SIAM J. COMPUT.
Vol. 6, No. 4, December 1977

PROSPECTS AND LIMITATIONS OF AUTOMATIC ASSERTION GENERATION FOR LOOP PROGRAMS*

JAYADEV MISRA†

Abstract. The problem of generation of loop invariants from the input, output assertions of a loop program (**while** B **do** S) is considered. The problem is theoretically unsolvable in general. As a special case we consider assertions of the form $x\,R\,y$, where R denotes a binary relation, x denotes the variables manipulated by the program and y denotes variables that are not modified by the program. We derive conditions for R such that if any loop program has $x\,R\,y$ as the input and output assertions, then $x\,R\,y$ is a loop invariant. These conditions for R are shown to be necessary and sufficient in that if some R' does not meet these conditions, then there are loop programs for which $x\,R'\,y$ holds at entrance and exit, though not following every iteration. In particular it is shown that if R is an equivalence relation, then under certain reasonable restrictions on the loop, $x\,R\,y$ holds at entrance and exit of the loop if and only if it holds after every iteration.

Key words. assertion, loop program, verification

1. Introduction. One of the major difficulties in mechanical program proving is to generate suitable assertions for a given program, given its input, output specifications. In theory, the problem is unsolvable. The most difficult aspect of assertion generation (by humans or algorithms) is in locating a "loop invariant" [4] for every loop. For a loop of the form {**while** B **do** S}, a loop invariant is a proposition P such that $P \wedge B\ \{S\}\ P$.[1] It then follows that if P is true on entrance to the loop, it can be asserted to be true on exit. In order to show that a proposition Q_2 is true at exit given that a proposition Q_1 is true on entry, it is sufficient to locate a proposition P such that (i) $Q_1 \Rightarrow P$ (ii) $P \wedge B\ \{S\}\ P$ (iii) $P \wedge \neg B \Rightarrow Q_2$.

An important problem in mechanical program verification is to obtain a loop invariant P as above, given Q_1, Q_2. Several heuristic techniques have been reported [5], [8], [10]. Recently an interesting scheme called "subgoal induction" [9] has been introduced which seems to be effective in a large number of cases.

Due to lack of suitable general techniques, it is important to characterize certain classes of input/output propositions for which the invariant may be obtained algorithmically. Such a characterization is interesting if it includes most of the commonly occurring forms of propositions which arise in actual programs. The present paper is a step in that direction.

We will consider loops of the form {**while** $B(x)$ **do** $S(x)$}: x denotes the set of variables on which the loop operates; x has an initial value on entry to the loop. The value of x is modified by the loop body. Let $x\,R\,y$ denote that x is related to y under R. A loop *preserves* a relation R if

$$x\,R\,y\ \{\textbf{while } B(x) \textbf{ do } S(x)\}\ x\,R\,y.$$

* Received by the editors July 22, 1975, and in final revised form February 2, 1977. A preliminary version of some of these results appears in [6].

† Department of Computer Sciences, University of Texas at Austin, Austin, Texas 78712. This work was supported by the National Science Foundation under Grants GJ36424 and DCR75-09842.

[1] Using the notation introduced by Hoare [4].

y denotes some variables that are not modified by the loop. Verbally, the loop *preserves* the relation R if $x\,R\,y$ true on entrance to the loop implies it remains true on exit from the loop with the modified x, assuming termination. For instance, $x > y$ {**while** $B(x)$ **do** $x := x+1$} $x > y$, for any B. A loop *uniformly preserves* a relation R if $x\,R\,y$ is a loop invariant; i.e., if

$$x\,R\,y \wedge B(x)\,\{S(x)\}\,x\,R\,y.$$

Clearly, if R is uniformly preserved, then R is preserved by the loop. The converse however is not true.

In this paper, we characterize the class of relations R having the property that if R is preserved by *any* loop, then it is uniformly preserved. Clearly, the characterization conditions are trivial for any specific loop **while** $B(x)$ **do** $S(x)$, namely $x\,R\,y \wedge B(x)$ $\{S(x)\}$ $x\,R\,y$; the definition itself. Our interest in studying such characterization is to prove/disprove $x\,R\,y$ {**while** $B(x)$ **do** $S(x)$} $x\,R\,y$, when R meets the given characterization, by proving/disproving that $x\,R\,y$ is a loop invariant.

