpha \ell \land c_{\alpha}(\gamma)] \rightarrow [(\pi; \gamma) := (r_{\alpha}(\pi); f_{\alpha}(\gamma))] \\
\alpha
\end{array}
\]

A property \( Q(\pi; \gamma) \) is said to be inductive for \( P \) if the following verification condition holds for each transition \( \alpha \) in \( P \):

\[
V_{\alpha} : [\alpha \ell \land c_{\alpha}(\gamma) \land Q(\pi; \gamma)] \supset Q(r_{\alpha}(\pi); f_{\alpha}(\gamma)).
\]

Intuitively, \( Q \) is inductive if it is inherited along every transition i.e., if it was true before the transition and the transition was enabled, it will necessarily be true after the transition. Note that the verification condition is classical, in the sense that it contains no temporal operators, and can therefore be established using classical proof techniques.

Our proof rule for invariance may now be formulated as follows:
The Invariance Principle

Let $Q(x; y)$ be a state property of a program $P$ such that:

1. $Q$ is true initially; i.e.,
   \[
   I: \ [\exists t_0 \wedge \varphi(x)] \supset Q(x; f_0(x))
   \]
   holds, where $t_0 = (t_0^1, \ldots, t_0^n)$ the vector of initial locations.

2. $Q$ is inductive for $P$; i.e., the verification condition
   \[
   V_\alpha: \ [\exists t \wedge c_\alpha(y) \wedge Q(x; y)] \supset Q(r_\alpha(x); f_\alpha(y))
   \]
   holds for every transition $\alpha$ in $P$.

Then we may deduce
\[
\forall [\exists t_0 \wedge \varphi(x)] \supset \Box Q(x; y).
\]

Condition 1 ensures that $Q$ is true initially, provided we restrict ourselves to inputs $x$ satisfying $\varphi$ and condition 2 ensures that once $Q$ is true it remains so. The conclusion is that $Q$ is invariantly true for all $(P, \varphi)$-computations.

Note that this proof principle reduces the proof of a temporal formula of the invariance class into a classical proof of a set of formulas, namely the initial condition $I$ and the verification conditions $V_\alpha$.

The principle of invariance described here is the most general method known for proving invariance properties of concurrent programs. It can be shown to underlie all other proposed proof methods for invariance properties.

PRAGMATIC CONSIDERATIONS IN CHECKING FOR INDUCTIVENESS

In principle, when checking for the inductiveness of an assertion $Q$ one has to check the verification condition $V_\alpha$ for all transitions $\alpha$ in the program. However, in practice, we can immediately discard many transitions as automatically preserving $Q$, based on syntactic considerations alone.

If the property $Q$ does not contain any of the location variables $x$, then the required verification conditions $V_\alpha$ are reduced to
\[
V'_\alpha: \ [c_\alpha(y) \wedge Q(y)] \supset Q(f_\alpha(y)).
\]

In particular, $V'_\alpha$ is trivially true for any transition $\alpha$ where $f_\alpha$ does not modify the variables on which $Q$ actually depends.

A typical case is that of semaphores. We have the following property:

*The Semaphore Variable Rule:* For a semaphore variable $y$,
if its initial value is a nonnegative integer
and if it is modified only by request and release instructions, then

\[ \vartriangleleft (y \geq 0). \]

The only two instructions that may modify the value of a semaphore variable are:
request(y), which is equivalent to

\[ y > 0 \to [y := y - 1] \]

and release(y), which is equivalent to

\[ \text{true} \to [y := y + 1] \]

For the request case the verification condition is

\[ [(y > 0) \land (y \geq 0)] \supset (y - 1 \geq 0). \]

For the release transition the verification condition is

\[ [\text{true} \land (y \geq 0)] \supset (y + 1 \geq 0). \]

Both conditions are trivially true. Thus, since the initial value of the semaphore variable y is nonnegative and it is modified only through the semaphore instructions request(y) and release(y), it follows, by the Invariance Principle, that y is invariantly nonnegative, i.e. \[ \vartriangleleft (y \geq 0). \]

For another example, let us consider a typical assertion of the form:

\[ Q(\pi; y) : \text{at } L \supset \phi(y), \]

where L is a set of locations in P and \( \phi \) does not depend on the location variables. For an arbitrary transition \( \alpha \) of the form

\[ \ell \xrightarrow{c_\alpha(y)} [y := f_\alpha(y)] \xrightarrow{\alpha} \ell' \]

the verification condition is

\[ V_\alpha : \{c_\alpha(y) \land [(\ell \in L) \supset \phi(y)]\} \supset [(\ell' \in L) \supset \phi(f_\alpha(y))], \]

or equivalently,

\[ \{c_\alpha(y) \land [(\ell \not\in L) \lor \phi(y)] \land (\ell' \in L) \supset \phi(f_\alpha(y))\}. \]

There are three cases to consider.
Case: $\ell' \not\in L$ (outside or leaving $L$). Then $V_\alpha$ is trivially true, since the antecedent of the implication is false.

Case: $\ell \not\in L$, $\ell' \in L$ (entering $L$). Then $V_\alpha$ is reduced to

$$c_\alpha(y) \supset \phi(f_\alpha(y)).$$

Case: $\ell, \ell' \in L$ (within $L$). Then $V_\alpha$ is reduced to

$$[c_\alpha(y) \land \phi(y)] \supset \phi(f_\alpha(y)).$$

Thus, we only have to consider $\alpha$'s which fall into the two latter cases.

**EXAMPLE: CONSUMER-PRODUCER**

Let us illustrate an application of the invariance principle to the Consumer-Producer program (program $CP$ of [MP2]).

$$b := A, \quad s := 1, \quad cf := 0, \quad ce := N$$

$$\ell_0 : compute y_1 \quad m_0 : request(cf)$$
$$\ell_1 : request(ce) \quad m_1 : request(s)$$
$$\ell_2 : request(s)$$
$$m_2 : y_2 := head(b)$$

\[
\begin{align*}
\ell_3 & : t_1 := b \times y_1 \\
\ell_4 & : b := t_1 \\
\ell_5 & : release(s) \\
\ell_6 & : release(cf) \\
\ell_7 & : go to \ell_0
\end{align*}
\]

\[
\begin{align*}
m_3 & : t_2 := tail(b) \\
m_4 & : b := t_2 \\
m_5 & : release(s) \\
m_6 & : release(ce) \\
m_7 & : compute using y_2 \\
m_8 & : go to m_0
\end{align*}
\]

--- $P_1$: Producer --- $P_2$: Consumer --

The producer $P_1$ computes a value into $y_1$ without using any other program variables; the computation details being irrelevant. It then adds $y_1$ to the end of the buffer $b$. The consumer $P_2$ removes the first element of the buffer into $y_2$ and then uses this value for its own purposes (at $m_7$). It is assumed that the maximal capacity of the buffer $b$ is $N > 0$. The ‘compute using $y_2$’ instruction references $y_2$ but does not modify any of the shared program variables.

In order to ensure the correct synchronization between the processes we use three semaphore variables: The variable $s$ ensures that the accesses to the buffer are protected and provides exclusion between the sections ($\ell_3, \ell_4, \ell_5$) and ($m_2, m_3, m_4, m_5$). The variable $ce$ ("count of empties") counts the number of free available slots in the buffer $b$. The variable $cf$ ("count of fulls") counts how many items the buffer currently holds.
The initial condition is given by:
\[ \text{at}t_0 \land \text{at}m_0 \land (b = A) \land (s = 1) \land (cf = 0) \land (ce = N). \]

We will use invariances to prove several properties of this program.

First, we observe that due to the semaphore variable rule

(1) \[ \Box [ (s \geq 0) \land (cf \geq 0) \land (ce \geq 0) ]. \]

Mutual Exclusion

The exclusive access to the critical sections

\[ L = \{ \ell_3, \ell_4, \ell_5 \} \]
\[ M = \{ m_2, m_3, m_4, m_5 \} \]

can be expressed as:

\[ \Box \sim (\text{at} \ell \land \text{at} M), \]

i.e., it is never the case that \( \pi_1 \in L \) and \( \pi_2 \in M \) simultaneously.

Since only one \( \text{at} \ell \) and only one \( \text{at} m \), can be true at a given instant it is sufficient to prove:

(2) \[ \Box [(\text{at} \ell + \text{at} M) \leq 1]. \]

Note the mixed notation that treats propositions as numerically valued with \( \text{true} = 1, \text{false} = 0 \).

Formula (2) states an invariance property. It will be proved by showing the invariance of the assertion:

\[ Q_1 : \quad \text{at} \ell + \text{at} M + s = 1. \]

By the invariance principle we have to show that \( Q_1 \) is true initially and that \( Q_1 \) is inductive for \( P \).

Initially, we have that \( s = 1 \) and that \( \text{at} t_0 = \text{at} m_0 = 1 \) which implies that \( \text{at} \ell = \text{at} M = 0 \). Thus the left-hand side of the equality in \( Q_1 \) evaluates to 1 and we have that \( Q_1 \) holds initially.

Next, we have to check that \( Q_1 \) is inductive, i.e., preserved by every transition in \( P \). From inspection of the variables on which \( Q_1 \) depends, it is clear that it is sufficient to check the transitions that either modify \( s \) or modify the \( \text{at} \ell \) or \( \text{at} M \) propositions. The only candidates for modifying \( Q_1 \) are therefore the transitions \( \ell_2 \rightarrow \ell_3, \ell_5 \rightarrow \ell_6, m_4 \rightarrow m_2, \) and \( m_5 \rightarrow m_6 \).

Take, for example, the transition \( \ell_2 \rightarrow \ell_3 \). Going through this transition changes \( \text{at} \ell \), from 0 to 1 increasing the sum by 1. But, as \( s \) is decremented by 1, the sum remains constant. Similar checks of the other transitions will show that they all leave the sum invariant. This establishes the inductiveness of \( Q_1 \).
We may therefore conclude by the Invariance Principle that

\[ \square Q_1 \]

i.e., \( Q_1 \) is an invariant of the program \( P \).

The combination of \( \square Q_1 \) and the semaphore property \( \square (s \geq 0) \) implies property (2) that proves mutual exclusion.

**Proper Management of the Buffer**

Here we would like to show that

\( (3) \quad \square (0 \leq |b| \leq N), \)

i.e., the buffer's maximum capacity is never exceeded throughout the execution and no attempt is made to remove an element from an empty buffer.

We first establish the invariance of the following inductive assertion:

\[ Q_2 : \quad cf + ee + atI_{2..6} + atm_{1..6} = N \]

We use here our abbreviated notation, where \( atI_{2..6} \) stands for \( at\{l_2, \ldots, l_6\} \), i.e., \( \pi_1 \in \{l_2, \ldots, l_6\} \), and \( atm_{1..6} \) stands for \( at\{m_1, \ldots, m_6\} \), i.e., \( \pi_2 \in \{m_1, \ldots, m_6\} \). As before, the whole conjunction is interpreted arithmetically: 1 standing for true and 0 for false. By inspection of the relevant transitions we verify that \( Q_2 \) is indeed inductive and initially true, and thus is invariant, i.e.,

\[ \square Q_2. \]

Next consider another necessary invariant assertion:

\[ Q_3 : \quad cf + atl_{5,6} + atm_{1,4} = |b|, \]

where \( |b| \) is the size of the buffer \( b \). To establish the invariance of \( Q_3 \) we have to also establish the invariance of

\[ Q_4 : \quad atl_{4} \cup (|t_1| = |b| + 1) \]

and

\[ Q_5 : \quad atm_{4} \cup (|t_2| + 1 = |b|). \]

We will check for the joint invariance of \( Q_3, Q_4, \) and \( Q_5 \) and establish \( \square (Q_3 \land Q_4 \land Q_5) \).

The conjunction \( Q_3 \land Q_4 \land Q_5 \) is initially of the form \( (0 = 0) \land (false \lor \ldots) \land (false \lor \ldots) \) which is clearly true.

In order to check the inductiveness of \( Q_3 \land Q_4 \land Q_5 \) we must check every relevant transition of the program \( CP \). Let us consider two typical transitions:

\[ \quad \]
\( \ell_3 \rightarrow \ell_4: \)

\( Q_3 \) and \( Q_5 \) are not affected at all. In \( Q_4 \), both \( \text{atl}_4 \) and \( |t_1| = |b| + 1 \) become true on this transition, so that \( Q_4 \) is true after the transition.

\( \ell_4 \rightarrow \ell_5: \)

Here, \( Q_3, Q_4, \) and \( Q_5 \) are all affected by the transition and we would like, therefore, to illustrate the proof of a verification condition along this transition in greater detail. The verification condition is:

\[
\begin{align*}
[ \text{atl}_4 \land Q_3(\pi; \bar{y}) \land Q_4(\pi; \bar{y}) \land Q_5(\pi; \bar{y}) ] \\
\implies [ Q_3(r(\pi); f(\bar{y})) \land Q_4(r(\pi); f(\bar{y})) \land Q_5(r(\pi); f(\bar{y})) ]
\end{align*}
\]

where

\[
\begin{align*}
& \quad r(\pi_1, \pi_2) = (s_5, s_2) \\
& \quad f(b, s, cf, ce, t_1, t_2) = (t_1, s, cf, ce, t_1, t_2).
\end{align*}
\]

The proof proceeds in the following steps:

1. \( \text{atl}_4 \) \hspace{1cm} \text{given}
2. \( \text{atl}_5, b = 0 \) \hspace{1cm} \text{from 1}
3. \( cf + \text{atm}_{1..4} = |b| \) \hspace{1cm} \text{by \( Q_3 \)}
4. \( |t_1| = |b| + 1 \) \hspace{1cm} \text{by \( Q_4 \) using 1}
5. \( cf + 1 + \text{atm}_{1..4} = |b| + 1 \) \hspace{1cm} \text{by adding 1 to both sides of 3}
6. \( cf + (t_5 \in \{s_5, s_6\}) + \text{atm}_{1..4} = |t_1| \) \hspace{1cm} \text{from 5 using 4}
7. \( Q_3(r(\pi); f(\bar{y})) \) \hspace{1cm} \text{by definition of \( r \) and \( f \) using \( Q_3 \)}

Consider next \( Q_4(r(\pi); f(\bar{y})) \):

8. \( (t_5 = t_4) \supset (|t_1| = |t_1| + 1) \) \hspace{1cm} \text{tautology}
9. \( Q_4(r(\pi); f(\bar{y})) \) \hspace{1cm} \text{by definition of \( r \) and \( f \) using \( Q_4 \)}

As for \( Q_5(r(\pi); f(\bar{y})) \):

10. \( \sim \text{atm}_4 \) \hspace{1cm} \text{by 1 and mutual exclusion (2)}
11. \( \text{atm}_4 \supset (|t_2| + 1 = |t_1|) \) \hspace{1cm} \text{from 12}
12. \( Q_5(r(\pi); f(\bar{y})) \) \hspace{1cm} \text{by definition of \( r \) and \( f \) using \( Q_5 \)}
This concludes the proof of the verification condition for transition $t_4 \rightarrow t_5$. Therefore $Q_3 \land Q_4 \land Q_5$ is inductive along the transition $t_4 \rightarrow t_5$. We can similarly check that it is inductive along all the other transitions.

