1985

Eliminating Proofs of Interference-freedom from Levin-Gries CSP Program Proofs

Thomas P. Murtagh

Report Number:
85-525

Eliminating Proofs of Interference-freedom from Levin-Gries CSP Program Proofs

Thomas P. Murtagh

July 9, 1985

CSD - TR - 525

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

ABSTRACT

The proof system for Hoare's CSP language proposed by Levin and Gries requires that for each predicate used in the proof of a process interference-freedom proofs be given for each command that can be executed in parallel with the code of the process. In the worse case, the effort required to provide such proofs could be enormous. To address this problem, Levin and Gries suggest the use of *synchronously altered varialbes* and *universal assertions*. In a recent paper, Prasad claimed that the Levin-Gries system could be modified in a manner that eliminated the need for many interference-freedom proofs. Unfortunately, the system he proposed was incorrect. In this paper we propose a system that provides a similar reduction in the number of proofs of non-interference required and prove that it is equivalent to the Levin-Gries system.

In a recent paper [4], Prasad presented a new proof system for Hoare's CSP language[1]. His system was based on an earlier system proposed by Levin and Gries [2], but improved on their system by reducing the amount of work required to prove non-interference between the proof of one process and the activities of other processes. Prasad's proof system has several technical flaws. Fortunately, however, his intuition that much of the effort required for non-interference proofs in the Levin-Gries system is unessential was correct. In this paper we will discuss the problems with Prasad's system and present an approach that corrects these problems.

Our discussion will be divided into 3 sections. In the next section, we will review the nature of the Levin-Gries system emphasizing the role of interference-freedom proofs. In section 2, we will discuss Prasad's system and explain one of its flaws. In section 3 we will present our alternative to his system.

1. The Levin-Gries CSP Proof System

In the proof system for CSP proposed by Levin and Gries, one can prove a statement of the form:

$$\{ P_1 \& \cdots \& P_n \}\ [A_1::S_1;\ \ldots\ A_n::S_n]\ \{ Q_1 \& \cdots \& Q_n \}$$

by finding proofs of the form

$$\{ P_i \}\ S_i\ \{ Q_i \}$$

for all i that have properties which Levin and Gries call satisfaction and interference-freedom.

Satisfaction involves assumptions made about inter-process communication in the proofs. The axioms for the communication statements of CSP used in the Levin-Gries system are

$\{P\}\ A!T(\bar{e})\ \{Q\}$

and

$\{P\}\ A?T(\bar{x})\ \{Q\}$

where P and Q can be any predicates. The intention is that Q be used to state assumptions about the overall state of the program when communication occurs that are needed to prove correctness of individual processes. The validity of these assumptions is ensured by requiring that the proofs given for all processes involved must satisfy the *Satisfaction rule* which states that for any pair of communications statements of the form:

$[B :: \ldots \{P\}\ A?T(\bar{x})\ \{Q\} \ \ldots \parallel A :: \ldots \{R\}\ B!T(\bar{e})\ \{S\} \ \ldots\ ]$

it must be the case that

$(P \ \& \ R) \Rightarrow (Q\&S)^{\bar{x}}_{\bar{e}}$

where for any predicate P, $P^{\bar{x}}_{\bar{e}}$ denotes the predicate obtained by simultaneously replacing all free occurrences of variables in the list $\bar{x}$ by the corresponding expressions in the list $\bar{e}$.

Interference-freedom involves the use of auxiliary variables in proofs [3]. Interference between processes in CSP is not possible since the language does not allow any variable changed in one process to be referenced or modified in any other process. The Levin-Gries system, however, allows one to add assignments to auxiliary variables to a program for the purpose of proving it. An auxiliary variables may be referenced and modified in several processes. As a result, assignments and input commands involving auxiliary variables that appear in one process may interfere with predicates used in the proof of other processes.

Levin and Gries define a command S to be parallel to an assertion P if S is contained in some process of a parallel command and P is contained in a different process of the same parallel command. In addition, they say that a matching pair of communication com-

mand's S and R is parallel to P if both S and R are parallel to P. To complete the proof of a parallel command, one must prove

{P & pre(S)} S {P}

for every assertion P in the proof of the individual processes and for every command S parallel to a given P, where pre(S) denotes the pre-condition of S in the proof, and also prove that

(P & pre(S) & pre(R)) ==> $P^{\bar{x}}_{\bar{e}}$

for every matching communication pair, S : A!T($\bar{e}$) and R : B?T($\bar{x}$), that is parallel to P.