We will use the notion of *closure* introduced in [1]. We will define a class of relations called generalized equivalence relations (GE relation) and show that if R is a GE relation and is preserved by any loop having closure, then it is uniformly preserved. Conversely, if R is not a GE relation, then there exists a loop that preserves R, but does not preserve it uniformly. Any equivalence relation is shown to be a GE relation. Conditions for proving/disproving that a loop computes a certain function can be derived from this characterization.

This paper generalizes the results in [1]. However, a knowledge of that paper is not necessary to follow the results presented here. Implications of these results in automatic program verification are discussed.

2. Some preliminary notions. We will be working with loops of the form {**while** B **do** S}. We need to make explicit mention of variables on which the program operates. Consider the following schema, which we call $W(B, S)$.

```
begin
    declaration for variables $t$; {This is optional}
    while $B$ do $S$
end;
```

We adopt the following conventions about $W(B, S)$.

(i) $W(B, S)$ accepts input in certain global variables. The set of global variables will usually be denoted by x. Let x_0 denote the initial values of x before entry into the loop.

(ii) The variables t as defined above are called local variables of $W(B, S)$. Local variables initially have undefined values. A local variable gets a value when it is assigned one during computation. For the rest of the paper, global and local refer to global and local variables x, t of $W(B, S)$ respectively.

(iii) B is a predicate over some or all global variables. The rationale for such a requirement is that local variables have undefined values on entry and certain clauses in B may otherwise be undefined as a result.

(iv) The program $W(B, S)$ does not terminate if it ever accesses (examines/uses) a variable having an undefined value.

(v)The output of $W(B, S)$ appears in global variables x. Thus, the effect of execution of the loop is to modify the values of x.

Local variables t of $W(B, S)$ are indeed in a certain sense global to "**while** B **do** S". However, we believe that the distinction between local and global variables is important considering (i), (ii), (iv) and (v). Furthermore, local variables t of $W(B, S)$ are different from any local variable that S may have; during iterations t may retain values from iteration to iteration whereas local variables of S have undefined values at the beginning of every iteration.

Our treatment of variables is general. We are not specifically interested in the kind of data that a variable may represent: one variable may represent a tree, another may represent a file segment etc. We require that the variable values be drawn from a prespecified domain, but there is no restriction on the domain itself.

The next notion is fundamental; it was introduced in [1]. ("Domain" refers to the set of initial variable values of interest.)

Definition 1. A domain D is *closed* with respect to $W(B, S)$ if and only if $x \in D$ is a loop invariant, i.e.

$$x \in D \wedge B(x) \{S(x)\} x \in D.$$

Observation. If D is closed, then starting with any initial value $x_0 \in D$, the variable values after every iteration must be from D. If the loop terminates, the final values are from D.

The importance of closure was demonstrated in [1], [7] where it was shown that a knowledge of closure is essential in locating a suitable loop invariant. We will assume closure of the input for the rest of the paper.

Example 1.

```
while $v \neq 0$ do
begin
    $u := u+1; v := v-1$
end;
```

Let

$$D = \{(u, v) | u, v \text{ integer}; v \geqq 0\},$$

$$D' = \{(u, v) | u, v \text{ integer}; v \geqq 30\},$$

$$D'' = \{(u, v) | u, v \text{ integer}; u \geqq 30\},$$

D, D'' are closed with respect to the given program. D' is not closed since with $(u, v) = (5, 30) \in D'$, we obtain $(6, 29)$ after one iteration, which is not in D'. □

Depending on the context $S(x)$ would either denote that S uses variables x (as in $x \, R \, y \, \{S(x)\} \, x \, R \, y$) or the value computed by S when x denotes the initial values of its variables (as in $S(x) \, R \, y$).