Thus we have established:

$$\Box (Q_3 \land Q_4 \land Q_5).$$

Let us now proceed to infer the proper management of the buffer $b$, i.e., $\Box (0 \leq |b| \leq N)$.

First observe that by $Q_3$, $|b|$ is equal to a sum of variables all of which are nonnegative. Thus we have

$$\Box (|b| \geq 0).$$

On the other hand we have by $Q_3$ and $Q_2$ that

$$|b| - cf = at_{5,6} + atm_{1..4} \leq at_{2..6} + atm_{1..8} = N - (cf + ce)$$

The first equality is a direct consequent of $Q_3$. The inequality results from the fact that $\{\ell_5, \ell_6\}$ is a subset of $\{\ell_2, \ldots, \ell_6\}$ and $\{m_1, \ldots, m_4\}$ is a subset of $\{m_1, \ldots, m_6\}$. The second equality is a direct consequence of $Q_2$.

Thus, we have

$$|b| - cf \leq N - (cf + ce)$$

which simplifies to

$$|b| \leq N - ce.$$ 

Since $ce$ is a semaphore variable we have $ce \geq 0$ which gives

$$\Box (|b| \leq N).$$

Thus we conclude that property (3),

$$\Box (0 \leq |b| \leq N),$$

holds.
Comments

• Modifying the program

The need for the auxiliary invariants $Q_4$ and $Q_5$ resulted from the splitting of the statements concerning $b$ into several statements according to the single-access rule.

Having first established the mutual exclusion of the regions $L = \{\ell_5, \ell_4, \ell_3\}$ and $M = \{m_2, \ldots, m_5\}$ we can observe that $b$ is not really a shared variable, in that only one process at a time can access it. Correspondingly, we could transform the program, after having established exclusion, by replacing

\[
\ell_3 : \ t_1 := b \circ y_1 \\
\ell_4 : \ b := t_1
\]

by

\[
\ell'_3 : \ b := b \circ y_1
\]

and

\[
m_2 : \ y_2 := head(b) \\
m_3 : \ t_2 := tail(b) \\
m_4 : \ b := t_2
\]

by

\[
m'_2 : \ (y_2, b) := (head(b), tail(b)).
\]

This would greatly simplify the subsequent analysis by making $Q_3$ directly verifiable without using $Q_4$ and $Q_5$.

• Using virtual variables

Instead of introducing the auxiliary invariants $Q_4, Q_5$ it is possible to define a virtual variable $b^*$ by:

\[
b^* = \text{if } at_{\ell_4} \text{ then } t_1 \text{ else (if } at_{m_4} \text{ then } t_2 \text{ else } b)
\]

and then directly prove a modified version of $Q_3$:

\[
Q_3^* : \ c f + at_{\ell_4, 6} + at_{m_{1..3}} = |b^*|.
\]

The variable $b^*$ represents the intended value of $b$, where we use $t_i$ ($i = 1, 2$) instead of $b$ if $b$ is about to be changed to $t_i$. Because we are focusing our attention on the value as soon as it is obtained, we have modified $Q_3$ by extending the region $\{\ell_5, \ell_6\}$ into $\{\ell_4, \ell_5, \ell_6\}$ and contracting $\{m_1, m_2, m_3, m_4\}$ into $\{m_1, m_2, m_3\}$.

A SYSTEMATIC SEARCH FOR LINEAR INVARIANTS

In order to dispel the illusion of "magically" drawing the invariants $Q_1, Q_2, Q_3$ out of thin air, let us describe a method for a systematic search for such invariants. (See also [FRA], [CLA].)
An invariant of the form discussed here is composed of three parts, such that the sum of the first two is equal to the third. We represent such an invariant by:

\[(B + Z) = C.\]

(a) \(B\) is the body of the invariant and is a linear expression in the semaphore variables and other variables which are incremented by constants (linearly) during cycles in the program.

(b) \(Z\) is a sum of expressions of the form \(\pi_j \in I\) for some region \(I \subseteq L_j\) and will be called a compensation expression.

(c) \(C\) is a constant.

We start constructing such an invariant by finding an appropriate body.

(a) In the body we look for a linear combination of variables \(E = \sum a_i y_i\) such that the net change in each cycle of each process is 0. Obviously, we restrict ourselves to cyclic programs, i.e., non-terminating programs, in which each process eventually returns to its initial location \(l_0\) and to variables whose change along a cycle is constant and independent of the program flow. Semaphore variables usually have this property.

Let us denote for these variables the net change in \(y_i\) resulting from a full cycle in process \(I_j\) by \(\Delta^j_i\). Then our combination \(E = \sum a_i y_i\) should satisfy

\[\Delta^j E = \Sigma a_i \Delta^j_i = 0\]

for \(j, 0 \leq j \leq m\). That is, we require that the value of the expression remains unchanged as a result of a complete cycle of each of the processes.

In our consumer-producer example all our variables are linearly incremented and we have the following table:

<table>
<thead>
<tr>
<th>(\Delta^1_i)</th>
<th>(\Delta^2_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

We look for a combination

\[E = a_1 \cdot s + a_2 \cdot b + a_3 \cdot cf + a_4 \cdot ce\]

such that \(\sum a_i \Delta^j_i = 0\) for \(j = 1, 2\). This yields the set of equations

\[a_1 \cdot 0 + a_2 + a_3 - a_4 = 0\]
\[a_1 \cdot 0 - a_2 - a_3 + a_4 = 0.\]

We will be interested in a nontrivial set of independent solutions to these equations.
In this case the equations possess three degrees of freedom, and hence three linearly independent solutions are possible. The exact choice is irrelevant and we pick the following:

1. \( a_1 = 1 \quad a_2 = a_3 = a_4 = 0 \)
2. \( a_3 = a_4 = 1 \quad a_1 = a_2 = 0 \)
3. \( a_2 = a_4 = 1 \quad a_1 = a_3 = 0 \).

Thus for the following independent linear combinations, the net change in each cycle of each process is 0:

\[
\begin{align*}
B_1 & : s \\
B_2 & : cf + ce \\
B_3 & : |b| + ce.
\end{align*}
\]

Note that \( B_1 \) and \( B_2 \) correspond to the bodies of \( Q_1 \) and \( Q_2 \) respectively, while \( B_3 \) is a different invariant which will enable us to derive the same conclusion as the combination of \( Q_2 \) and \( Q_3 \). For the choice \( a_1 = a_4 = 0, a_2 = -1 \) and \( a_3 = 1 \), we could get \( B_3 : cf - |b| \) which corresponds to \( Q_3 \) itself.

(b) Having a body \( B \), to derive the right-hand side \( C \) of the invariant, we only have to substitute the initial values implied by \( \phi(x) \) into the body. Doing this for our three invariants we obtain:

\[
\begin{align*}
C_1 & : 1 \\
C_2 & : N \\
C_3 & : N.
\end{align*}
\]

(c) Next, we determine the compensation expressions. Consider a given \( C \) and \( B(y) \) and a process \( P_j \) with locations \( \{t_0, \ldots, t_e \} \). Since we assume cycling, \( t_e \) is not a terminal location but branches back to \( t_0 \). By our assumption, the changes in \( B(y) \) can be traced and are constant. Denote by \( B_i(y) \) the value of \( B \) at location \( t_i, i = 0, 1, \ldots e \) in the process, and let

\[
\delta_i = B_0(y) - B_i(y).
\]

Then the compensating expression for process \( P_j \) is given by

\[
Z_j = \sum_{i=0}^{e} \delta_i \cdot (at_{t_i}).
\]

For example, to evaluate \( \delta_i \) for \( B_1 = s \) in \( P_1 \) above we have to compute:

\[
s|_{at_{t_0}} - s|_{at_{t_1}}.
\]

Assuming that \( P_1 \) is operating alone, (which is the basic assumption in the computation of the \( \delta_i \)) we take the difference between the value of \( s \) at \( t_1 \) and its initial value at \( t_0 \). Thus, we have

\[
\delta_0 = \delta_1 = \delta_2 = \delta_6 = \delta_7 = 0,
\]
since when \( P_1 \) is being executed alone the value of \( s \) at locations \( \ell_0, \ell_1, \ell_2, \ell_5, \ell_7 \) is equal to the value of \( s \) at \( \ell_0 \), i.e., \( s = 1 \). Moreover,

\[
\delta_3 = \delta_4 = \delta_5 = 1;
\]

since when \( P_1 \) is executing alone, the value of \( s \) at locations \( \ell_3, \ell_4, \ell_5 \) is smaller by 1 than the value of \( s \) at \( \ell_0 \). Hence, the compensation expression for the body \( s \) in \( P_1 \) is

\[
Z_1 = at \ell_3 + at \ell_4 + at \ell_5.
\]

Computing the compensation expression \( Z_j \) for the body \( H \) for each process \( P_j \) we form the full invariant:

\[
B + \sum_{j=1}^{m} Z_j = C.
\]

For the three bodies we considered, we obtain the following three invariants:

\[
\begin{align*}
  I_1 & : s + at \ell_3.5 + at m2.6 = 1 \\
  I_2 & : cf + cc + at \ell_2.6 + at m1.6 = N \\
  I_3 & : |b| + cc + at \ell_2.4 + at m5.6 = N.
\end{align*}
\]

Note that \( Q_3 \) can be obtained by forming the difference \( I_2 - I_3 \).

This method of deriving invariants has the advantage that no further proof is needed; indeed, any invariant derived by the method is automatically a true invariant of the program. But it may only be applied to variables which are modified by a constant in atomic instructions, or to programs which can be transformed so as to satisfy this restriction.

**EXAMPLE: BINOMIAL COEFFICIENT**

Consider next the program \( BC ([MP^2]) \) for the distributed computation of the binomial coefficient \( \binom{n}{k} \) for input parameters \( n \geq k \geq 0 \).

*Program \( BC_1 \) (Binomial Coefficient first version)*

\[
y_1 := n, \quad y_2 := 0, \quad y_3 := 1, \quad y_4 := 1
\]

\[
\ell_0 : \text{if } y_1 = (n - k) \text{ then go to } \ell_6 \\
\ell_1 : \text{request}(y_4) \\
\ell_2 : \ell_1 := y_3 \cdot y_1 \\
\ell_3 : y_3 := \ell_1 \\
\ell_4 : \text{release}(y_4) \\
\ell_5 : y_1 := y_1 - 1 \\
\ell_6 : \text{go to } \ell_0 \\
\ell_7 : \text{halt} \\
\ell_8 : \text{go to } m_0 \\
\ell_9 : \text{halt}
\]

\[
P_1 \\
\ell_0 \\
\ell_1 \\
\ell_2 \\
\ell_3 \\
\ell_4 \\
\ell_5 \\
\ell_6 \\
\ell_7 \\
\ell_8 \\
\ell_9
\]

\[
P_2 
\]
The task of computing the binomial coefficient
\[
\binom{n}{k} = \frac{n \cdot (n-1) \cdot \cdots \cdot (n-k+1)}{1 \cdot 2 \cdot \cdots \cdot k}
\]
is distributed between the two processes by having \(P_1\) perform all the multiplications while \(P_2\) is in charge of the divisions. The values of \(y_1\), i.e., \(n, n-1, \ldots, n-k+1\), are used to compute the numerator in \(P_1\) (the last value of \(y_1\), \(n-k\), is not used), and the values of \(y_2\), i.e., \(1, 2, \ldots, k\), are used to compute the denominator (the first value of \(y_2\), \(0\), is not used). The two processes must synchronize in order that the accumulated product be evenly divisible by the divisors used at \(m_4\) by \(P_2\). This synchronization is realized by the waiting loop at \(m_2\) which essentially ensures that execution will proceed to \(m_3\) only when at least \(y_2\) factors have been multiplied into \(y_3\). We rely here on the mathematical theorem that the product of \(i\) consecutive positive integers:
\[k \cdot (k+1) \cdot \cdots \cdot (k+i-1)\]
is always divisible by \(i!\).

The critical sections \(L = \{l_4, l_5, l_6\}\) and \(M = \{m_4, m_5, m_6\}\), protected by the semaphore variable \(y_4\), ensure exclusive access to the shared variable \(y_3\). Note that this program satisfies the single critical access rule ([MP2]) since for example in the expression \(y_1 + y_2\) appearing at \(m_2\) only \(y_1\) is critically accessed.

The invariant
\[
l_0: \ a l_2,4 + a l_4,6 + y_4 = 1
\]
ensures the mutual exclusion of the critical sections. It is verifiable by the invariance principle in the usual way.

Once this exclusion is established we can transform this program to a simpler program \(HC_2\) such that there is a faithful correspondence between executions of \(HC_1\) and executions of \(HC_2\). This implies that the correctness of \(HC_1\) will follow from that of \(HC_2\).