The work involved in proving interference-freedom in a proof could be enormous. Essentially, a separate proof is required for each pair of statements in distinct processes. Fortunately, most of these proofs are trivial. For example, if S is an assignment statement to a variable that does not appear in P, then

{P} S {P}

is a trivial instance of the axiom of assignment and

{P & pre(S)} S {P}

follow immediately from the rule of consequence. Local variables of one process cannot occur in the predicates used in the proof of a different process in the same parallel command. Accordingly, all non-interference proofs involving assignments to local variables will be trivial and can be justifiably skipped. There are, however, still a significant number of potentially non-trivial non-interference proofs. Any assignment to a non-local variable could potentially interfere with the assertions inserted between the statements of any other process in its parallel command.

2. Prasad's Approach

The goal of Prasad's work is to provide a proof system for CSP which requires fewer non-interference proofs. His approach was to present a proof system for a language whose semantics differs from that of CSP in ways that make the correctness of his proof system for that language obvious and then argue that the semantic differences do not effect program behavior in any way that is perceptible through the proof system. Accordingly, he concludes that his system is a valid proof system for CSP. Unfortunately, his argument is flawed. The normal semantics of CSP allow execution sequences that produce final states that are impossible under his semantics.

The essential difference between his semantics and the normal semantics of CSP is that in his version *all* processes must synchronize whenever information is passed between processes. This has two implications. First, it is no longer possible for two processes to exchange a message while other processes execute assignment statements or other non-communication code in parallel. When a process attempts to send or receive a message it must be suspended until all other processes are suspended. Once all processes are ready to communicate, they may all be allowed to proceed. Second, in an annotated program, assignments to auxiliary variables must be treated as communication code, since the changes made by such assignments are visible in all processes. Thus, whenever a process attempts to execute an auxiliary assignment or transmit or receive a message, it must be suspended until all processes are terminated or ready to perform one of these operations.

The advantage of this semantics is that it implies that interference is impossible. In the Levin-Gries system, auxiliary assignments and message transfers were the only operations that could interfere with predicates in other processes. In Prasad's semantics, these actions can no longer interfere with the post-conditions of statements other than auxiliary

assignments and message transfers, because they can only occur when all processes are involved in some form of communication. As a result, by modifying the Levin-Gries satisfaction rule slightly and applying it to auxiliary assignment statements in addition to communication primitives, one eliminates the need to do any non-interference proofs.

Unfortunately, Prasad's claim that his semantics are indistinguishable from the standard semantics as far as the proof system is concerned is false. There does not appear to be any problem with making processes wait for all other processes to wait before communicating, but Prasad requires all processes that can communicate to communicate once all processes are waiting, and this does cause troubles.

For example, consider the program skeleton shown in Fig. 1. Under the normal

```
[   P1::
        P4!e1;
        ...
        P3!e2
        ...
||  P2::
        P3!e3
        ...
||  P3::
        [P1?x -> S1   []
         P2?x -> S2
        ]
        ...
||  P4::
        P1?y
        ...
]
```

Figure 1.

semantics for CSP, either S1 or S2 could be executed in process P3. Under Prasad's model, however, only S2 could be executed. This is because his model requires that the communication between P1 and P4 be delayed until P2 and P3 are also ready to communicate and that all four processes communicate when this occurs. Accordingly, the guard in P3 which accepts input from P2 must be selected and the communication completed before P1 can communicate with P3. Any attempt to provide complete proof rules for a language with the semantics he describes would have to account for the behavior of the example in Fig. 1. In particular, if S1 was

x := 4

and S2 was

x := 5

one would expect to be able to prove that

x = 5

held after the alternative construct, even though such a statement would be false in normal CSP.

It appears that this problem could be corrected in a straightforward manner. The problem, as mentioned above, is that processes are forced to proceed once all processes reach a synchronization point in Prasad's semantics. Suppose we change his semantics by relaxing this requirement. That is, we will still insist that processes can only execute assignments to auxiliary variables and communication commands when all processes are waiting to do so, but we will assume that once all processes are waiting, the system nondeterministically decides whether or not to execute each assignment and each matching pair of communication commands. If the system decides not to execute a command in a given process, then that process remains suspended until all processes are again waiting at

synchronization points.