Definition 2. A set D is *range inclusive* with respect to a function F if for every $a \in D$, $F(a)$ is defined and $F(a) \in D$.

Definition 3. $W(B, S)$ *computes* a function F over a domain D if

(i) D is range inclusive with respect to F

and

(ii) for every input $x_0 \in D$, $W(B, S)$ halts and produces $F(x_0)$ as the output (in the global variables).

The following theorem is the basis of the results appearing in the next section. It is from [1], [7].

THEOREM 1′ (see [1], [7]). *Suppose D is closed with respect to $W(B, S)$. Let D be range inclusive with respect to a given function F. $W(B, S)$ computes F over D if and only if all of the following conditions hold.*

1) *$W(B, S)$ terminates for every input from D.*
2) $(x \in D \wedge \neg B(x)) \Rightarrow (F(x) = x)$.
3) *$[F(x) = F(y)]$ is a loop invariant for $W(B, S)$.*

Furthermore, conditions 1), 2) *and* 3) *are mutually independent.*

Condition 3) is the important invariant condition. It states that if F is the function computed by $W(B, S)$, then $F(x)$ remains identical during successive iterations with modified values of x in each iteration. Clearly, $F(x)$ must be defined for every such x generated during iterations: this is guaranteed by the requirement of closure on D.

The following are the significant aspects of Theorem 1′:

(i) the conditions in the theorem are necessary and sufficient. Thus proving/disproving these conditions proves/disproves the claim. This is in contrast to many assertion generation systems which provide only sufficient conditions.

(ii) The form of the invariant is independent of B and S.

3. Relations uniformly preserved by a loop. Conditions 1), 2) and 3) in the statement of Theorem 1′ may be labeled as termination, boundary and iteration conditions. The boundary condition is easy to derive (and usually simple to prove) by considering the exit conditions. The iteration condition is the one that leads to the loop invariant which captures the "dynamics" of the loop. In this paper, we are primarily interested in the generation of the iteration condition.

The major contribution of this paper is a generalization of Theorem 1′. We characterize the class of binary relations which are preserved by any loop if and only if they are uniformly preserved.

The motivation behind this extension is twofold. First, we often want to prove a certain relationship between input and output of a loop without knowing the exact functional relationship. For instance, we may want to show that the output value is larger in magnitude than the input or that the output array is a permutation of the input array, etc. Secondly, we hope to establish a theoretical limitation on what kinds of loop invariants can be generated without examining the loop body.

Clearly, if R is uniformly preserved then it is preserved. However, the converse is not true, even for transitive relations (such as $\leqq$ on integers), as shown in the following example.

Example 2. Let $W(B, S)$ be the following program.

```
while $v \neq 1$ do
    if odd($v$) then $v := v + 1$
                else $v := v/2$;
```

$D = \{v | v \geqq 1$ and v integer$\}$. Let the relation R be defined as follows:

$$v\,R\,u \Leftrightarrow v \leqq u.$$

Clearly,

$$v\,R\,u\,\{W(B,S)\}\,v\,R\,u.$$

However, $v\,R\,u$ is not a loop invariant (as can be seen with $v = 3$ and $u = 3$). □

We first derive the conditions on R, dependent on B and independent of S. Next, we remove the dependence on B. We are thus given

(i) a binary relation R on a domain D;

(ii) that D is closed with respect to $W(B, S)$ and $W(B, S)$ terminates for every input from D;

(iii) and that $W(B, S)$ preserves the relation R. We ask for the necessary and sufficient conditions, independent of S, under which $x\,R\,y$ is a loop invariant.

DEFINITION 4. Given a domain D and a binary relation R on D, $(a, b \in D)a \geqq b$ if and only if $(\forall\, c \in D)(b\,R\,c \Rightarrow a\,R\,c)$; $a \equiv b$ if and only if $a \geqq b$ and $b \geqq a$. Note that, if $(\nexists\, c \in D, b\,R\,c)$, $(\forall\, a \in D)(a \geqq b)$. $\geqq$ will be called the *derived relation* of R.