**Program HC_2 (Binomial Coefficient second version)**

\[
y_1 := n, \quad y_2 := 0, \quad y_3 := 1
\]

\[
l_0: \ if \ y_1 = (n-k) \ then \ go \ to \ l_6
\]

\[
l_1: \ y_3 := y_3 \cdot y_1
\]

\[
l_2: \ y_1 := y_1 - 1
\]

\[
l_3: \ go \ to \ l_0
\]

\[
l_6: \ halt
\]

\[
m_0: \ if \ y_2 = k \ then \ go \ to \ m_6
\]

\[
m_1: \ y_2 := y_2 + 1
\]

\[
m_2: \ loop \ until \ y_1 + y_2 < n
\]

\[
m_3: \ y_3 := y_3/y_2
\]

\[
m_4: \ go \ to \ m_0
\]

\[
m_5: \ halt
\]

\[P_1 \quad P_2\]
Next we introduce two virtual variables:

\[ y_1^* = \text{if at } \ell_2 \text{ then } y_1 - 1 \text{ else } y_1 \]
\[ y_2^* = \text{if at } m_{2,3} \text{ then } y_2 - 1 \text{ else } y_2. \]

The need for the virtual variables is similar to that of the compensation expressions discussed above. The main invariant on which the correctness of the program is based is \( I_3 \) below

\[ y_3 = \left[ n \cdot (n - 1) \cdots (y_1^* + 1) \right] / \left[ 1 \cdot 2 \cdots y_2^* \right] \]

which ties together \( y_1, y_2 \) and \( y_3 \) (or their virtual versions). It is invariant in the sense that it is preserved after \( y_1, y_2 \) and \( y_3 \) has each been properly updated. However since the updating of \( y_1 \) and \( y_3 \) in \( P_1 \) for example cannot occur simultaneously, we define \( y_1^* \) which is the anticipated updated value of \( y_1 \) as soon as \( y_3 \) is updated at \( \ell_1 \). Similarly, \( y_2^* \) differs from \( y_2 \) between the updating of \( y_2 \) and the updating of \( y_3 \) in \( P_2 \).

We use the following invariants:

1. \( I_1: \left[ (n - k + 1) \leq y_1 \leq n \right] \wedge \left[ 0 \leq y_2 \leq (k - 1) \right] \)
2. \( I_2: \text{at } m_3 \supset (y_1 + y_2) \leq n \)
3. \( I_3: y_3 = \left[ n \cdot (n - 1) \cdots (y_1^* + 1) \right] / \left[ 1 \cdot 2 \cdots y_2^* \right] \)

In \( I_3 \), the product of a zero number of terms evaluates to 1.

The initiality of \( I_1 \) to \( I_3 \) is easily verifiable.

The two parts of \( I_1 \) can be verified separately by considering the transitions \( \ell_0 \rightarrow \ell_1, \ell_2 \rightarrow \ell_3 \) and \( m_0 \rightarrow m_1, m_1 \rightarrow m_2 \) respectively.

To verify \( I_2 \) we observe that on entering \( m_3 \), \( y_1 + y_2 \leq n \) holds true. Any possible \( P_1 \) transition while \( P_2 \) is at \( m_3 \) can only decrease the value of \( y_1 + y_2 \).

Consider now the verification of \( I_3 \). The only relevant transitions are \( \ell_1 \rightarrow \ell_2 \) and \( m_3 \rightarrow m_4 \). Denoting the values of the variables after the transition by \( y_1^{*'}, y_2^{*'}, y_3^{*'} \) respectively, we obtain for \( \ell_1 \rightarrow \ell_2 \):

\[ y_3 = \left[ n \cdot (n - 1) \cdots (y_1^{*'} + 1) \right] / \left[ 1 \cdot 2 \cdots y_2^{*'} \right] \]
\[ \Rightarrow y_3 \cdot y_1 = \left[ n \cdot (n - 1) \cdots (y_1^{*'} + 1) \cdot y_1 \right] / \left[ 1 \cdot 2 \cdots y_2^{*'} \right] \]
\[ \Rightarrow y_3' = \left[ n \cdot (n - 1) \cdots (y_1^{*'} + 1) \right] / \left[ 1 \cdot 2 \cdots y_2^{*'} \right]. \]

as at \( \ell_1, y_1 = y_1^* \)

Similarly for the \( m_3 \rightarrow m_4 \) transition:

\[ y_3 = \left[ n \cdot (n - 1) \cdots (y_1^{*'} + 1) \right] / \left[ 1 \cdot 2 \cdots y_2^{*'} \right] \]
\[ \Rightarrow y_3 / y_2 = \left[ n \cdot (n - 1) \cdots (y_1^{*'} + 1) \right] / \left[ 1 \cdot 2 \cdots (y_2^{*'} + 1) \right] \]
\[ y_3 = \frac{[n \cdot (n-1) \cdots (y_1^* + 1)]}{[1 \cdot 2 \cdots y_2^*]} \]

The even divisibility of \( y_3 \) by \( y_2 \) at \( m_3 \) is ensured by the fact that by \( I_2 \) we have that

\[ y_1^* \leq y_1 \leq n - y_2. \]

Thus the number of consecutive factors in the numerator of \( y_3 \) is at least \( y_2 \) which is evenly divisible by \( y_2! \)

**PROVING EVENTUALITIES**

Here we will consider general methodologies for proving properties of the form

\[ \mathbf{I} \supset \Diamond Q. \]

Many of the cases that we will study focus on a special kind of eventualities called *accessibility statement*. Its characteristic form is

\[ \mathbf{at l} \supset \Diamond \mathbf{at l'} \]

guaranteeing that being at \( l \) we will eventually reach \( l' \). In more general form it can appear as:

\[ (\mathbf{at l} \land \phi) \supset \Diamond (\mathbf{at l'} \land \phi'), \]

where we associate a pre-condition \( \phi \) with the visit at \( l \) and a post-condition \( \phi' \) with the visit at \( l' \). The Intermittent-Assertion Method (see [BUR], [MW]) uses this implication as the basic statement for reasoning. Many useful eventuality properties are representable in this form. In this discussion we assume that \( l \) and \( l' \) belong to the same process. It is however possible to consider generalizations in which this assumption may be relaxed.

Our approach for proving eventuality properties, called *proof by eventuality chains*, is based on establishing a chain of eventualities that by transitivity leads to the ultimate establishing of the desired goal (see also [OL]). The main transitivity argument used here is:

\[ \forall \phi_1 \supset \Diamond \phi_2 \text{ and } \forall \phi_2 \supset \Diamond \phi_3 \Rightarrow \forall \phi_1 \supset \Diamond \phi_3. \]

Some common techniques that we use in our proofs are:

- We split a situation into several subcases and pursue each case to its conclusion.

- To establish implications of the form

\[ \forall (3k.\phi(k)) \supset \Diamond \phi' \]

we use induction

\[ \forall \phi(0) \supset \phi' \text{ and } \forall n. [\phi(n) \supset \Diamond (\phi(n-1) \lor \phi')] \Rightarrow \forall (3k.\phi(k)) \supset \Diamond \phi'. \]
• We frequently establish $\neg \phi \lor \phi'$ by contradiction: we assume $\phi \land \square \neg \phi'$ and pursue the consequences of this assumption. If we succeed in showing

$$\neg \left[ \phi \land \square \neg \phi' \right] \lor \text{false},$$

then we will have established our desired result. This technique is particularly useful in the verification of a statement of the form

$$\forall t \lor \diamond \neg \forall t$$

in concurrent systems. The reason for that is that by assuming $\square \forall t$ we are momentarily (for the duration of the analysis) halting one of the processes at $t$ and have only to analyze the possible movements of the other processes. This usually results in a significant simplification.

We start by presenting an example with an informal proof of its correctness relative to accessibility.

EXAMPLE: MUTUAL EXCLUSION (DEKKER) INFORMAL PROOFS

As a first example, consider the solution to the mutual exclusion problem that was first given by Dekker and described in ([Dijkstra]). Here, we assume a shared variable $t$ that may be modified by both processes and two private boolean variables $y_1$ and $y_2$, each being set only by its owning process but may be examined by the other.

Program DK (Mutual Exclusion - Dekker's Solution):

$$t := 1, \; y_1 := y_2 := F$$

$t_0$ : execute
$t_1$ : $y_1 := T$
$t_2$ : if ($y_2 = F$) then go to $t_7$
$t_3$ : if ($t = 1$) then go to $t_2$
$t_4$ : $y_1 := F$
$t_5$ : loop until ($t = 1$)
$t_6$ : go to $t_1$
$t_7$ : $t := 2$
$t_8$ : $y_1 := F$
$t_9$ : go to $t_0$

$m_0$ : execute
$m_1$ : $y_2 := T$
$m_2$ : if ($y_1 = F$) then go to $m_7$
$m_3$ : if ($t = 2$) then go to $m_2$
$m_4$ : $y_2 := F$
$m_5$ : loop until ($t = 2$)
$m_6$ : go to $m_1$
$m_7$ : $t := 1$
$m_8$ : $y_2 := F$
$m_9$ : go to $m_0$

The variable $y_1$ in process $P_1$ (and $y_2$ for $P_2$ respectively) is set to $T$ at $t_1$ to signal the intention of $P_1$ to enter its critical section at $t_7$. Next $P_1$ tests at $t_2$ if $P_2$ has any interest in entering its own critical section. This is tested by checking if $y_2 = T$. If $y_2 = F$, $P_1$ proceeds immediately to its critical section. If $y_2 = T$ we have a competition between the two processes on the access right to their critical sections. This competition is resolved by using the variable $t$ (turn) that has the
value 1 if in case of conflict \( P_1 \) has the higher priority and the value 2 if \( P_2 \) has the higher priority. If \( P_1 \) finds that \( t = 1 \) it knows it is its turn to insist and it leaves \( y_1 \) on and just loops between \( \ell_2 \) and \( \ell_3 \) waiting for \( y_2 \) to drop to \( F \). If it finds that \( t = 2 \) it realizes it should yield to the other and consequently it turns \( y_1 \) off and enters a loop at \( \ell_5 \), waiting for \( t \) to change to 1. It knows that as soon as \( P_2 \) exits its critical section it will set \( t \) to 1 so it will not be waiting forever. Once \( t \) has been detected to be 1, \( P_1 \) returns to the active competition at \( \ell_2 \).

We will proceed to prove for this program both mutual exclusion and accessibility. They are complementary properties in this case. The first assures that the two processes cannot simultaneously enter their respective critical sections. The second assures that once a process wishes to enter its critical section it will eventually get there.

Mutual exclusion

To prove mutual exclusion we show the joint invariance of the following three assertions:

\[ Q_1 : (y_1 = T) \equiv at{\ell_2, \ell_3, \ell_4, \ell_7, \ell_8} \]
\[ Q_2 : (y_2 = T) \equiv at{m_2, m_3, m_4, m_7, m_8} \]
\[ Q_3 : \sim at{\ell_1, \ell_8} \lor \sim at{m_7, m_8} \]

That is,

\[ \equiv \Box (Q_1 \land Q_2 \land Q_3), \]

where the initial condition is given by

\[ at{\ell_0} \land at{m_0} \land (t = 1) \land (y_1 = y_2 = F). \]

The inductiveness of the first two assertions is easily checked by considering the different transitions in each of the processes. They certainly hold initially.

To show the invariance of \( Q_3 \) which is the statement of mutual exclusion consider the possible transitions that could potentially falsify this assertion.

One such transition is \( \ell_2 \rightarrow \ell_7 \) while \( at{m_7, m_8} \). However by \( Q_2 \), \( at{m_7, m_8} \) implies \( y_2 = T \) so that the transition \( \ell_2 \rightarrow \ell_7 \) is disabled. Similarly for the transition \( m_2 \rightarrow m_7 \) while \( at{t_7, t_8} \).

Accessibility

Accessibility in this program is given for \( P_1 \) (the case for \( P_2 \) is similar) by

\[ \equiv at{\ell_1} \supset \Diamond at{\ell_7}. \]

The process \( P_1 \) signals its wish to enter the critical section by moving from \( \ell_0 \) to \( \ell_1 \). We then would like to prove that it eventually reaches the critical section at \( \ell_7 \).
In analyzing this program we have to interpret the execute instructions at \( \ell_0 \) and \( m_0 \) as a non-critical section. Consequently we cannot assume that being at \( \ell_0 \) we will eventually get to \( \ell_1 \). Hence the transition graph representation of the execute instruction at \( \ell_0 \) (and similarly at \( m_0 \)) should be represented as:

![Transition Graph]

That is, there is a nondeterministic choice between staying at \( \ell_0 \) and proceeding to \( \ell_1 \).

We will proceed to prove

**Theorem:** \( \models P \subseteq \Diamond at \ell_7 \).

Here we will present an informal proof of the statement, followed by the justification of some of the steps used in the proof. Motivated by recurrent patterns in the informal proof we will then introduce proof principles that could be used to construct a formal version of the same proof.

The proof of the theorem consists of a sequence of lemmas.

**Lemma A:** \( \models [at \ell_3 \land (t = 1)] \supset \Diamond at \ell_7 \)

**Proof of Lemma A:**

Assume to the contrary that \( P_1 \) never takes the \( \ell_2 \rightarrow \ell_7 \) transition; then henceforth

\[ \Box[(at \ell_2 \lor at \ell_3) \land (t = 1)] \]

since the only instruction assigning to \( t \) a value different from 1 is at \( \ell_7 \) and as long as \( t = 1 \) and the transition \( \ell_2 \rightarrow \ell_7 \) is not taken, \( P_1 \) is restricted to \( \{\ell_2, \ell_3\} \).

Under this invariance assumption \( at\{\ell_2, \ell_3\} \land (t = 1) \), let us check the locations of \( P_2 \).