Prasad's satisfaction rule would then be replaced by the requirement that given sequential proofs of the processes in a parallel command

$[P1::S_1 \parallel \ldots \parallel Pn::S_n]$

for every pair of set of synchronization commands, W and E, such that:

(1) E contains only assignments to auxiliary variables and matching pairs of communication commands,

(2) $E \subseteq W$, in particular, we assume $E = \{c_1, \ldots, c_m\}$ and $W = \{c_1, \ldots, c_n\}$,

(3) each c_i appears in a distinct branch of the parallel command with the annotation

$\{p_i\}\ c_i\ \{q_i\}$

and

(4) $\bar{x}$ is the list of variables modified by executing the commands in E and $\bar{e}$ is the set of expressions whose values will be stored in these variables

we prove the implication

$$p_1 \& \cdots \& p_n \Longrightarrow (q_1 \& \cdots \& q_m \& p_{m+1} \& \cdots \& p_n)^{\bar{x}}_{\bar{e}}$$

In this rule, each W represents a set of points where all processes could be ready to execute synchronization commands. Each E represents a subset of these commands that the system might decide to execute. The implication whose proof is required ensures that for each such E and W, the the postconditions of all commands in E will hold after the synchronization commands are completed and that the execution of the commands in E will not interfere with the preconditions of the commands in W-E.

3. An Alternative Approach to Reducing Non-interference Proofs

While the new satisfaction rule presented in the preceding section and the semantics associated with it appear to fix the problem that we observed in Prasad's system, we will see that they do not provide a sound proof system for normal CSP. The problem is subtle and it suggests that it will be difficult to have confidence in the validity of any variation of Prasad's system without some more formal approach to the problem of verifying its correctness.

There are four distinct semantic descriptions of CSP invoived in Prasad's work: 1) the informal (English) semantics for normal CSP, 2) the informal semantics for his non-standard version of CSP, 3) the Levin-Gries axiomatic semantics for normal CSP and 4) Prasad's axiomatic semantics for the non-standard version of CSP. The goal is to establish that all four are equivalent – in particular that some modification of Prasad's axiomatic semantics is equivalent to the informal semantics for normal CSP.

As Fig. 2 suggests, there are two approaches that could be used to establish this. Prasad attempted to argue that a) his axiomatic semantics was correct for his non-standard

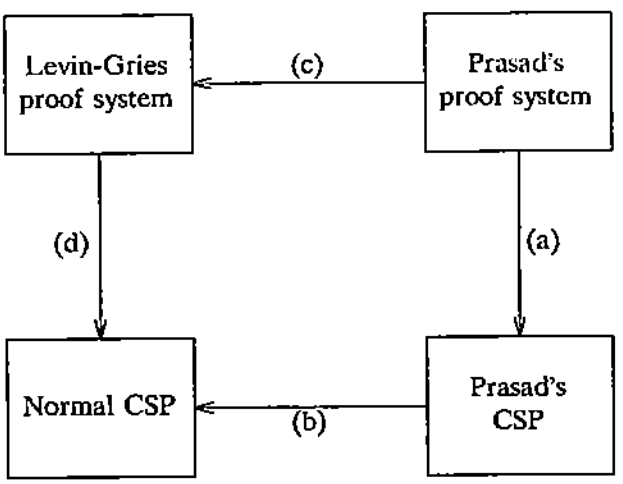

Figure 2.

informal semantics and b) that his non-standard informal semantics was equivalent to the normal semantics for purposes of program proof. An alternative approach would be to show c) that the new axiomatic semantics was equivalent to the Levin-Gries axiomatic semantics and d) that the Levin-Gries system is correct. This second approach would have two advantages. First, given the simplicity of the system involved and the fact that the Levin-Gries system is well accepted, it seems safer to assume (d) then (a). Second, since (c) involves two formal systems we can attempt to give a formal proof of their equivalence, while any argument to establish (b) must be informal. While we have not succeeded in using this second approach to establish the validity of our correction to Prasad's system, we have used it to show the validity of another variation of the Levin-Gries system that provides almost the same reduction in the effort required to show non-interference as Prasad's system.

The intuition behind our system is simple. If a statement, S, does not alter global auxiliary variables, then it cannot effect any information that a distinct process could interfere with. If S is annotated as

{P} S {Q}

and we have shown that Q is interference-free, P need not contain any additional information that could be interfered with. Accordingly, if it is impossible to prove P interference-free it must be because in the proof we included some facts in P that were not needed to conclude Q. That is, we should expect that in the case that P cannot be proven to be interference free, one could rewrite the proof replacing P with a new predicate P' such that interference-freedom can be shown for P' and P ==> P'.