Observation. For any R, the derived relation $\geqq$ is reflexive and transitive and $\equiv$ is an equivalence relation.

Notation. $\neg(x\,R\,y)$ will denote that x is not related to y under R.

LEMMA 1. *Let F denote the function computed by $W(B, S)$ on the closed domain D. If R is preserved by $W(B, S)$, then $F(x) \geqq x$, $\forall\, x \in D$.*

Proof. If R is preserved then $x\,R\,y \Rightarrow F(x)\,R\,y$. Hence the lemma follows from definition. □

DEFINITION 5. R is a *GE relation (generalized equivalence relation) with respect to B* if and only if

$$(\forall\, a, b \in D)[B(a) \wedge B(b) \wedge \exists\, c \in D[c \geqq a \wedge c \geqq b] \Rightarrow a \equiv b].$$

Observation. If R is a GE relation with respect to B then

$$[B(a) \wedge B(b) \wedge a \geqq b] \Rightarrow [a \equiv b].$$

This follows by using the fact that $a \geqq a$.

Example 3. The following are examples of GE relations:

(i) Let $D = \{x \mid x \text{ integer}; x \geqq 0\}$. For some fixed k,

$$B(x)\colon x > k.$$

Define R to be

$$x\,R\,y \Leftrightarrow |x - y| \leqq k, \qquad x, y \in D.$$

Note that if $B(x)$ is true then there is no $z \neq x$ for which $z \geqq x$. Hence (trivially) R is a GE relation with respect to B.

(ii) $D = \{x \mid x \text{ is an undirected graph}\}$.

$$(x, y \in D)\; x\,R\,y \Leftrightarrow x, y \text{ are isomorphic.}$$

It can be shown (see next lemma) that R is a GE relation for any B. □

LEMMA 2. *If R is an equivalence relation, then it is a GE relation with respect to every B.*

Proof. We first show that

$$[x \geqq y] \Leftrightarrow [x\, R\, y]$$

(i) $x\, R\, y \Rightarrow x \geqq y$:

$$x\, R\, y \wedge y\, R\, z \Rightarrow x\, R\, z,$$

since R is an equivalence relation. Thus, $x\, R\, y \Rightarrow x \geqq y$.

(ii) $x \geqq y \Rightarrow x\, R\, y$:

$$\forall z\, [y\, R\, z \Rightarrow x\, R\, z].$$

Since R is an equivalence relation, $y\, R\, y$ holds. Hence, $x\, R\, y$.

It thus follows that $[x \geqq y] \Leftrightarrow [x\, R\, y]$.

i.e., $[x \geqq y] \Leftrightarrow [x\, R\, y] \Leftrightarrow [y\, R\, x] \Leftrightarrow [y \geqq x]$,

i.e., $[x\, R\, y] \Leftrightarrow [x \equiv y]$,

i.e., $\exists\, z \in D[z \geqq x \wedge z \geqq y] \Rightarrow \exists\, z \in D[z \equiv x \wedge z \equiv y] \Rightarrow [x \equiv y]$.

Thus R is a GE relation for any B. □

The following theorem is the central result. In the statement of the theorem, only those $W(B, S)$ are considered for which D is closed and $W(B, S)$ terminates for all inputs from D.

THEOREM 2. *Let R be a binary relation and B a predicate on a given domain D. If R is a GE relation with respect to B then R is uniformly preserved by any $W(B, S)$ if it is preserved. Conversely, suppose R is not a GE relation with respect to B and for some S is preserved by $W(B, S)$. Then there exists $W(B, S')$ for which R is preserved though not uniformly.*