**case a:** \( P_2 \) is at \( m_3 \). Then \( y_2 = F \) and will stay so. By fairness \( P_1 \) must eventually get to \( \ell_2 \) and in the next transition out of \( \ell_2 \) must go to \( \ell_7 \) (\( y_2 \) being \( F \)). Thus

\[ \models atm_5 \supset \Diamond at \ell_7. \]

**case b:** \( P_2 \) is at \( m_4 \). Then by the fairness requirement it will eventually reach \( m_5 \) so that by case a

\[ \models atm_4 \supset \Diamond at \ell_7. \]

**case c:** \( P_2 \) is at \( m_3 \). Then in the next transition out of \( m_3 \), \( t \) is still 1 so the \( m_4 \) branch must be taken. Consequently by case b

\[ \models atm_3 \supset \Diamond at \ell_7. \]

**case d:** \( P_2 \) is at \( m_2 \). Then since, by \( Q_1, (at \ell_2 \lor at \ell_3) \supset y_1 = T \), and since we assumed that \( P_1 \) is restricted to \( \{\ell_2, \ell_3\} \), the next transition of \( P_2 \) will take us to \( m_3 \). Thus by case c
also have
\[ \text{atm}_2 \supset \text{at} t_7. \]

**case e:** \( P_2 \) is at \( m_1 \). Then obviously eventually \( P_2 \) will reach \( m_2 \) so that by case \( d \) we have
\[ \text{atm}_1 \supset \text{at} t_7. \]

**case f:** \( P_2 \) is at \( m_8 \). Then eventually \( P_2 \) will get to \( m_1 \), so by case \( e \)
\[ \text{atm}_6 \supset \text{at} t_7. \]

**case g:** \( P_2 \) is at \( m_0 \). Then either it will stay in \( m_0 \) forever or eventually exit to \( m_1 \). In the case that it stays in \( m_0 \) forever we have by \( Q_2, \square(y_2 = f) \). Thus in the next transition out of \( t_2 \) we must proceed to \( t_7 \). Otherwise \( P_2 \) will eventually get to \( m_1 \) which by case \( f \) leads again to \( \text{at} t_7 \). Thus in any case
\[ \text{atm}_0 \supset \text{at} t_7. \]

**case h:** Obviously by fairness
\[ \text{atm}_7 \supset \text{atm}_8 \supset \text{atm}_9 \supset \text{atm}_0, \]
so that by case \( g \), any of these cases also leads to the eventual realization of \( \text{at} t_7 \).

Thus by analyzing all the possible values of \( \pi_2 \) in \( P_2 \) we showed that \( \text{at} t_7 \) is eventually realized in any of them. Consequently we have that
\[ \text{at} t_3 \land (t = 1) \supset \text{at} t_7. \]

which is the desired result of Lemma A.

**Lemma B:** \[ \text{at}\{\ell_3, \ldots, \ell_6\} \land (t = 2) \supset \text{at}\{m_8, m_9, m_0\} \]

**Proof of Lemma B:**

Consider first the invariance of the following statement:
\[ Q_4: \ (t = 2) \supset \text{atm}_8. \]

The transitions which may possibly falsify this statement are:
- \( \ell_7 \rightarrow \ell_8 \) while \( P_2 \) is at \( m_8 \). However, due to \( Q_3, \text{at} \ell_7 \land \text{atm}_8 \) is an impossible situation.
- \( m_7 \rightarrow m_8 \) while \( t = 2 \), but the transition sets \( t = 1 \), so that \( Q_4 \) does hold after the transition.

Having established \[ \mathbf{\Box} Q_4 \] we proceed to establish \[ \mathbf{\Box} Q_5 \] where
\[ Q_5: \ [\text{at}\{\ell_3, \ldots, \ell_6\} \land (t = 2)] \supset \text{at}\{m_9, m_0\}. \]
Let us investigate the transitions that could possibly falsify $Q_5$. The relevant transitions are:

- $t_2 \rightarrow t_3$ while $at\{m_9, m_0\}$. However by $Q_2$, $at\{m_9, m_0\}$ implies that $y_2 = F$ which disables this transition.

- $m_9 \rightarrow m_9$ while $t = 2$. However in view of $Q_4$ the situation $(t = 2) \land atm_9$ is impossible so that the transition is also impossible.

Taking the conjunction of $Q_4$ and $Q_5$ we can infer the result of Lemma B. ■

**Lemma C:** $\equiv atm_5 \supset \Diamond atm_7$.

**Proof of Lemma C:**

If we are at $t_5$ there are two possibilities. Either we will eventually get to $t_6$ with $t = 1$ or we will stay forever in $t_5$ with $t = 2$ continuously.

In the first case we proceed to $t_1$ and reach $t_2$. There we either enter $t_7$ immediately or get to $t_3$ with $t = 1$. The value of $t$ will not change on the way since the only possible change of $t$ from 1 to 2 is performed by $P_1$ at $t_7 \rightarrow t_9$. By lemma $A$, being at $t_3$ with $t = 1$ ultimately leads to $t_7$.

The other case is in which $\Box(t = 2 \land atm_5)$. By lemma $B$ we have that $\Box(\neg atm_8, m_9, m_0)$. Since $atm_5$ is permanently true so will be $y_1 = F$ by $Q_1$.

Consider now all the possible locations of $\pi_2$ in $P_2$ excluding $m_8$, $m_9$, and $m_0$:

- $atm_7$ will eventually lead us to $m_8$ and turn $t$ to 1.
- $atm_2$ will lead us to $m_7$ since $y_1 = F$ and then to $m_8$.
- $atm_3$ will lead us to $m_2$ since $t = 2$.
- $atm_4$ leads to $m_2$.
- $atm_6$ leads to $m_4$.
- $atm_5$ will eventually lead to $m_6$, having $t = 2$.
- $atm_1$ leads to $m_5$.

Consequently all the locations in $P_2$ eventually cause $t$ to turn to 1 and $P_1$ will eventually get out of $t_5$ and proceed to $t_3$ with $t = 1$. Lemma $A$ then establishes the desired result. ■

We are ready now to prove the desired accessibility theorem, that $\equiv atm_1 \supset \Diamond atm_7$.

**Proof of Theorem:**

Proceed with $P_1$ from $t_1$ to $t_2$. There we either immediately enter $t_7$ or arrive at $t_3$. Consider the next instant in which $P_1$ is scheduled. If $t = 1$ we are assured by lemma $A$ that we will ultimately get to $t_7$. If $t = 2$ we proceed to $t_4$ and $t_5$ from which we are assured by lemma $C$ of eventually getting to $t_7$. Thus we will get to $t_7$ in all cases. ■
PROOF PRINCIPLES FOR EVENTUALITIES

In order to present proofs such as the above in a more rigorous perhaps even machine checkable style, we proceed to develop several proof principles. These will enable us to establish the basic accessibility steps ensuring the eventual passage from a location to its successor under the assumption of fairness.

All predicates below are “state predicates” expressed by classical formulas, and will generally depend on the location variables \( \pi \) as well as on the program variables \( \bar{y} \).

A predicate \( \phi = \phi(\pi; \bar{y}) \) is said to be \( \chi \)-invariant, where \( \chi = \chi(\pi; \bar{y}) \), if for every transition

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

the following formula holds:

\[
\lnot \forall \ell \land e(\bar{y}) \land \chi(\pi; \bar{y}) \land \chi(\pi; f(\bar{y})) \land \phi(\pi; \bar{y}) \supset \phi(\pi; f(\bar{y})).
\]

That is, \( \phi \) is preserved by any transition which preserves \( \chi \).

In all the following we will use \( \square \chi \) to denote that \( \chi \) is an invariant externally given and guaranteed to be continuously true. It will be useful in conducting conditional proofs.

The Escape Principle for Single Location

Consider a location \( \ell \) in process \( P_j \). Let \( \Sigma = \{\alpha_1, \ldots, \alpha_k\} \) be a set of transitions originating in \( \ell \). Let \( \ell^1, \ldots, \ell^k \) be the locations to which the transitions \( \alpha_1, \ldots, \alpha_k \) lead and \( e_1, \ldots, e_k \) the enabling conditions associated with \( \alpha_1, \ldots, \alpha_k \), respectively. We do not require that \( \Sigma \) be the set of all transitions originating in \( \ell \).

We require that location \( \ell \) be deterministic, that is, the conditions \( e \) and \( e' \) on any two distinct transitions \( \alpha \) and \( \alpha' \) (not necessarily in \( \Sigma \)) originating in \( \ell \) must be disjoint, i.e., \( \sim e \sim e' \). In all the programs that we will study all locations would be deterministic except for those that contain an execute instruction. We will never apply the escape rule to these locations.
The Rule of Escape (ESC):

Let $\phi$, $\chi$, and $\psi$ be predicates such that:

A: $\phi$ is $(at \ell \land \chi)$-invariant.
   This means that as long as we stay at $\ell$ and $\chi$ is preserved, so is $\phi$.

B: Any of the $\alpha_i$, $i = 1, \ldots, k$, transitions of $\Sigma$ that preserves $\chi$ and is initiated with $\phi$ true, achieves $\psi$, i.e., $\psi$ will hold after the transition. This is expressed by

$$[at \ell \land c_i(y) \land \phi(x; y) \land \chi(x; y) \land \chi(r_i(x); f_i(y))] \supset \psi(r_i(x); f_i(y))$$

for every $i = 1, \ldots, k$.

C: $\phi \land \chi$ at $\ell$ ensures that at least one $c_i$, $i = 1, \ldots, k$, is true (the transition is enabled), i.e.,

$$[at \ell \land \phi(x; y) \land \chi(x; y)] \supset \bigvee_{i=1}^{k} c_i(y).$$

Then under these three conditions we may conclude

$$\equiv [at \ell \land \phi \land \Box \chi] \supset \Diamond \psi.$$ 

That is, being at $\ell$ with $\phi$ true and being assured of the continuous holding of $\chi$ guarantees eventual realization of $\psi$.

To justify the principle consider an execution which starts at $\ell$ with $\phi$ true and continuous assurance of $\chi$. By condition A as long as $P_j$ is not scheduled we remain at $\ell$ with $\phi \land \chi$ true. By condition C this implies that all that time $\bigvee_{i=1}^{k} c_i$ is also continuously true. Therefore by fairness eventually $P_j$ must be scheduled in a state in which $\phi$, $\chi$, $\bigvee_{i=1}^{k} c_i$ all hold. Consequently by determinism of $\ell$ one of the $\alpha_i \in \Sigma$ transitions must be taken and by condition B, $\psi$ must be realized.

There are some variations and generalizations of this basic principle which are discussed next.

The Rule of Alternatives for Regions

The first generalization considers exits out of a region (set of locations) rather than a single location. This principle applies also to nondeterministic locations.

Let $L \subseteq \ell_j$ be a set of locations in the process $P_j$ and $\Sigma = \{\alpha_1, \ldots, \alpha_k\}$ the set of all transitions originating in $L$ and leading to locations $\ell^1$, $\ldots$, $\ell^k$ outside of $L$, i.e., $\ell^i \not\in L$.  

24
The Rule of Alternatives (ALT):

Let $\phi$, $\chi$, $\psi$ be predicates such that:

A: $\phi$ is $(\text{at } L \land \chi)$ invariant.
   This means that as long as we stay in $L$ and $\chi$ is preserved so is $\phi$.

B: Any of the $\alpha_i$, $i = 1, \ldots, k$, transitions of $\Sigma$ that preserves $\chi$ and is initiated with $\phi$ true, achieves $\psi$, i.e., $\psi$ will hold after the transition. This is expressed by:

$$[[\text{at } L \land c_i(y) \land \phi(\pi; y) \land \chi(\pi; y) \land \chi(\tau_i(\pi); f_i(y))] \supset \psi(\tau_i(\pi); f_i(y))]$$

for every $i = 1, \ldots, k$.

Then under these conditions we may conclude:

$$\models [[\text{at } L \land \phi \land \Box \chi]] \supset [[\Box [\text{at } L \land \phi]] \lor \Diamond \psi].$$

That is, being initially in $L$, with $\phi$ true and being assured of the continuous holding of $\chi$ guarantees that we have two alternatives: either we stay in $L$ with $\phi$ permanently true, or achieve $\psi$.

Note that since we do not have any condition similar to C above that guarantees the eventual realization of $\psi$, we must also consider the possibility of remaining in $L$ and satisfying $\phi$ forever.

To justify the principle, consider an execution which starts in $L$ with $\phi$ true and continuous assurance of $\chi$. By condition A as long as we stay in $L$, $\phi$ will remain true. By condition B once we take any of the $\alpha_i$ transitions in this situation $\psi$ will be realized. Hence the conclusion follows.

Note that the ALT rule can be applied to a region consisting of a single location. Thus for an execute instruction:

$$\text{true} \rightarrow [[L]] \xrightarrow{\alpha_2} L \xrightarrow{\alpha_1} \text{true} \rightarrow [[L]] \xrightarrow{\alpha_1} L$$

25
we may take \( L = \{ \ell \} \) and \( \Sigma = \{ \alpha_1 \} \) to obtain

\[ \psi \ \text{at} \supset [\Box \text{at} \lor \Diamond \text{at}']. \]

The Semaphore Rule

Rule ESC above is adequate for dealing with locations for which the disjunction of all their exit conditions (on all the outgoing transitions) is identically true. A location which does not satisfy this requirement is called a semaphore location since in a semaphore request instruction, represented by

\[ \ell \quad y > 0 \rightarrow [y := y - 1] \quad \ell' \]

the exit condition \( E_t \) is \( y > 0 \) and is not identically true, nor is it necessarily continuously enabled. Consequently rules ESC and ALT are only sufficient for reasoning about programs that contain no semaphore locations. Once we have semaphore locations we need a stronger rule.

Let \( \ell \) be a (possibly semaphore) location and \( \Sigma = \{ \alpha_1, \ldots, \alpha_k \} \) the set of all the transitions originating in \( \ell \). Let \( \ell' \) and \( c_i \), for \( i = 1, \ldots, k \), be respectively the location to which \( \alpha_i \) leads and the condition enabling it.
The Semaphore Rule (SEM):

Let \( \phi, \chi \) and \( \psi \) be state predicates such that:

A: \( \phi \) is \( (\text{at}\,\ell \land \chi) \)-invariant.

This means that as long as we stay at \( \ell \) and \( \chi \) is preserved, so is \( \phi \).

B: Any of the \( \alpha_i, i = 1, \ldots, k \), transitions of \( \Sigma \), which preserves \( \chi \) and is initiated with \( \phi \) true, achieves \( \psi \), i.e., \( \psi \) will hold after the transition. This is expressed by:

\[
[\text{at} \land c_i(\bar{y}) \land \phi(\bar{y}) \land \chi(\bar{y}) \land \chi(r_i(\bar{y}); f_i(\bar{y}))] \supset \psi(r_i(\bar{y}); f_i(\bar{y}))
\]

for every \( i = 1, \ldots, k \).