In fact, we can prove the following:

Theorem: Given a proof

{P} S {Q}

for a CSP program in the Levin-Gries system in which interference freedom has been shown only for Q, the invariants of all loops, the postconditions of each component of all parallel command, and the pre-conditions of all communication commands and assignments to auxiliary variables, we can mechanically construct a proof

{P'} S {Q}

such that P ==> P' is provable and every assertion used in the proof is provably interference free.

[]

This allows us to conclude that the Levin-Gries system can be replaced by a system in which proofs of interference-freedom are only required for loop invariants, the post-conditions of processes in parallel commands, and the pre-conditions of communication commands and auxiliary assignment statements.

Like many proposals for program proving systems, the Levin-Gries system ignores the question of exactly what proof system should be used to prove predicates (as opposed to Hoare triples). We do not wish to specify a particular proof system for reasoning about predicates, but for our work we need to make some assumptions about the power of the proof system used. In particular, we assume that:

A1) If a predicate P is provable, then P^x_e is provable for any expression e.

A2) For any predicate P, we assume

P => P

is provable.

A3) For any predicates P,Q and R, if P => Q and Q => R are provable then P => R is provable.

A4) If the predicates $P_1 \Rightarrow Q_1$, $P_2 \Rightarrow Q_2$, ... $P_n \Rightarrow Q_n$ are provable, then we assume that the predicate

$$P_1 \& \ldots \& P_n \Rightarrow Q_1 \& \ldots \& Q_n$$

is provable.

A5) For any predicates P, Q, and R, if P => Q is provable then

$$(R \Rightarrow P) \Rightarrow (R \Rightarrow Q)$$

is provable.

A6) For any predicates P,Q and R,

$$((P \Rightarrow Q) \& R) \Rightarrow (P \Rightarrow (Q \& R))$$

is provable.

A7) For any predicates $P_1, \cdots, P_n$ and $Q_1, \cdots, Q_n$

$$(((P_1 \Rightarrow Q_1) \& \cdots \& (P_n \Rightarrow Q_n)) \& P_i) \Rightarrow Q_i$$

is provable.

A8) For any predicates $P_1, \cdots, P_n$, $Q_1, \cdots, Q_n$ and R if

$$R \& P_i \Rightarrow Q_i$$

is provable for each i, then

$$R \Rightarrow ((P_1 \Rightarrow Q_1) \& \cdots \& (P_n \Rightarrow Q_n))$$

is provable.

Given these assumption, the theorem can be proved by induction over the depth of nesting of control structures in S and the length of S.

First, assuming no control structures in S, we must prove the theorem for arbitrary lists of assignment statements and communication commands. If S consists of a single communication command or auxiliary assignment, then P = P' satisfies the conditions of the theorem trivially. If S is a single assignment

$$x := e$$

where x is a local variable, then for some predicate R it must be provable that

$$P \Longrightarrow R_e^x$$

and

$$R \Longrightarrow Q$$

Since

$$\{P\}\ x := e\ \{Q\}$$

was either proved using a single application of the axiom of assignment in the original proof -- in which case R = Q, P = R_e^x and the implications are provable by assumption A2 -- or the original proof involved the axiom of assignment and the rule of consequence – in which case R and the proofs of the implications can be found in the original proof. Now, by assumptions A1 and A3,

$$R_e^x \Longrightarrow Q_e^x$$

and

$$P \Rightarrow Q_e^x$$

must be provable. By the axiom of assignment,

$$\{Q_e^x\}\ x := e\ \{Q\}$$

is also provable. Accordingly, Q_e^x is a possible candidate for P'. To see that it is P' we

must show that we could prove interference-freedom. Since we assume that interference-freedom has been shown for Q, for each assignment statement

a := e'

parallel to Q,

{Q & pre(a := e') } a := e' {Q}

must be provable. As above, however, this implies that for some R,

(Q & pre(a := e')) ==> $R^{a}_{e'}$

and

R ==> Q

must be provable. Since x is a local variable, it cannot appear in e' or in pre(a := e') and a cannot appear in e. Accordingly,

$(R^{x})^{a}_{e'} = (R^{a}_{e'})^{x}$

and

$pre(a:=e')^{x}_{e} = pre(a := e')$

These facts and assumption A1 allow us to conclude that

(Q^{x}_{e} & pre(x := e')) ==> $(R^{x})^{a}_{e'}$

and

R^{x}_{e} ==> Q^{x}_{e}

are provable. In this case, since

{ $(R^{x})^{a}_{e'}$ } a =: e' { R^{x}_{e}}

is an instance of the axiom of assignment we can prove

$\{Q_e^x \ \& \ pre(x := e')\}\ a := e'\ \{ Q_e^x \}$

using the rule of consequence.