Proof. Let R be a GE relation with respect to B. Let F be the function computed by any $W(B, S)$ on domain D. By assumption, D is closed with respect to $W(B, S)$ and $W(B, S)$ terminates for every input from D. First we will show that

$$x\, R\, y \wedge B(x)\, \{S(x)\}\, x\, R\, y$$

i.e.,

$$x\, R\, y \wedge B(x) \Rightarrow S(x)\, R\, y.$$

Case (i). $\neg B(S(x))$:

$$B(x) \Rightarrow F(x) = S(x)$$
$$x\, R\, y \Rightarrow F(x)\, R\, y,$$

since R is preserved. Hence,

$$x\, R\, y \wedge B(x) \Rightarrow S(x)\, R\, y.$$

Case (ii). $B(S(x))$: The proof is by contradiction. Suppose that

$$x\, R\, y \wedge B(x) \wedge \neg(S(x)\, R\, y).$$

Then

$$x \not\equiv S(x),$$

since $x\,R\,y$ and $\neg(S(x)\,R\,y$. Using Lemma 1, $F(x) \geqq x$ and $F(S(x)) \geqq S(x)$. Using Theorem 1′, $F(x) = F(S(x))$. We thus have

$$B(x) \wedge B(S(x)) \wedge F(x) \geqq x \wedge F(x) \geqq S(x) \wedge x \not\equiv S(x).$$

Hence, R is not a GE relation with respect to B, contradiction.

Next we show that if R is not a GE relation with respect to some B and, for some S, R is preserved by $W(B, S)$, then there exists S' such that

(i) D is closed with respect to $W(B, S')$ and
(ii) $W(B, S')$ terminates for every input from D, and
(iii) R is preserved by $W(B, S')$, and
(iv) R is not uniformly preserved by $W(B, S')$.

We first state a claim whose proof is similar to that of Lemma 1.

Claim. If R is uniformly preserved by $W(B, S)$ then $B(x) \Rightarrow S(x) \geqq x$.

If R is not a GE relation with respect to B then there exist $x_1, x_2, x_3 \in D$ such that

$$B(x_1) \wedge B(x_2) \wedge x_3 \geqq x_1 \wedge x_3 \geqq x_2 \wedge x_1 \not\equiv x_2.$$

Since $x_1 \not\equiv x_2$, either $x_1 \not\geqq x_2$ or $x_2 \not\geqq x_1$. Without loss in generality assume that $x_2 \not\geqq x_1$.

The proof proceeds by constructing S'. Consider the following program.

```
while $B$ do
    if $x = x_1$ then $x := x_2$
        else if $x = x_2$ then $x := F(x_3)$
            else $S$;
```

It can be verified that conditions (i), (ii), (iii) are met by this program. Next we show that R is not uniformly preserved by this program. With input x_1, we obtain x_2 and then $F(x_3)$. However, $x_2 \not\geqq x_1$. Hence, according to the previous claim R is not uniformly preserved. Note that the first "else" clause in S' ensures that the program would terminate when input with x_1. □

Theorem 2 says that given any $W(B, S)$ and R which is a GE relation with respect to B, in order to prove that R is preserved, it is necessary and sufficient to prove that R is uniformly preserved. Conversely if R is not a GE relation with respect to B, it is sufficient though not necessary to prove that R is uniformly preserved in order to show that R is preserved.

COROLLARY 1. *Let R be an equivalence relation. For any $W(B, S)$ (assuming termination and closure of domain) R is preserved, if and only if it is uniformly preserved.*

Proof. Use Lemma 2 and Theorem 2. □

Example 4. Consider a program $W(B, S)$ for sorting an array x of integers. It is required to prove at the output that the resulting array is sorted and is a permutation of the input array. To prove the latter, we need to prove

$$PERM\,(x, x_0)\,\{W(B, S)\}\,PERM\,(x, x_0),$$

where $PERM\,(x, x_0)$ stands for "x is a permutation of x_0". Clearly $PERM$ is an equivalence relation. It is then necessary and sufficient to prove that $PERM\,(x, x_0)$

is a loop invariant,

$$PERM\,(x, x_0) \wedge B(x) \Rightarrow PERM\,(S(x), x_0),$$

i.e.,

$$B(x) \Rightarrow PERM\,(x, S(x)). \quad \square$$

The conditions derived in Theorem 2 clearly apply to the iteration condition in Theorem 1′. This can be seen easily by defining an equivalence relation R on D such that $x\,R\,y \Leftrightarrow [F(x) = F(y)]$. Clearly if $W(B, S)$ computes F, R is preserved. Using Theorem 2, it follows that it must be uniformly preserved.