C: If \( \phi \land \chi \) holds permanently at \( \ell \) then eventually one of the \( c_i, \ i = 1, \ldots, k \), will be true. That is

\[
\equiv \Box(\text{at} \land \phi \land \chi) \supset \bigvee_{i=1}^k c_i.
\]

Then under these conditions we may conclude:

\[
\equiv (\text{at} \land \phi \land \Box \chi) \supset \Diamond \psi.
\]

That is, being at \( \ell \) with \( \phi \) true and being assured of the continuous holding of \( \chi \) guarantees the eventual realization of \( \psi \).

Note that condition C of SEM is weaker than condition C of ESC in that it does not require \( E_\ell = \bigvee_{i=1}^k c_i \) to be true whenever \( \text{at} \land \phi \land \chi \) holds but only requires it to be eventually realized. However, condition C here is a temporal statement and requires temporal reasoning for its justification, while condition C of ESC is static and requires only classical justification.

To justify this rule consider an execution which starts at \( \ell \) with \( \phi \) true and \( \chi \) continuously maintained. Condition A ensures that as long as we stay at \( \ell \), \( \phi \land \chi \) will be preserved. It is impossible that we stay at \( \ell \) forever because by condition C this would imply that \( E_\ell = \bigvee_{i=1}^k c_i \), which is the full exit condition of node \( \ell \), is enabled infinitely often while process \( P_j \) is never scheduled. By fairness we must have \( P_j \) scheduled at least once while \( E_\ell \) is true. This, by condition B and the permanence until this moment of \( \phi \land \text{at}\,\ell \land \chi \), will cause \( \psi \) to be realized.

It is important to realize the differences between a "semaphore location" and a "busy waiting" location. For comparison consider the following two simplified cases:

(a) Semaphore location:

(b) Busy waiting location:
(a) In the semaphore location case the fairness requirement demands that the scheduler will schedule this process at least once while its c condition is true provided the condition is true infinitely often. Thus for the SEM principle which is appropriate to this case we only require that c is realized infinitely often. This is exactly condition C which in this case is

\[ \Box (at \ell \land \phi \land \chi) \supset \Diamond c, \]

or is equivalently

\[ \Box (at \ell \land \phi \land \chi) \supset \Box \Diamond c. \]

(b) For the “busy waiting” situation, since the exit condition is \( c \lor \neg c = true \), the only obligation that the scheduler has is to eventually schedule this process. There is however nothing to prevent the process from being scheduled at exactly those instants in which \( c \) is false. Consequently, an infinitely often true \( c \) is not sufficient to ensure an exit to \( \ell' \). Instead we must require a stronger guarantee, that \( c \) be permanently true. Therefore, the corresponding condition \( C \) for the “busy waiting” situation for this case is

\[ (at \ell \land \phi \land \chi) \supset c, \]

which is equivalent to

\[ \Box (at \ell \land \phi \land \chi) \supset \Box \Diamond c. \]

That is, if staying forever at \( \ell \) guarantees the permanence of \( c \) then we will eventually exit from \( \ell \) to \( \ell' \). This can be derived from the ESC rule.

Since \( \Box c \supset \Diamond c \) we have the following robustness metatheorem:

A program that has been proven correct for an interpretation of its semaphores as “busy waiting” locations, is automatically correct for the implementation of these locations as true “semaphore” locations.

Consider, for example, the problem of accessibility of critical sections for the mutual exclusion program ME. In the roof to be given later we will reach the conclusion

\[ \Box \Box at \ell_5 \supset \Diamond \Box (y_1 \neq y_2), \]

where the instruction at \( \ell_5 \) is

\[ \ell_5: \text{loop while } y_1 = y_2. \]

Thus, this proof is sound for the interpretation of the loop primitive as “busy waiting”. By the robustness metatheorem any more efficient implementation of the loop primitive, in fact any implementation at all which is “just”, i.e., eventually schedules each process, will also cause the program to behave correctly.

The Single Path Rule

In this derived rule we repetitively apply the ESC rule to a chain of locations.
Let \( l_1, l_2, \ldots, l_{k+1} \) be a path of deterministic locations in \( P' \) with an immediate transition \( \alpha_i \) from every \( l_i \) to \( l_{i+1} \), \( i = 1, \ldots, k \).

\[
\begin{array}{c}
\begin{array}{c}
l_1 \xrightarrow{c_1(y) \rightarrow [y := f_1(y)]} \\
\quad \alpha_1
\end{array} \\
\end{array} \quad [y := f_k(y)] \rightarrow \alpha_k \rightarrow l_{k+1}
\]

### The Single Path Rule (SP):

Let \( \chi, \phi_1, \ldots, \phi_k, \) and \( \phi_{k+1} = \psi \) be predicates such that:

**A:** Each \( \phi_i \) is \((\text{at} \; l_i \land \chi)\) invariant, \( i = 1, \ldots, k \).
This means that as long as we stay at \( l_i \) and \( \chi \) is preserved so is \( \phi_i \).

**B:** Each transition \( \alpha_i, \quad i = 1, \ldots, k \), which preserves \( \chi \) and is initiated with \( \phi_i \) true achieves \( \phi_{i+1} \), that is

\[
\text{at} \; l_i \land \phi_i \land \chi \land \chi(r_i; f_i(y)) \models \phi_{i+1}(r_{i+1}; f_{i+1}(y)).
\]

**C:** \((\phi_i \land \chi)\) at \( l_i \) ensures that \( c_i \) is true, i.e.,

\[
\text{at} \; l_i \land \phi_i \land \chi \models c_i.
\]

Then under these three conditions we may conclude

\[
\forall Y_{k+1} \left( \text{at} \; l \land \phi_1 \land \square \chi \right) \models \diamond \psi.
\]

That is, if we start anywhere in the path with the appropriate \( \phi_i \) true and \( \chi \) continuously maintained we eventually wind up having \( \psi \).

This rule is obviously a generalization of ESC and is justified by a repeated application of ESC to \( l_1, \ldots, l_k \) (with \( Y_i = \{\alpha_i\} \)) respectively.

This rule can be somewhat generalized to a more general graph than a path. The SP principle also applies instead to a tree in which every node has an edge directed towards its ancestor.

This concludes the list of semantic proof rules reflecting the structure of the program and its influence on the possible execution sequences.

\[
\bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet
\]

In the following "formal" proofs of eventuality properties, we will intentionally omit manipulations which are pure temporal logic deductions, since we have not included an axiomatic system for temporal logic in this paper. Instead we will justify these deductions by saying "temporal reasoning" or "temporal deduction." The reader is invited to convince himself semantically that these deductions are indeed sound, that is, any sequence that satisfies the premises must also satisfy the consequence. Thus our proofs will consist, similarly to regular proofs, of a sequence of temporal formulas with a justification for each line in the sequence. A line in a proof may be justified in one of the following ways:
(a) If it is a valid first-order temporal logic formula.

(b) If it is an instance of one of the proof rules above.

(c) If it is a logical or temporal consequence of some preceding lines.

Given a deductive system for our logic (see [MAN2]) we will be able to justify steps of the form b and c using the axioms and rules of inference. Alternatively, c steps can be justified using a decision procedure for validity in (propositional) temporal logic ([BMP]). For our purpose of presenting proofs at a level which is not too formal, yet displays sufficient detail to be convincing, the style of semantic proofs seems most appropriate.

Note that our only reference to the program itself is through the proof principles ESC, ALT, SEM and SP⁺.

In presenting formal (semantic) proofs we will work our way gradually through examples that use only the ESC and SP⁺ rules first, then examples that use also the ALT rule and finally examples using semaphores and the corresponding SEM rule.

EXAMPLE: COUNTING TREE NODES

Consider first the use of eventuality chains in proving the total correctness of the sequential program TN for counting the nodes of a binary tree.

Program TN (Counting the nodes of a tree):

\[
S := (X), \quad C := 0
\]

\[\ell_0 : \text{if } S = () \text{ then } \text{goto } \ell_e\]

\[\ell_1 : (T, S) := (\text{hd}(S), \text{tl}(S))\]

\[\ell_2 : \text{if } T = A \text{ then } \text{goto } \ell_0\]

\[\ell_3 : C := C + 1\]

\[\ell_4 : S := f(T) \cdot r(T) \cdot S\]

\[\ell_5 : \text{goto } \ell_0\]

\[\ell_e : \text{halt.}\]

The program operates on a tree variable T and a variable S which is a stack of trees. The input variable X is a tree. The output is the value of the counter C. Each node in a tree may have zero, one or two descendants.

The available operations on trees are the functions \(f(T)\) and \(r(T)\) that yield the left and right subtrees of a tree \(T\) respectively. If the tree does not possess one of these subtrees the functions return the value \(A\).

The stack \(S\) is initialized to contain the tree \(X\). Taking the head and tail of a stack (functions \(hd\) and \(tl\) respectively) yields the top element and rest of the stack respectively. The operation in \(\ell_1\) pops the top of the stack into the variable \(T\). The operation at \(\ell_4\) pushes both the right subtree and the left subtree of \(T\) onto the top of the stack.

At any iteration of the program, the stack \(S\) contains the list of subtrees of \(X\) whose nodes have not yet been counted. Each iteration removes one such subtree from the stack. If it is the
empty subtree, $T = A$, we proceed to examine the next subtree on the stack. If it is not the empty subtree we add one to the counter $C$ and pushes the left and right subtrees of $T$ to the stack. When the stack is empty, $S = ()$, the program halts.

Denoting by $|X|$ the number of nodes in the tree $X$, the statement to be proved is formulated as

**Theorem:** $\forall A:\text{at} \ t_0 \supset \Diamond \left( \text{at} \ t_s \land C = |X| \right)$.

In order to prove the theorem we first prove a lemma:

**Lemma:** $\forall A:\left[\text{at} \ t_0 \land S = t \cdot s \land C = c\right] \supset \Diamond \left[\text{at} \ t_0 \land S = s \land C = c + |t|\right]$.

The lemma states that being at $t_0$ with a tree $t$ at the top of the stack $S$, we are assured of a later visit at $t_0$ where $t$ has been removed from the stack and its node count $|t|$ has been added to $C$.

Denote by $E(n)$ the statement:

$$E(n) : \forall t, s, c \left[\text{at} \ t_0 \land S = t \cdot s \land C = c \land |t| \leq n\right] \supset \Diamond \left[\text{at} \ t_0 \land S = s \land C = c + |t|\right].$$

This statement is the restriction of the lemma to trees with node count not exceeding $n$ for some natural number $n \geq 0$.

**Proof of Lemma:**

The lemma can then be stated as $\forall n. E(n)$; it is proved by induction. We have to show

(a) $E(0)$

(b) $E(n) \supset E(n + 1)$.

(a) Since $t \cdot s \neq ()$ and $|t| = 0 \supset t = A$ we may apply the SL rule to the path $t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow t_0$ and obtain

1. $\forall A:\left[\text{at} \ t_0 \land S = t \cdot s \land C = c \land |t| = 0\right] \supset \Diamond \left[\text{at} \ t_0 \land S = s \land C = c\right].$

This establishes $E(0)$.

(b) To show $E(n) \supset E(n + 1)$, consider an arbitrary $n, n \geq 0$, and assume

2. $E(n)$.

Then

3. $\forall A:\left[\text{at} \ t_0 \land S = t' \cdot s' \land C = c' \land |t'| = n + 1\right] \supset \Diamond \left[\text{at} \ t_0 \land S = t(t') \cdot r(t') \cdot s' \land C = c' + 1 \land |t'| = n + 1\right]$.

31
by the SP rule applied to the path $\ell_0 \rightarrow \ell_1 \rightarrow \ell_2 \rightarrow \ell_3 \rightarrow \ell_4 \rightarrow \ell_5 \rightarrow \ell_0$, using $|\ell'| = n + 1 \implies \ell' \neq A$.

We now use an instantiation of $E(n)$ with $t = \ell t', s = r t' \cdot s', \text{ and } e = e' + 1$ (which is justified since $|t| = |r t'| < n + 1$) to obtain

4. $\blacklozenge [\text{at } \ell_0 \wedge S = \ell t' \cdot r t' \cdot s' \wedge C = e' + 1] \supset \diamond [\text{at } \ell_0 \wedge S = r t' \cdot s' \wedge C = e' + 1 + |\ell t'|].$

By 3 and 4 we have

5. $\blacklozenge [\text{at } \ell_0 \wedge S = t' \cdot s' \wedge C = e' \wedge |\ell'| = n + 1] \supset \diamond [\text{at } \ell_0 \wedge S = r t' \cdot s' \wedge C = e' + 1 + |\ell t'| \wedge |\ell'| = n + 1].$

We now apply an instance of $E(n)$ again, this time with $t = r t', s = s'$, and $e = e' + 1 + |r t'|$ (which is justified since $|t| = |r t'| < n + 1$) to obtain

6. $\blacklozenge [\text{at } \ell_0 \wedge S = r t' \cdot s' \wedge C = e' + 1 + |\ell t'|] \supset \diamond [\text{at } \ell_0 \wedge S = s' \wedge C = e' + 1 + |r t'| + |\ell t'|].$

By 5 and 6 we have

7. $\blacklozenge [\text{at } \ell_0 \wedge S = t' \cdot s' \wedge C = e' \wedge |\ell'| = n + 1] \supset \diamond [\text{at } \ell_0 \wedge S = s' \wedge C = e' + 1 + |\ell t'| + |r t'|].$

Using the property $|t| > 0 \implies |t| = 1 + |\ell t'| + |r t'|$

we obtain:

8. $\blacklozenge [\text{at } \ell_0 \wedge S = t' \cdot s' \wedge C = e' \wedge |\ell'| = n + 1] \supset \diamond [\text{at } \ell_0 \wedge S = s' \wedge C = e' + |\ell t'|].$

Universally quantifying over the variables $t'$, $s'$ and $e'$ and then renaming them to $t$, $s$ and $e$, respectively, we obtain

9. $\forall t, s, e \{ [\text{at } \ell_0 \wedge S = t \cdot s \wedge C = e \wedge |t| = n + 1] \supset \diamond [\text{at } \ell_0 \wedge S = s \wedge C = e + |t|] \}.$

Line 9 holds under assumption 2 for every $n, n \geq 0$. Combined with 1 this gives

10. $E(n) \models E(n + 1)$.
Therefore, by the deduction theorem we have

11. \( E(n) \supset E(n + 1) \).