Similarly, for any matching communication pair, S : A!T($\vec{e}$) and R : B?T($\vec{a}$), parallel to Q_e^x we can assume that

$(Q \ \& \ pre(S) \ \& \ pre(R)) \ ==> Q_{\vec{e}}^{\vec{a}}$

is provable. Since x cannot occur in either of the preconditions or in $\vec{e}$ and none of the variables in $\vec{a}$ can appear in e, applying A1 allows us to conclude that

$(Q_e^x \ \& \ pre(S) \ \& \ pre(R)) \ ==> (Q_e^x)_{\vec{e}}^{\vec{a}}$

is also provable. Thus, Q_e^x can be proven free from interference.

Now, if we assume that the theorem holds for sequences of less than n simple commands, we can easily show that it holds for n. Given that we have proved

$\{P\}\ S\ \{Q\}$

and proved interference-freedom for those assertions in this proof required by the statement of the theorem, we know that at some point in this proof we must have shown that

$\{P\}\ S_1\ \{R\}$

and

$\{R\}\ S_2\ \{Q\}$

for some R, S_1, and S_2 such that $S = S_1;S_2$. By our inductive assumption, we can construct a proof of

$\{R'\}\ S_2\ \{Q\}$

such that R ==> R' and all assertions in the proof are provably interference-free. Using the rule of consequence we can prove

$\{P\}\ S_1\ \{R'\}$

and then our inductive assumption implies that this proof can be rewritten yielding a proof of

$\{P'\}\ S_1\ \{R'\}$

such that $P \Longrightarrow P'$ and all assertions in the proof are provably interference-free. Finally, applying the rule of composition (which Levin and Gries refer to as the rule of sequence) yields a proof of

$\{P'\}\ S\ \{Q\}$

satisfying the conditions of the theorem. Note that this argument did not depend on the absence of control structures in S. Accordingly, to complete the proof of the theorem we need only show that it holds for any single structured statement.

So, assume that the theorem holds for any sequence of statements in which control structures are nested less than n levels deep and that we have a proof

$\{P\}\ S\ \{Q\}$

in which S is a single statement in which structured statements are nested n levels deep and interference freedom has only been proven for those assertions in the proof required by our theorem. We need to consider three cases: S is an alternative command, S is a repetitive command or S is a parallel command.

If S is an alternative command of the form

```
if b1; c1 --> S1
[] b2; c2 --> S2
 .
 .
 .
[] bn; cn --> Sn
fi
```

then the original proof must contain a proof of the statement

$\{P \;\&\; b_i\}\ c_i;\ S_i\ \{Q\}$

for each i. Since S_i may not have control structures nested to a depth of n or greater, we may assume that we can construct proofs of the form:

$\{P_i\}\ c_i;\ S_i\ \{Q\}$

in which all assertions are provably interference-free and such that

$P \;\&\; b_i \;\Longrightarrow P_i$

The assumption (A7) that

$(((b_1 \Rightarrow P_1) \;\&\; \cdots \;\&\; (b_n \Rightarrow P_n)) \;\&\; b_i) \Rightarrow P_i$

is provable for each i implies we can use the rule of consequence to prove assertions of the form

$\{((b_1 \Rightarrow P_1) \;\&\; \cdots \;\&\; (b_n \Rightarrow P_n)) \;\&\; b_i\}\ c_i;\ S_i\ \{Q\}$

and this allows us to conclude

$\{(b_1 \Longrightarrow P_1) \;\&\; \ldots \;\&\; (b_n \Longrightarrow P_n)\}$ if ... fi $\{Q\}$