We next formulate the conditions on R independent of B.

DEFINITION 6. R is a *GE relation on a domain D* if and only if it is a GE relation with respect to every predicate B (binary valued total function on D).

THEOREM 3. *R is a GE relation on D if and only if its derived relation ($\geqq$) is an equivalence relation.*

Proof. If $\geqq$ is an equivalence relation then

$$[c \geqq a \wedge c \geqq b] \Rightarrow [a \geqq b \wedge b \geqq a] \Rightarrow [a \equiv b].$$

Hence R is a GE relation for any B. Conversely, if $\geqq$ is not an equivalence relation then there exist a, b such that $a \geqq b$ and $a \not\equiv b$. Consider some predicate B for which $B(a)$ and $B(b)$ are true. R is not a GE relation with respect to this B since

$$B(a) \wedge B(b) \wedge a \geqq a \wedge a \geqq b \wedge a \not\equiv b. \quad \square$$

For any relation R, if we define the successor set of a, $T(a) = \{b \mid a\,R\,b\}$, then R is a GE relation if and only if no successor set $T(a)$ strictly includes another successor set $T(b)$, since otherwise $a \geqq b$ and $b \not\geqq a$.

Observation. If R is a GE relation then either all successor sets are null or none is.

In program proving, it would be easier to consider general GE relations rather than GE relations with respect to a specific B. Theorem 3 provides a useful technique for proving that a certain R is a GE relation. The next theorem essentially provides the verification conditions that must be proved to ensure that a GE relation is preserved.

THEOREM 4. *Let R be a GE relation. Then (assuming termination and closure)*

$$x\,R\,y\,\{W(B, S)\}\,x\,R\,y,$$

if and only if

$$B(x) \Rightarrow [x \equiv S(x)].$$

Proof. We first show that if R is a GE relation,

$$x\,R\,y\,\{W(B, S)\}\,x\,R\,y$$

if and only if

$$x \equiv z\,\{W(B, S)\}\,x \equiv z.$$

Let F be the function computed by $W(B, S)$ (on the domain D). Then if R is preserved by $W(B, S)$, $F(x) \geqq x$ or $F(x) \equiv x$, since R is a GE relation. This says that the input x and output $F(x)$ belong to the same equivalence class under $\equiv$.

Conversely, let

$$x \equiv z \ \{W(B, S)\} \ x \equiv z.$$

Thus, $F(x) \equiv x$. Hence $F(x) \geqq x$ or $x \, R \, y \Rightarrow F(x) \, R \, y$. Hence, R is preserved by $W(B, S)$.

Using Corollary 3,

$$x \equiv z \ \{W(B, S)\} \ x \equiv z$$

if and only if $\equiv$ is uniformly preserved, i.e.,

$$x \equiv z \wedge B(x) \Rightarrow S(x) \equiv z,$$

i.e.,

$$B(x) \Rightarrow [x \equiv S(x)]. \quad \square$$

Example 5. The following program is claimed to compute the greatest common divisor of m, n using successive subtraction.

```
begin
    integer $t$;
    while $m \neq n$ do
    begin
      if $m < n$ then begin $t := m; m := n; n := t$ end;
      $m := m - n$
    end
end;
```

Let $D = \{(m, n) \mid m, n \text{ integer}; m, n > 0\}$. Let GCD be the function of two arguments that has the value of the greatest common divisor of the arguments. Let H be a function from domain D to range D, defined as follows:

$$H(m, n) = (\text{GCD}(m, n), \text{GCD}(m, n)).$$

We wish to show that

$$[H(m, n) = H(m_0, n_0)] \{W(B, S)\} [H(m, n) = H(m_0, n_0)],$$

where $W(B, S)$ represents the above program and $(m, n), (m_0, n_0) \in D$ at entrance to the loop.

The reason for using H instead of GCD is that the former is a function from D to D, as required by Theorem 4, whereas the latter is a function from D to a subset of positive integers.