This concludes the proof of the lemma.

**Proof of Theorem:**

To prove the theorem we observe that

12. \( \models \varphi[\mu t \wedge S = (X) \wedge C = 0] \supset \Diamond [\mu t \wedge S = () \wedge C = |X|] \)

by the lemma with \( t = X \), \( s = () \), and \( c = 0 \). But

13. \( \models \varphi[\mu t \wedge S = () \wedge C = |X|] \supset \Diamond [\mu t \wedge C = |X|] \)

by SP applied to \( \ell_0 \rightarrow \ell_e \). Therefore, by combining 12 and 13, we have

14. \( \models \varphi[\mu t \wedge S = (X) \wedge C = 0] \supset \Diamond [\mu t \wedge C = |X|] \)

i.e.,

15. \( \models \Diamond [\mu t \wedge C = |X|] \).

One cannot fail to see the close resemblance between the temporal proof presented here and the informal intermittent-assertion proof conducted in [BUR] and [MW]. Our SP principle replaces the "little hand simulation" of [BUR].

**EXAMPLE: MUTUAL EXCLUSION (DEKKER) FORMAL PROOFS**

We will now present a formal proof of the accessibility proof of the program \( DK \). An informal proof of this was presented before and we advise the reader to refer to it while reading the following proof. The accessibility statement to be proved is

**Theorem:** \( \models \mu t \supset \Diamond \mu t \).

We will make use of the invariants derived before, namely:

\( \Box (Q_1 \wedge Q_2 \wedge Q_3 \wedge Q_4) \)

where

- \( Q_1 : (y_1 = T) \equiv \text{at}\{\ell_2, \ell_3, \ell_4, \ell_7, \ell_8\} \)
- \( Q_2 : (y_2 = T) \equiv \text{at}\{m_2, m_3, m_4, m_7, m_8\} \)
- \( Q_3 : \sim \text{at}\{\ell_7, \ell_8\} \vee \sim \text{at}\{m_7, m_8\} \)

33
and

\[ Q_4 : \quad \text{at}\{\ell_3, \ldots, \ell_6\} \land (t = 2) \supset \text{at}\{m_8, m_9, m_0\}. \]

\( Q_4 \) was proved by the standard invariance rule in Lemma 4 and will not be reproven here.

The proof of the theorem consists of a sequence of lemmas.

**Lemma A:** \( \forall [\text{at} \ell_2,3 \land (t = 1)] \supset \diamond \text{at} \ell_7 \)

**Proof of Lemma A:**

1. \( \forall [\text{at} \ell_2,3 \land (t = 1)] \supset \{\Box[\text{at} \ell_2,3 \land (t = 1)] \lor \diamond \text{at} \ell_7\} \)

by the \( \text{ALT} \) rule at \( \ell_2,3 \) where \( \phi \) is \( t = 1 \). Note that by \( t = 1 \), the \( \ell_3 \rightarrow \ell_4 \) transition is never possible.

2. \( \forall [\text{at} \ell_2,3 \land (t = 1) \land \text{at} m_5] \supset [\text{at} \ell_2,3 \land (t = 1) \land \text{at} m_5 \land (y_2 = F)] \)

by \( Q_2 \).

3. \( \forall [\text{at} \ell_2,3 \land (t = 1) \land \text{at} m_5 \land (y_2 = F)] \supset \diamond \text{at} \ell_7 \)

by \( SP \) applied to the path \( \ell_3 \rightarrow \ell_2 \rightarrow \ell_7 \) where \( \phi_3 = \phi_2 \) is \( (t = 1) \land \text{at} m_5 \land (y_2 = F) \) and \( \psi \) is \( \text{at} \ell_7 \).

4. \( \forall \{\Box[\text{at} \ell_2,3 \land (t = 1)] \land \text{at} m_5\} \supset \diamond \text{at} \ell_7 \)

is a temporal conclusion of 2 and 3.

This corresponds to case \( a \) of Lemma 4 in the informal proof.

Next we have

5. \( \forall [\Box[\text{at} \ell_2,3 \land (t = 1)] \supset \Box[\text{at} \ell_2,3 \land (t = 1) \land (y_1 = T)] \)

by \( Q_1 \).

6. \( \forall \{\Box[\text{at} \ell_2,3 \land (t = 1) \land (y_1 = T)] \land \text{at}\{m_1,4;m_6\}\} \supset \diamond \text{at} m_5 \)

by the \( SP \) rule applied to the path \( m_6 \rightarrow m_1 \rightarrow m_2 \rightarrow m_3 \rightarrow m_4 \rightarrow m_5 \) where \( \chi \) is \( \text{at} \ell_2,3 \land (t = 1) \land (y_1 = T) \).

7. \( \forall \{\Box[\text{at} \ell_2,3 \land (t = 1)] \land \text{at}\{m_1,4;m_6\}\} \supset \diamond \text{at} m_5 \)

by 5 and 6.

8. \( \forall [\Box[\text{at} \ell_2,3 \land (t = 1)] \land \text{at} m_{1,6}] \supset \diamond \text{at} \ell_7 \)

34
by 7 and 4.

This covers cases $b$, $c$, $d$, $e$, $f$ of the informal Lemma $A$.

We have

9. $\vdash atm_0 \supset [atm_0 \land (y_2 = F)]$

by $Q_2$.

10. $\vdash [atm_0 \land (y_2 = F)] \supset \{[atm_0 \land (y_2 = F)] \lor \Diamond atm_1\}$

by ALT at $m_0$ where $\phi$ is $y_2 = F$. Therefore

11. $\vdash atm_0 \supset [\Box(y_2 = F) \lor \Diamond atm_1]$

by 9 and 10.

12. $\vdash [\Box(y_2 = F) \land at \ell_{2,3} \land (t = 1)] \supset \Diamond at \ell_7$

by the SP rule applied to $\ell_3 \rightarrow \ell_2 \rightarrow \ell_7$ where $\phi_3 = \phi_2$ is $t = 1$ and $\chi$ is $y_2 = F$.

13. $\vdash \{[\Box at \ell_{2,3} \land (t = 1)] \land [\Box(y_2 = F)]\} \supset \Diamond at \ell_7$

is a consequence of 12. By taking the disjunction of 13 and 8 we get

14. $\vdash \{[\Box at \ell_{2,3} \land (t = 1)] \land ([\Box(y_2 = F) \lor atm_{1,8}]\} \supset \Diamond at \ell_7$

and then

15. $\vdash \{[\Box at \ell_{2,3} \land (t = 1)] \land atm_0\} \supset \Diamond atm_0$

is a consequence of 11 and 14.

This covers case $g$ of the informal Lemma $A$.

We also have

16. $\vdash \{[\Box at \ell_{2,3} \land (t = 1)] \land atm_{7,9}\} \supset \Diamond atm_0$

by the SP rule applied to the path $m_7 \rightarrow m_8 \rightarrow m_9 \rightarrow m_0$.

17. $\vdash \{[\Box at \ell_{2,3} \land (t = 1)] \land atm_{7,9}\} \supset \Diamond atm_0$

by 15 and 16.

This covers case $h$ of the proof.

Taking the disjunction of 8, 15 and 17 we obtain

18. $\vdash [\Box at \ell_{2,3} \land (t = 1)] \supset \Diamond atm_0$. 

35
Taking together 1 and 18 yields

17. $\models [at\ell_{3,3} \land (t = 1)] \supset \Diamond at\ell_{7}$

which is the result of Lemma A.

Lemma B is an invariance property $\models Q_4$ and is proved using the invariance principle.

Lemma C: $\models at\ell_5 \supset \Diamond at\ell_7$

Proof of Lemma C:

1. $\models at\ell_5 \supset \{\Box at\ell_5 \lor \Diamond [at\ell_6 \land (t = 1)]\}$

by the ALT rule at $\ell_5$.

2. $\models \Box (t = 2) \lor \Diamond (t = 1)$

is a temporal tautology using the obvious invariance $(t = 1) \lor (t = 2)$.

3. $\models \Box at\ell_5 \supset \{\Box [at\ell_5 \land (t = 2)] \lor \Diamond [at\ell_6 \land (t = 1)]\}$

is a temporal consequence of 2.

4. $\models [at\ell_5 \land (t = 1)] \supset \Diamond [at\ell_6 \land (t = 1)]$

by the ESC rule at $\ell_5$ where $\phi$ is $t = 1$.

5. $\models \Box at\ell_5 \supset \{\Box [at\ell_5 \land (t = 2)] \lor \Diamond [at\ell_6 \land (t = 1)]\}$

is a temporal consequence of 3 and 4.

6. $\models at\ell_5 \supset \{\Box [at\ell_5 \land (t = 2)] \lor \Diamond [at\ell_6 \land (t = 1)]\}$

by 1 and 5.

7. $\models \Box [at\ell_5 \land (t = 2)] \supset \Box [at\ell_5 \land (t = 2) \land (y_1 = F) \land atm_{1..7}]$

by $Q_1$ and $Q_4$.

We have

8. $\models \{\Box [at\ell_5 \land (t = 2)] \land atm_{7}\} \supset \Diamond [at\ell_5 \land (t = 1)]$

by the ESC rule at $m_7$ where $\chi$ is $at\ell_5 \land (t = 2)$, $\psi$ is $at\ell_5 \land (t = 1)$.

9. $\models \{\Box [at\ell_5 \land (t = 2)] \land atm_{7}\} \supset \Diamond [at\ell_6 \land (t = 1)]$
by 8 and 4.

This covers case a of the informal Lemma C.

Denoting

\[ \chi_0 : \text{at} \ell_5 \land (t_2 = 2) \land (y_1 = F) \land \text{atm}_{1..7} \]

we have

10. \[ \Box \chi_0 \land \text{at}\{m_{1,2}, m_{4..7}\} \supset \Diamond [\chi_0 \land \text{atm}_7] \]

by the SI' rule applied to the path \( m_4 \rightarrow m_5 \rightarrow m_6 \rightarrow m_1 \rightarrow m_2 \rightarrow m_7 \).

11. \[ \Box \chi_0 \land \text{at}\{m_{1,2}, m_{4..7}\} \supset \Diamond [\text{at} \ell_6 \land (t = 1)] \]

by 10 and 9.

This covers cases b, d, e, f, g of the informal Lemma C.

We have

12. \[ \Box \chi_0 \land \text{atm}_3 \supset \Diamond \text{atm}_2 \]

by the ESC rule at \( m_3 \). Thus

13. \[ \Box \chi_0 \land \text{atm}_3 \supset \Diamond [\text{at} \ell_6 \land (t = 1)] \]

by 11 and 12.

This covers case c of the informal Lemma C.

Taking the disjunction of 11 and 13 and noting that \( \chi_0 \supset \text{atm}_{1..7} \) we obtain

14. \[ \Box \chi_0 \supset \Diamond [\text{at} \ell_6 \land (t = 1)]. \]

Combined with 7 this gives

15. \[ \Box [\text{at} \ell_5 \land (t = 2)] \supset \Diamond [\text{at} \ell_6 \land (t = 1)]. \]

Combined with 6 we obtain

16. \[ \text{at} \ell_5 \supset \Diamond [\text{at} \ell_6 \land (t = 1)]. \]

Now we can derive

17. \[ [\text{at} \ell_{1,6} \land (t = 1)] \supset \Diamond [\text{at} \ell_{2,3} \land (t = 1)] \]

by the SI' rule applied to the path \( \ell_6 \rightarrow \ell_1 \rightarrow \ell_2 \) where \( \phi_6 = \phi_1 \) is \( t = 1 \), \( \psi \) is \( \text{at} \ell_{2,3} \land (t = 1) \). Using now Lemma A we obtain

18. \[ [\text{at} \ell_{1,6} \land (t = 1)] \supset \Diamond \text{at} \ell_7 \]

which together with 16 gives

19. \[ \text{at} \ell_5 \supset \Diamond \text{at} \ell_7. \]
Proof of Theorem:

Consider now the final proof of the theorem

1. $\Downarrow \mathsf{at}\ell_1 \supset \Diamond \mathsf{at}\ell_2$
   by ESC rule at $\ell_1$

2. $\Downarrow \mathsf{at}\ell_2 \supset \Diamond [\mathsf{at}\ell_1 \lor \mathsf{at}\ell_3]$
   by the ESC rule at $\ell_2$

3. $\Downarrow \mathsf{at}\ell_2 \supset [\Diamond \mathsf{at}\ell_1 \lor \Diamond \mathsf{at}\ell_3]$
   which is temporally equivalent to 2

4. $\Downarrow \mathsf{at}\ell_3 \supset \{\Diamond [\mathsf{at}\ell_2 \land (t = 1)] \lor \Diamond \mathsf{at}\ell_4\}$
   by the ESC rule at $\ell_3$

5. $\Downarrow [\mathsf{at}\ell_2 \land (t = 1)] \supset \Diamond \mathsf{at}\ell_7$
   by Lemma A

6. $\Downarrow \mathsf{at}\ell_4 \supset \Diamond \mathsf{at}\ell_5$
   by ESC rule at $\ell_4$

7. $\Downarrow \mathsf{at}\ell_4 \supset \Diamond \mathsf{at}\ell_7$
   by Lemma C and 6

8. $\Downarrow \mathsf{at}\ell_3 \supset \Diamond \mathsf{at}\ell_7$
   by 4, 5, and 7

9. $\Downarrow \mathsf{at}\ell_2 \supset \Diamond \mathsf{at}\ell_7$
   by 3 and 8

10. $\Downarrow \mathsf{at}\ell_1 \supset \Diamond \mathsf{at}\ell_7$
    by 1 and 9

This concludes the proof of the theorem.