We know that each P_i is interference free and we can therefore conclude that for any parallel assignment command

a := e'

and each P_i there is some predicate R such that

$P_i \;\&\; pre(a := e') \;\Longrightarrow R_e^a$

and

$R \;\Longrightarrow P_i$

Accordingly, by assumption A5 we can assume that the predicates

$(b_i \Rightarrow (P_i \,\&\, pre(a := e'))) \Longrightarrow (b_i \Rightarrow R^a_{e'})$

and

$(b_i \Rightarrow R) \Longrightarrow (b_i \Rightarrow P_i)$

are provable. Since b_i is used in a guard in a process distinct from that containing the assignment a := e', a cannot appear in b_i. Therefore,

$\{(b_i \Rightarrow R^a_{e'})\}\ a := e'\ \{b_i \Rightarrow R\}$

is an instance of the axiom of assignment and we can conclude that

$\{(b_i \Rightarrow P_i) \,\&\, pre(a := e')\}\ a := e'\ \{b_i \Rightarrow P_i\}$

using A6 and the rule of consequence. Thus, for each i, the predicate $(b_i \Rightarrow P_i)$ is interference free. In addition, for any set of predicates T_1, ..., T_n all of which are provably interference free, T_1 & ... & T_n must also be provably interference free. This can be shown by an argument similar to those given above. Finally, we assume (A8) that

$P \Rightarrow ((b_1 \Rightarrow P_1) \,\&\, \cdots \,\&\, (b_n \Rightarrow P_n))$

is provable. Thus, $(b_1 \Longrightarrow P_1)$ & ... & $(b_n \Longrightarrow P_n)$ is the desired P'.

If S is a repetitive command of the form

```
do b1; c1 --> S1
[] b2; c2 --> S2
 .
 .
 .
[] bn; cn --> Sn
od
```

then there must be some I such that

$P \Longrightarrow I$

$I \,\&\, \neg b_1 \,\&\, \ldots \,\&\, \neg b_n \Longrightarrow Q$

and

$\{I \& b_i\}\ c_i;S_i\ \{I\}$

for each i. By the requirements of the theorem, I must be provably interference free. Accordingly, by the inductive assumption, for each i there is some I'_i such that $(I \& b_i) => I'_i$ and we can construct a proof of

$\{I'_i\}\ c_i;S_i\ \{I\}$

in which all assertions are provably interference free. Since b_i can only contain references to variables local to its process, b_i and $I \& b_i$ must be interference free. Accordingly, the rule of consequence allows us to extend the proof of

$\{I'_i\}\ c_i;S_i\ \{I\}$

to obtain a proof of

$\{I \& b_i\}\ c_i;S_i\ \{I\}$

in which all assertions are provably interference free and then use the rule for the repetitive command and the rule of consequence to prove

$\{I\}\ S\ \{Q\}$

Thus, I is the desired P'.

Finally, if S is the parallel command

$[\ A_1::S_1 \parallel A_2::S_2 \parallel \ldots \parallel A_n::S_n\ ]$

there must be predicates $P_1, \ldots, P_n$ and $Q_1, \ldots, Q_n$ such that

$P \Rightarrow P_1 \,\&\, \ldots \,\&\, P_n$

$Q_1 \,\&\, \ldots \,\&\, Q_n \Rightarrow Q$

and

$\{P_i\}\ S_i\ \{Q_i\}$

for each i. By our inductive assumption, for each i we can construct a proof of

$\{P'_i\}\ S_i\ \{Q_i\}$

such that all assertions used in the proof are provably interference free and $P_i \Rightarrow P'_i$ for each i. The inference rule for parallel commands and the rule of consequence then allow us to prove

$\{P'_1 \,\&\, \ldots \,\&\, P'_n\}\ S\ \{Q\}$

Since each P_i is provably interference free their conjunction must also be provably interference free. In addition, by assumptions A3 and A4,

$P \Rightarrow (P'_1 \,\&\, \ldots \,\&\, P'_n)$

Accordingly, $P'_1 \,\&\, \ldots \,\&\, P'_n$ is the desired P'. This concludes the proof and establishes the fact that the proof system we have suggested is equivalent to the Levin-Gries system.