We must first prove closure and termination.

(i) Closure:

$$m, n \text{ integer} \wedge m > 0 \wedge n > 0 \wedge m \neq n \ \{S\} \ m, n \text{ integer}, m > 0, n > 0.$$

Equivalently, we must show,

$$[m, n \text{ integer}, m > 0 \wedge n > 0 \wedge m < n] \Rightarrow [m, n - m \text{ integer} \wedge n - m > 0 \wedge m > 0]$$

and

$$[m, n \text{ integer} \wedge m > 0 \wedge n > 0 \wedge m > n] \Rightarrow [n, m-n \text{ integer} \wedge m-n > 0 \wedge n > 0].$$

(ii) Termination: It is then necessary and sufficient to show that $H(m, n) = H(m_0, n_0)$ is a loop invariant, i.e.

$$[H(m, n) = H(m_0, n_0)] \wedge m \neq n \ \{S\}\ [H(m, n) = H(m_0, n_0)],$$

i.e.

$$[m \neq n \wedge m < n] \Rightarrow [H(m, n) = H(n-m, m)]$$

and

$$[m \neq n \wedge m > n] \Rightarrow [H(m, n) = H(m-n, n)]$$

i.e.,

$$[m < n \Rightarrow \text{GCD}(m, n) = \text{GCD}(n-m, m)]$$

and

$$[m > n \Rightarrow \text{GCD}(m, n) = \text{GCD}(m-n, n)].$$

All that remains to be proved is the boundary condition given below, in order to show that m, n both have the value of $\text{GCD}(m_0, n_0)$ at the exit.

$$\{m = n \wedge [H(m, n) = H(m_0, n_0)]\} \Rightarrow \{m = \text{GCD}(m_0, n_0) \wedge n = \text{GCD}(m_0, n_0)\}. \quad \square$$

Finally, we show that a certain simple class of relations, as given in Example 3, can be shown to be preserved by extending the given relation to an equivalence relation and proving that the latter is preserved.

DEFINITION 7. Given any relation R on D, define the *reflexive, symmetric, transitive closure R^* of R* as follows.

$$a R^* b \Leftrightarrow (a = b) \vee a R b \vee b R a \vee [\exists c\ (a R^* c \wedge c R^* b)], \quad a, b, c \in D.$$

Thus, R^* is an equivalence relation. Under certain conditions, it is both necessary and sufficient to prove that R^* is uniformly preserved, in order to show that R is preserved.

THEOREM 5. *Let R be any relation on D and R^* be its reflexive, symmetric transitive closure. Suppose for some predicate B on D,*

$$x_1 R^* x_2 \wedge \neg B(x_1) \Rightarrow x_1 R x_2, \quad \forall x_1, x_2 \in D.$$

Then

$$x R y \ \{W(B, S)\}\ x R y$$

if and only if R^* is uniformly preserved, assuming closure and termination.

Proof. We show that

$$x R y \ \{W(B, S)\}\ x R y \quad \text{if and only if} \quad x R^* y \ \{W(B, S)\}\ x R^* y.$$

Since R^* is an equivalence relation, the statement in the theorem would follow. Let F be the function computed by $W(B, S)$. Suppose

$$x R y \ \{W(B, S)\}\ x R y.$$

Then

$$x\,R\,y \Rightarrow F(x)\,R\,y \Rightarrow F(x)\,R^*\,x.$$

Thus, x and $F(x)$ belong to the same equivalence class under R^*, or

$$x\,R^*\,y\,\{W(B,S)\}\,x\,R^*\,y.$$

Next, suppose, $x\,R^*\,y\,\{W(B,S)\}\,x\,R^*\,y$. Then on termination, $\neg B(x) \wedge x\,R^*\,y \Rightarrow x\,R\,y$, or $x\,R^*\,y\,\{W(B,S)\}\,x\,R\,y$.