EXAMPLE: CONSUMER PRODUCER

Consider next proving accessibility for the Consumer-Producer program (program CP). We assume that the computations at $\ell_0$ and at $\ell_7$ eventually terminate. The statement to be proved is:

Theorem: $\Downarrow \mathsf{at}\ell_0 \supset \Diamond \mathsf{at}\ell_3$

We will use in our proof the invariants which were established before

$\Downarrow \Box(Q_0 \land Q_1 \land Q_2)$

where

\[ Q_0: (c_f \geq 0) \land (c_e \geq 0) \land (s \geq 0) \]
\[ Q_1: \mathsf{at}\ell_{3.5} + \mathsf{at}m_{2.5} + s = 1 \]
\[ Q_2: c_f + c_e + \mathsf{at}\ell_{2.6} + \mathsf{at}m_{1.6} = N \]

Note that this is the first example that uses semaphores.
Assuming that the computation of $y_1$ at $t_0$ eventually terminates we may conclude

\[ \text{at}_{t_0} \supset \Box \text{at}_{t_1}. \]

The rest of the theorem is proved by two lemmas. Lemma A ensures that we get from $t_1$ to $t_2$ and Lemma B ensures that we get from $t_2$ to $t_3$.

**Lemma A:** \[ \text{at}_{t_1} \supset \Box \text{at}_{t_2} \]

**Proof of Lemma A:**

Since location $t_1$ contains a semaphore request instruction we will use the semaphore rule SEM to show that eventually $P_1$ will be granted access to $t_2$. The premise needed for the SEM rule is \( \Box \text{at}_{t_1} \supset \Box (ce > 0) \). An intuitive interpretation of this premise is that if we wait long enough at $t_1$, $ce$ will eventually turn positive. To show this, we give first an informal exposition inspecting the different locations in which $P_2$ may currently be.

**case a:** $P_2$ is at $m_6$. Then eventually it will execute the release($ce$) instruction to get $ce > 0$ as required.

**case b:** $P_2$ is at $m_2$, $m_3$, $m_4$ or $m_5$. Then it will eventually get to $m_6$ which by case a will cause $ce$ to turn positive.

**case c:** $P_2$ is at $m_1$. Then since $P_1$ is at $t_1$, $s = 1$ by $Q_1$. Since we assume that $P_1$ is waiting at $t_1$, $s$ will remain 1 as long as $P_2$ stays at $m_2$. By the semaphore axiom applied at $m_1$, $P_2$ will eventually proceed to $m_2$ and by case b, $ce$ will eventually turn positive.

**case d:** $P_2$ is at $m_0$. Then since $P_1$ is at $t_1$, $ef + ec = N > 0$ by $Q_2$. If $ce > 0$ we have proven our claim. Otherwise $ef > 0$ and will remain so as long as $P_2$ stays at $m_0$. Again by the semaphore axiom $P_2$ must eventually advance to $m_1$ and then by case c, $ce$ will eventually turn positive.

**case e:** $P_2$ is at $m_7$ or $m_8$. It will eventually get to $m_6$ and then by case d, $ce$ will eventually turn positive.

Let us now proceed with the more formal proof:

1. \[ \equiv [\Box \text{at}_{t_1} \land \text{at}_{m_6}] \supset [\Box \text{at}_{t_1} \land \text{at}_{m_6} \land (ce \geq 0)] \]
   by $Q_0$.

2. \[ \equiv [\Box \text{at}_{t_1} \land \text{at}_{m_6} \land (ce \geq 0)] \supset \Box (ce > 0) \]
   by ESC applied at $m_6$ where $\phi$ is $ce \geq 0$, $\chi$ is $at_{t_1}$, $\psi$ is $ce > 0$.

3. \[ \equiv [\Box \text{at}_{t_1} \land \text{at}_{m_6}] \supset \Box (ce > 0) \]
   is a conclusion of 1 and 2.

   This corresponds to case a above.

We have

4. \[ \equiv [\Box \text{at}_{t_1} \land \text{at}_{m_2\ldots 5}] \supset \Box \text{at}_{m_6} \]

39
by the SP rule applied to the path $m_2 \rightarrow m_3 \rightarrow m_4 \rightarrow m_5 \rightarrow m_6$.

5. $\models [\Box at\ell_1 \land atm_{2..5}] \supset \Diamond (ce > 0)$

is a conclusion of 4 and 3.

This covers case $b$ above.

We have

6. $\models [at\ell_1 \land atm_1] \supset (s = 1)$

by $Q_1$.

7. $\models [\Box at\ell_1 \land \Box atm_1] \supset \Diamond (s = 1)$

is a temporal consequence of 6.

8. $\models [\Box at\ell_1 \land atm_1] \supset \Diamond atm_2$

by the SEM rule at $m_1$ where $\chi$ is $at\ell_1$.

9. $\models [\Box at\ell_1 \land atm_1] \supset \Diamond (ce > 0)$

is a conclusion of 8 and 5.

This covers case $c$.

We have

10. $\models [\Box at\ell_1 \land atm_0] \supset [(cf > 0) \lor (ce > 0)]$

by $Q_2$.

11. $\models [\Box (at\ell_1 \land atm_0 \land (cf > 0))] \supset \Diamond (cf > 0)$

is a trivial temporal tautology.

12. $\models [\Box at\ell_1 \land atm_0 \land (cf > 0)] \supset \Diamond atm_1$

by the SEM rule at $m_0$, where $\phi$ is $cf > 0$, $\chi$ is $at\ell_1$.

13. $\models [\Box at\ell_1 \land atm_0 \land (cf > 0)] \supset \Diamond (ce > 0)$

is a conclusion of 12 and 9.

14. $\models [\Box at\ell_1 \land atm_0] \supset \Diamond (ce > 0)$

by a disjunction of 10 and 13.

This corresponds to case $d$. 

40
We have

15. \( \mathcal{P} \diamond [\Box \text{at}_{l_1} \land \text{atm}_{7,8}] \supset \diamond \text{atm}_0 \)

by the SP rule applied to the path \( l_7 \rightarrow l_8 \rightarrow l_0 \).

16. \( \mathcal{P} \diamond [\Box \text{at}_{l_1} \land \text{atm}_{7,8}] \supset (c > 0) \)

by 15 and 14.

This covers case e.

By taking the disjunction of 3, 5, 9, 14 and 16 we obtain

17. \( \mathcal{P} \diamond \Box (c > 0) \).

By applying the SEM rule at \( l_1 \) we obtain

18. \( \mathcal{P} \diamond \Box \text{at}_{l_2}. \)

Lemma B: \( \mathcal{P} \diamond \text{at}_{l_2} \supset \diamond \text{at}_{l_3} \)

Proof of Lemma B:

Here again we will apply the SEM rule, this time at \( l_2 \). The needed premise for its application is:

19. \( \mathcal{P} \diamond \Box (s > 0) \).

By inspecting the current location of \( P_2 \) we distinguish three cases:

case a: \( P_2 \) is at \( m_5 \). It will eventually advance to \( m_6 \) and turn \( s \) positive.

case b: \( P_2 \) is somewhere in \( \{m_2, m_3, m_4\} \). It will eventually get to \( m_5 \) and then by case a will turn \( s \) positive.

case c: \( P_2 \) is somewhere in \( \{m_0, m_1, m_7, m_8\} \). By \( Q_1 \), since \( P_1 \) is at \( l_1 \), \( s \) is currently equal to 1.

Thus the more formal proof is given by:

1. \( \mathcal{P} \Diamond [\Box \text{at}_{l_2} \land \text{atm}_5] \supset [\Box \text{at}_{l_2} \land \text{atm}_5 \land (s \geq 0)] \)

by \( Q_0 \).

2. \( \mathcal{P} \Diamond [\Box \text{at}_{l_2} \land \text{atm}_5 \land (s \geq 0)] \supset (s > 0) \)

by ESC applied at \( m_5 \) where \( \phi \) is \( s \geq 0 \), \( \chi \) is \( \text{at}_{l_2} \), \( \psi \) is \( s > 0 \)

3. \( \mathcal{P} \Diamond [\Box \text{at}_{l_2} \land \text{atm}_5] \supset (s > 0) \)
is a conclusion of 1 and 2.

This covers case a.

We have

4. \( \vDash [\Box \alpha \land \alpha] \supset (s > 0) \)

by the SP rule applied to the path \( m_2 \rightarrow m_3 \rightarrow m_4 \rightarrow m_5 \).

5. \( \vDash [\Box \alpha \land \alpha] \supset (s > 0) \)

by 4 and 3.

This covers case b.

We have

6. \( \vDash [\Box \alpha] \supset (s = 1) \)

by \( Q_1 \).

7. \( \vDash [\Box \alpha \land \neg \alpha] \supset (s > 0) \)

by 6.

This covers case c.

By taking the disjunction of 3, 5, and 7 we obtain

8. \( \vDash [\Box \alpha] \supset (s > 0) \).

Applying the SEM rule at \( \ell_2 \) yields

9. \( \vDash \alpha \land \alpha \).

which is the desired Lemma B.

EXAMPLE: BINOMIAL COEFFICIENT

We will now establish the termination of the program BC for the distributed evaluation of a binomial coefficient. Since we have already proved the partial correctness of this program, termination will guarantee total correctness.

The statement to be proved is:

**Theorem:** \( \vDash (\alpha \land \alpha) \)

The initial condition associated with the proper computation of the program is

\[ \alpha \land \alpha \land (y_1 = n) \land (y_2 = 0) \land (y_3 = 1) \land (y_4 = 1) \land (0 \leq k \leq n). \]
We will use in our proof the following invariants that were established above:

\[
\Box (Q_0 \land Q_1 \land Q_2),
\]

where

\[
\begin{align*}
Q_0 & \text{ is } at_{l,4} + atm_{x,6} + y_4 = 1 \\
Q_1 & \text{ is } ((n - k) \leq y_1 \leq n) \land (0 \leq y_2 \leq k) \\
Q_2 & \text{ is } atm \supset (y_1 = n - k).
\end{align*}
\]

We start by proving a sequence of lemmas:

**Lemma A1:** \(\Box (atl_1 \land (y_1 = u)) \supset \Box [atl_2 \land (y_1 = u)]\)

This lemma ensures that we never get stuck at \(l_1\), which is a semaphore instruction.

**Proof of Lemma A1:**

The proof distinguishes three cases according to the current location of \(P_2\). In all cases we assume that \(P_1\) is waiting at \(l_1\).

**case a:** \(P_2\) is at \(m_6\). The next time it will be scheduled will increment \(y_4\), making it positive.

**case b:** \(P_2\) is in \[m_4, m_5\}. Eventually it will get to \(m_6\) and increment \(y_4\).

**case c:** \(P_2\) is in \(\{m_0, m_1, m_2, m_3, m_7, m_e\}\). By \(Q_0\) and the fact that \(P_1\) is at \(l_1\), \(y_4\) is currently positive.

In all three cases we can show that the value of \(y_1\) never changes.

Thus we have:

1. \(\Box \Box at_{l_1} \land atm_{m_6} \supset \Box at_{l_1} \land atm_{m_6} \land (y_4 \geq 0)\)

by \(Q_0\).

2. \(\Box \Box at_{l_1} \land atm_{m_6} \land (y_4 \geq 0) \supset \Diamond (y_4 > 0)\)

by the ESC rule at \(m_6\) where \(\phi\) is \(y_4 \geq 0\), \(\chi\) is \(at_{l_1}\).

3. \(\Box \Box at_{l_1} \land atm_{m_6} \supset \Diamond (y_4 > 0)\)

by 2 and 1.

This covers case a.

We have

4. \(\Box \Box at_{l_1} \land atm_{x,6} \supset \Diamond atm_{m_6}\)
by the SI' rule applied to the path $m_4 \rightarrow m_5 \rightarrow m_6$.

5. $\models [\Box \text{at} l_1 \land \text{at} m_{4,5}] \supset \Diamond (y_4 > 0)$

by 4 and 3.

This covers case b.

We have

6. $\models [\Box \text{at} l_1 \land \lnot \text{at} m_{4,6}] \supset (y_4 > 0)$

by $Q_0$. Therefore

7. $\models [\Box \text{at} l_1 \land \lnot \text{at} m_{4,6}] \supset \Diamond (y_4 > 0)$

This covers case c.

By taking the disjunction of 3, 5 and 7 we obtain

8. $\models \Box \text{at} l_1 \supset \Diamond (y_4 > 0)$.

Applying the SEM rule at $l_1$ where $\phi$ is $y_1 = u$ we obtain

9. $\models [\text{at} l_1 \land (y_1 = u)] \supset \Diamond [\text{at} l_2 \land (y_1 = u)]$. 

Lemma A2: $\models \{[\text{at} l_{1,5} \land (y_1 = u + 1)] \lor [\text{at} l_6 \land (y_1 = u)]\} \supset \Diamond [\text{at} l_0 \land (y_1 = u)]$

This lemma ensures that being anywhere in $l_1$ to $l_5$ we return to $l_0$ with the value of $y_1$ smaller by 1 than the original and being at $l_6$ we return to $l_0$ with the value of $y_1$ unchanged.

Proof of Lemma A2:

After being ensured by Lemma A1 of not being blocked at $l_1$ all that remains is to trace the value of $y_1$. Indeed:

1. $\models [\text{at} l_1 \land (y_1 = u + 1)] \supset \Diamond [\text{at} l_2 \land (y_1 = u + 1)]$

by Lemma A1.

2. $\models \{[\text{at} l_{2,5} \land (y_1 = u + 1)] \lor [\text{at} l_6 \land (y_1 = u)]\} \supset \Diamond [\text{at} l_0 \land (y_1 = u)]$

by applying the SI' rule to the path $l_2 \rightarrow l_3 \rightarrow l_5 \rightarrow l_6 \rightarrow l_0$ where $\phi_2 = \phi_3 = \phi_4 = \phi_5$ is $y_1 = (u + 1)$, $\phi_6$ is $y_1 = u$, and $\psi$ is $at l_0 \land (y_1 = u)$.

3. $\models [\text{at} l_1 \land (y_1 = u + 1)] \supset \Diamond [\text{at} l_0 \land (y_1 = u)]$

by 1 and 2.