In the system we have described, proofs of interference-freedom are required for the pre-conditions of synchronization point and for loop invariants. It is also possible to prove the correctness of a system in which proofs of interference-freedom are required for the post-conditions of synchronization points. It is somewhat easier to prove the correctness of the system we have presented, because we can work backwards when constructing a proof in which all assertions are interference free. When one attempts to prove the other system, one finds that it is difficult to work forwards in a system with a backward assignment axiom. In the Levin-Gries system, one can use the introduction of auxiliary variables to

simulate a forward assignment rule. In particular, the key to proving the correctness of a proof system in which interference-freedom need only be shown for the post-conditions of synchronization points is the proof of the following:

Lemma: Given a sequential proof of

$$\{ P \}\ x := e\ \{ Q \}$$

where x is a local variable and a P is provably interference-free, one can construct a sequential proof of

$$\{ P \}\ m := x; \{ T \}\ x := e\ \{ Q' \}$$

where m is an auxiliary variable distinct from all others used in the proof being developed, T and Q' are provably interference-free, and

$$Q' \Rightarrow Q$$

□

It can be shown that the predicates

$$P^x_m \ \&\ e = e^x_m$$

and

$$P^x_m \ \&\ x = e^x_m$$

are the required T and Q'.

4. Discussion

There are three significant differences between our system and that proposed by Prasad. The first two involve the work required to prove non-interference. The third involves the use of assumptions after auxiliary commands.

The first difference is that Prasad only requires proofs of interference-freedom for predicates before synchronization points (i.e. communication commands, assignments to auxiliary variables and the ends of processes in parallel commands) while we also require them for the invariants of loops. This difference relates to the technical details of our approach to showing that our system is correct. Consider the simple case of a loop

```
do b1 --> S1
[] b2 --> S2
 .
 .
 .
[] bn --> Sn
od
```

containing no synchronization points. If we have a proofs of

$$\{I \;\&\; b_i\}\ S_i\ \{I\}$$

and

$$I \;\&\; \sim b_1 \;\&\; \ldots \;\&\; \sim b_n \Longrightarrow Q$$

such that Q is provably interference free, one would expect that I should also be interference free, unless the author of the proof unnecessarily included information about global variables in I that were not needed. So, one should expect that there is some I' that is provably interference free such that

$$\{I' \;\&\; b_i\}\ S_i\ \{I'\}$$

and

$$I' \;\&\; \sim b_1 \;\&\; \ldots \;\&\; \sim b_n \Longrightarrow Q$$

In our approach, however, we have not merely argued that interference free predicates exist, we have shown how to construct them. It is not, unfortunately, obvious, how to construct the desired I'. We suspect that one can in fact prove that I' must exist under

assumptions similar to those used to prove completeness of sequential proof systems, but we have not yet pursued this question.

The second difference is that our system, like the original Levin-Gries system, requires interference proofs for all pairs of assertions and parallel commands and satisfaction proofs for matching pairs of communication commands, while Prasad's system requires that the proofs be done for groups of processes containing one synchronization point from each component of a parallel command. We suspect that this difference may involve another flaw in Prasad's system. The problem involves nested parallel commands. Consider the program skeleton shown in Fig. 3.

```
[ A1 :: [ B1 :: ... B2!T( ē ) ... ‖
          B2 :: ... B1?T( x̄ ) ...
        ]
  A2 :: ... {P} S ; ...
]
```

Figure 3.

In proving satisfaction (in Prasad's sense) for the outer parallel command we must consider all pairs of synchronization points consisting of one member from A1 and one from A2. Accordingly, while either B2!T($\bar{e}$) or B1?T($\bar{x}$) might be the synchronization point chosen from A1, they will never be considered together. Accordingly, it appears to be the case that Prasad's rule will not account for the possibility that a change made to an auxiliary variable included in $\bar{x}$ might interfere with P in A2.

The final difference between our system and Prasad's is that we only change the requirements for proving interference freedom. His system, on the other hand, also changes the satisfaction rule in a way that allows one to make assumptions after assignments to auxiliary variables. We suspect that his intuition that such a change is possible is correct, but we have not yet investigated the possibility of proving so using our approach.

References

1. Hoare, C. A. R., Communicating Sequential Processes, *Communications of the ACM 21*, 8 (August 1978), 666-677.
2. Levin, G. M. and D. Gries, Proof Techniques for Communicating Sequential Processes, *Acta Informatica 15*, (1981), 281-302.
3. Owicki, S. and D. Gries, An Axiomatic Proof Technique for Parallel Programs, *Acta Informatica 6*, (1976), 319-340.
4. Prasad, V. R., Interference-freedom in Proofs of CSP Programs, *Proceedings of the 4th IEEE International Conference on Distributed Computing Systems*, San Francisco, May 1984, 79-86.