Since $x\,R\,y \Rightarrow x\,R^*\,y$, it follows that $x\,R\,y\,\{W(B,S)\}\,x\,R\,y$. □

Example 6.

```
begin
    while $v \neq 0$ and $v \neq 1$ do $v := v - 2$
end;
```

$D = \{v \mid v \geqq 0;\ v \text{ integer}\}$. $v, u \in D$, $v\,R\,u \Leftrightarrow (v \leqq u)$ and u, v have identical parity (both even or odd). $v\,R^*\,u \Leftrightarrow u, v$ have identical parity.

$$(v, u \in D)\,\{v\,R^*\,u \wedge [v = 0 \vee v = 1]\} \Rightarrow [v \leqq u],$$

and u, v have identical parity.

Hence, according to Theorem 5, it is necessary and sufficient to prove that $v\,R^*\,u$ is a loop invariant, in order to show that R is preserved; i.e.

$$v\,R^*\,u \wedge v \neq 0 \wedge v \neq 1 \Rightarrow (v-2)\,R^*\,u.$$

Termination and closure must be proven separately. □

4. Summary and conclusion. We have shown that any equivalence relation is uniformly preserved if it is preserved by a loop program. Theorem 5 extends the results somewhat for relations that are essentially equivalence relations except for certain boundary conditions. A practical outcome of this result is that loop invariants may be generated algorithmically for certain classes of input/output relations. There is no need to look through the body of the loop to generate the invariant, provided closure and termination have been proven separately.

Unfortunately, the results also establish that such conditions cannot be obtained for any other classes of relations. Thus, even a simple transitive relation such as "$\leqq$" on positive integers is not uniformly preserved even though it is preserved. We believe that one needs to look at the program body S, for generating the loop invariant, for all other classes of relations.

One promising direction of research is to consider other classes of relations and "reasonable" programs. The loop invariant could be generated if the program meets certain reasonable restrictions; for example, we may assume that a program operating on stacks may not process any other element, before processing the top element, (i.e. it should not be allowed to save the top element, process and remove the second element from top and then restore the top element). Some preliminary results appear in [3], [7].

Another problem we have not considered in this paper is the problem of nonclosed domains. Frequently, a loop is preceded by initializations which restrict the input domain. Most of the time, the domain will not be closed with respect to the program. It is often required to prove a certain relation (such as a functional equality) in the restricted domain. This problem has been considered for the case of functional equality in [2], [3], [7]. This seems to be the major problem in synthesis of loop invariants. It seems likely, however, that by suitably restricting the operations of the program, it may be possible to prove that the program computes a certain relation over a superset of the given domain (which is closed) from which the stated conjecture may be proven.

Acknowledgment. The author is indebted to the referees for their constructive comments.

REFERENCES

[1] S. BASU AND J. MISRA, *Proving loop programs*, IEEE Trans. on Software Engrg., 1 (1975), pp. 76–86.
[2] ———, *Deterministic generation of inductive assertions*, IEEE Workshop on Automated Theorem Proving, Argonne National Lab., Argonne, IL, 1975.
[3] ———, *Some classes of naturally provable programs*, Proc. Second International Symposium on Reliable Software, San Francisco, 1976.
[4] C. A. R. HOARE, *An axiomatic approach to computer programming*, Comm. ACM, 12 (1969), pp. 576–580, 583.
[5] S. KATZ AND Z. MANNA, *A heuristic approach to program verification*, Proc. 3rd International Conference on Artificial Intelligence, Stanford Univ., Stanford, CA, 1973.
[6] J. MISRA, *Relations uniformly conserved by a loop*, Proc. 9th International Symposium on Proving and Improving Programs, Arc et Senans, France, 1975, pp. 71–80.
[7] ———, *Some aspects of verification of loop computation*, unpublished manuscript.
[8] M. MORICONI, *Semiautomatic synthesis of inductive predicates*, ATP-16, Dept. of Mathematics, Univ. of Texas at Austin, 1974.
[9] J. H. MORRIS AND B. WEGBREIT, *Subgoal Induction*, Xerox Palo Alto Research Center, 1975.
[10] B. WEGBREIT, *Heuristic methods for mechanically deriving inductive assertions*, Proc. 3rd International Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, 1973.