4. $\models \{[\text{at} l_{1,5} \land (y_1 = u + 1)] \lor [\text{at} l_6 \land (y_1 = u)]\} \supset \Diamond [\text{at} l_0 \land (y_1 = u)]$
by 2 and 3.

This establishes Lemma \( A2 \). \( \Box \)

\textbf{Lemma A3:} \( [\text{at}\ell_0 \land (y_1 \geq n - k)] \supset [\text{at}\ell_e \land (y_1 = n - k)]. \)

This lemma establishes the termination of \( P_1 \) if started at \( \ell_0 \) with \( y_1 \geq n - k \).

\textbf{Proof of Lemma A3:}

Define the auxiliary assertion:

\[
E_1(u) : [\text{at}\ell_0 \land (y_1 = u)] \supset [\text{at}\ell_e \land (y_1 = n - k)].
\]

We will establish the lemma by showing that

\( (u \geq n - k) \supset E_1(u) \).

This will be established by induction on \( u \geq n - k \). We will have to show first

\( (a) \quad E_1(n - k) \)

and then

\( (b) \quad [(u \geq n - k) \land E_1(u)] \supset E_1(u + 1). \)

\( (a) \) To prove part a we observe that \( E_1(n - k) \) just says that if we are at \( \ell_0 \) with \( y_1 = n - k \) we will eventually get to \( \ell_e \) with \( y_1 = n - k \). This is obvious since when \( y_1 = n - k \), \( P_1 \) proceeds directly from \( \ell_0 \) to \( \ell_e \). Indeed:

1. \( \Rightarrow [\text{at}\ell_0 \land (y_1 = n - k)] \supset [\text{at}\ell_e \land (y_1 = n - k)] \)

by the ESC rule applied at \( \ell_0 \) where \( \phi \) is \( y_1 = n - k \) considering just the exit \( \ell_0 \rightarrow \ell_e \) whose enabling condition \( c \) is \( y_1 = n - k \). In other words,

1'. \( E_1(n - k) \)

\( (b) \) To prove part b we assume that \( u \geq n - k \) and \( E_1(u) \) is true and consider an execution that starts at \( \ell_0 \) with \( y_1 = u + 1 \). Since \( u + 1 \geq n - k \) we will proceed to \( \ell_1 \) with \( y_1 = u + 1 \). By Lemma A2 we will return to \( \ell_0 \) with \( y_1 = u \). Now by the assumption of \( E_1(u) \) we will eventually get to \( \ell_e \) with \( y_1 = n - k \).

For the formal proof, we assume:

2. \( \Rightarrow u \geq n - k \)

and

3. \( \Rightarrow E_1(u), \)

i.e.,

3'. \( [\text{at}\ell_0 \land (y_1 = u)] \supset [\text{at}\ell_e \land (y_1 = n - k)]. \)
Then

4. \[ \text{Ilat}_0 \prec [\text{atl}_0 \land (y_1 = u + 1)] \iff [\text{atl}_0 \land (y_1 = u + 1) \land (y_1 > n - k)] \]

by 2.

5. \[ \text{Ilat}_0 \prec [\text{atl}_0 \land (y_1 = u + 1) \land (y_1 > n - k)] \iff \Diamond [\text{atl}_1 \land (y_1 = u + 1)] \]

by the ESC rule at \( l_0 \) using only the \( l_0 \rightarrow l_1 \) exit where \( \phi \) is \( y_1 > n - k \).

6. \[ \text{Ilat}_0 \prec [\text{atl}_0 \land (y_1 = u + 1)] \iff [\text{atl}_1 \land (y_1 = u + 1)] \]

by 4 and 5.

7. \[ \text{Ilat}_0 \prec [\text{atl}_0 \land (y_1 = u + 1)] \iff \Diamond [\text{atl}_0 \land (y_1 = u)] \]

by 6 and Lemma A2.

8. \[ \text{Ilat}_0 \prec [\text{atl}_0 \land (y_1 = u + 1)] \iff \Diamond [\text{atl}_e \land (y_1 = n - k)] \]

by 7 and 3', i.e., by the definition of \( E_1 \),

8'. \[ E_1(u + 1) \]

Applying the deduction theorem to 2, 3, and 8', we obtain

9. \[ (u \geq n - k) \iff [E_1(u) \circ E_1(u + 1)]. \]

Now we may combine parts a and b (i.e., 1' and 9) to deduce the lemma using the induction principle.

Lemma A4: \[ \Diamond [\text{atl}_e \land (y_1 = n - k)] \]

This states that no matter where we are in a properly initialized execution of the program, we will eventually wind up at \( l_e \) with \( y_1 = n - k \).

Proof of Lemma A4:

There are three cases to be considered according to the current location of \( P_1 \).

case a: \( P_1 \) is already at \( l_e \). Then we have by \( Q_2 \) that \( y_1 = n - k \).

case b: \( P_1 \) is at \( l_0 \). Then we are assured by \( Q_1 \) that \( y_1 \geq n - k \); hence, by Lemma A3, we will wind up at \( l_e \) with \( y_1 = (n - k) \).

case c: \( P_1 \) is anywhere else, that is in \( \{l_1, \ldots, l_6\} \). Then we will eventually get to \( l_0 \) by Lemma A2, which is already covered by case b.

We proceed with the formal proof. We have

1. \[ \text{atl}_e \circ [\text{atl}_e \land (y_1 = n - k)] \]

46
by $Q_2$.
This corresponds to case $a$.

We have
$$2. \quad \begin{array}{l}
\text{at} \ell_0 \supset [\text{at} \ell_0 \land (y_1 \geq n - k)]
\end{array}$$
by $Q_1$.

$$3. \quad \begin{array}{l}
\text{at} \ell_0 \supset \diamond [\text{at} \ell \land (y_1 = n - k)]
\end{array}$$
by Lemma $A3$.
This covers case $b$.

We have
$$4. \quad \begin{array}{l}
\text{at} \ell_{1..6} \supset \diamond \text{at} \ell_0
\end{array}$$
by Lemma $A2$.

$$5. \quad \begin{array}{l}
\text{at} \ell_{1..6} \supset \diamond [\text{at} \ell \land (y_1 = n - k)]
\end{array}$$
by 4 and 3.
This covers case $c$.

Taking the disjunction of 1, 3 and 5 we obtain
$$6. \quad \begin{array}{l}
\diamond [\text{at} \ell \land (y_1 = n - k)]
\end{array}$$
which establishes the lemma. $lacksquare$

We now turn to the termination of $P_2$.

Lemma $B0$: $\begin{array}{l}
\begin{array}{l}
\text{at} m_2 \land (y_2 = u) \supset \diamond [\text{at} m_3 \land (y_2 = u)]
\end{array}
\end{array}$
This lemma states that we can never get blocked at $m_2$.

Proof of Lemma $B0$:

By Lemma $A4$ we are guaranteed that $P_1$ will eventually get to $\ell_6$ with $y_1 = n - k$. In the worst case, by the time $P_1$ gets to $\ell_6$, $P_2$ is still waiting at $m_2$. But then by $Q_1$, $y_2 \leq k$ and $y_1 + y_2 \leq n$ which enables the exit condition and leaves it enabled until $P_2$ moves. This proof should not be considered as saying that $P_2$ will indeed wait at $m_2$ until $P_1$ terminates, but this approach provides the easiest proof.

Proceeding with more formal proof we have
$$1. \quad \begin{array}{l}
\text{at} m_2 \land (y_2 = u) \supset \{\square [\text{at} m_2 \land (y_2 = u)] \lor \diamond [\text{at} m_3 \land (y_2 = u)]\}
\end{array}$$
by the ALT rule at $m_2$ where $\phi$ is $y_2 = u$.

2. $\Box [atm_2 \land (y_2 = u)] \supset \Diamond [atm_2 \land (y_2 = u) \land atm_e \land (y_1 = n - k)]$

by Lemma A1.

3. $\Box [atm_2 \land (y_2 = u) \land atm_e \land (y_1 = n - k)] \supset [atm_2 \land (y_2 = u) \land atm_e \land (y_1 + y_2 \leq n)]$

using $y_2 \leq k$ given by $Q_1$.

4. $\Box [atm_2 \land (y_2 = u) \land atm_e \land (y_1 + y_2 \leq n)] \supset \Diamond [atm_3 \land (y_2 = u)]$

by ESC at $m_2$ considering only the exit $m_2 \rightarrow m_3$ where $\phi$ is $(y_2 = u) \land atm_e \land (y_1 + y_2 \leq n)$.

5. $\Box [atm_2 \land (y_2 = u)] \supset \Diamond [atm_3 \land (y_2 = u)]$

by 2, 3, and 4.

6. $\Box [atm_2 \land (y_2 = u)] \supset \Diamond [atm_3 \land (y_2 = u)]$

by 1 and 5.

Lemma B1: $\Box [atm_3 \land (y_2 = u)] \supset \Diamond [atm_4 \land (y_2 = u)]$

This lemma states that $P_2$ does not get blocked at $m_3$ but eventually proceeds to $m_4$ with an unchanged value of $y_2$.

It is analogous to Lemma A1 and has a very similar proof. In that proof we distinguish three cases according to the location of $P_1$. They are: $P_1$ at $\ell_4$, $P_2$ in $\{\ell_2, \ell_3\}$, and $P_2$ elsewhere. Their analysis is identical to that of Lemma A1.

Lemma B2: $\Box \{ [atm_1 \land (y_2 = u)] \lor [atm_{2.7} \land (y_2 = u + 1)] \} \supset \Diamond [atm_0 \land (y_2 = u + 1)]$

This lemma states that if we are anywhere in $m_1$ to $m_7$ we will eventually return to $m_0$ with $y_2$ properly adjusted.

proof of Lemma B2:

1. $\Box [atm_{4.7} \land (y_2 = u + 1)] \supset \Diamond [atm_0 \land (y_2 = u + 1)]$

by the SP rule applied to the path $m_4 \rightarrow m_5 \rightarrow m_6 \rightarrow m_7 \rightarrow m_0$ where $\phi_4 = \phi_5 = \phi_6 = \phi_7$ is $y_2 = u + 1$ and $\psi$ is $atm_0 \land (y_2 = u + 1)$.

2. $\Box [atm_3 \land (y_2 = u + 1)] \supset \Diamond [atm_0 \land (y_2 = u + 1)]$

by Lemma B1 and 1.

3. $\Box [atm_2 \land (y_2 = u + 1)] \supset \Diamond [atm_0 \land (y_2 = u + 1)]$
by Lemma B0 and 2.

4. \( \models [\text{atm}_1 \land (y_2 = u)] \supset \Diamond [\text{atm}_2 \land (y_2 = u + 1)] \)

by the ESC rule at \( m_1 \) where \( \phi \) is \( y_2 = u \) and \( \psi \) is \( \text{atm}_2 \land (y_2 = u + 1) \).

5. \( \models [\text{atm}_1 \land (y_2 = u)] \supset \Diamond [\text{atm}_0 \land (y_2 = u + 1)] \)

by 4 and 3.

By taking the disjunction of 1, 2, 3 and 5 we obtain:

6. \( \models \{[\text{atm}_1 \land (y_2 = u)] \lor [\text{atm}_{2, \ldots} \land (y_2 = u + 1)]\} \supset \Diamond [\text{atm}_0 \land (y_2 = u + 1)]. \)

**Lemma B3:** \( \models [\text{atm}_0 \land (y_2 \leq k)] \supset \Diamond [\text{atm}_x \land (y = k)] \)

This lemma establishes the termination of \( P_2 \) if started at \( m_0 \) with \( y_2 \leq k \).

**Proof of Lemma B3:**

Similarly to the proof of Lemma A3 we define the auxiliary assertion

\[
E_2(u) : \quad [\text{atm}_0 \land (y_2 = u)] \supset \Diamond [\text{atm}_x \land (y_2 = k)].
\]

The lemma is established by showing that

\( \models (u \leq k) \supset E_2(u). \)

Analogously to A3 this is proven by descending induction on \( u \leq k \). We show the two clauses:

(a) \( \models E_2(k) \)

and

(b) \( \models [(u < k) \land E_2(u + 1)] \supset E_2(u). \)

Part a is proved by observing the direct path from \( m_0 \) to \( m_x \) in the case that \( y_2 = k \). Part b is proved by tracing the execution from \( m_0 \) with \( y_2 = u < k \) to \( m_1 \) with \( y_2 = v + 1 \) and use the induction hypothesis to finally guarantee \( \text{atm}_x \land (y_2 = k) \).

The details of the formal proof are very similar to those of A3.

**Lemma B4:** \( \models \Diamond \text{atm}_x \)

This statement says that regardless of where we are in a properly initialized execution of the program, we eventually wind up at \( m_x \).

**Proof of Lemma B4:**

Similarly to the proof of Lemma A4 there are three cases to be considered:
case a: \( P_2 \) already at \( m_e \).

case b: \( P_2 \) currently at \( m_0 \). Then we have by \( Q_1 \) that \( y_2 \leq k \) and hence by Lemma B3 we will eventually reach \( m_e \).

case c: \( P_2 \) is elsewhere. Then we will eventually get to \( m_0 \) by Lemma B2.

The formal details are similar to those of Lemma A4.

Proof of Theorem:

To conclude the proof of the theorem we observe that:

1. \( \models \ell_e \supset \Box\text{at} \ell_e \)

by the ALT rule since \( \ell_e \) has no exits.

2. \( \models \Diamond \Box\text{at} \ell_e \)

by Lemma A4 and 1. Similarly,

3. \( \models \Diamond \Box\text{at} m_e \)

using Lemma B4 and the ALT rule at \( m_e \).

A temporal consequence of 2 and 3 is

\( \models \Diamond [\text{at} \ell_e \land \text{at} m_e] \).

Acknowledgements

We thankfully acknowledge the help extended to us by Yoni Malachi, Pierre Wolper, Frank Yellin, Joe Weening, and Rivi Zarhi in reading the earlier drafts of the manuscript. Special thanks are due to Evelyn Eldridge-Diaz for TEXing the manuscript.

REFERENCES


