AUTOMATIC DISCOVERY OF LINEAR RESTRAINTS AMONG VARIABLES OF A PROGRAM

Patrick Cousot* and Nicolas Halbwachs**
Laboratoire d’Informatique, U.S.M.G., BP. 53
38041 Grenoble cédex, France

1. INTRODUCTION

The model of abstract interpretation of programs developed by Cousot[1976], Cousot[1977] is applied to the static determination of linear equality or inequality relations among variables of programs.

For example, consider the following sorting procedure (Knuth[1973], p.107) :

procedure BUBBLESORT(integer value N;
    integer array[1:N] K)

begin Integer B,J,T;
    B=N;
    {1} while B>1 do
        {2} J=1; T=0;
        {3} while J=B-1 do
            {4} if K[J]<K[J+1] then
                {5} EXCHANGE(J,J+1); {no side effects on
                N,B,J,T}
            {10} if T=0 then return fi;
            {11} B=J;
        {12} od;
    end;

Without user provided inductive assertions nor human interaction we have automatically determined (in 1.582 seconds of C.P.U. time) that the following restraints must hold among the variables of the above procedure:

{1} B=N
{2} 1≤B≤N
{3} 1≤B≤N, J=1, T=0
{4},{5},{6} B≤N, T≥0, T+1≤J, J+1≤B
{7} B≤N, J≥1, J+1≤B, J=T
{8} B≤N, J+1≤B, J≥1, T≥0, T+1≤J
{9} T≥0, J+1≤N, J=0, T=0

A certain number of classical data flow analysis techniques are included in or generalized by the determination of linear equality relations among program variables. For example constant propagation can be understood as the discovery of very simple linear equality relations among variables (such as X=1, Y=5). However the resolution of the more general problem of determining linear equality relations among variables allows the discovery of symbolic constants (such as X=N, Y=5+N+1). The same way, common subexpressions can be recognized which are not formally identical but are semantically equivalent because of the relationships among variables. Also the loop invariant computations as well as loop induction variables (modified inside the loop by the same loop invariant quantity) can be determined on a basis which is not purely syntactical. The problem of discovering linear inequality relations is in fact a particular case of the one of discovering linear inequality relations among the program variables. The main use of these inequality relationships is to determine at compile time whether the value of an expression is within a specified numeric or symbolic subrange of the integers or reals. This includes compile time overflow, integer subrange and array bound checking. In contrast to Suzuki-Ishihata[1977] we do not simply try to verify the legality assertions (such as verifying that array subscripts are within the declared range) but instead we try to discover the assertions (of linear type) that can be deduced from the semantics of the program. The advantage is that we can often discover relations which are never stated explicitly in the program. For example, we can discover that an integer variable lies in a subrange of its declared range or that two references A[I] and A[J] to two elements of the same array refer to different storage locations.

{10},{11} B≤N, J≤B, T≥0, T+1≤J, B≤J+1
{12} J≤B, T≥0, T+1≤J, J≤B
{13} B≤N, B≤J

(since e.g. I≥J+1) or that some piece of code is dead.

The problem of determining equality relationships between a linear combination of the variables of the program and a constant was solved by Karr [1978]. His approach was based on Wegbreit[1975]'s algorithm which requires that the properties to be discovered form a lattice every strictly increasing chain of which is of finite length. This assumption is not valid when considering inequality relationships (because of chains such as (x<1), (1≤x<2),...,(1≤x<n),...).
The model of Cousot [1977] is general enough to cope
with this problem and we briefly recall it in
section 2 as formulated in Cousot [1976]. In section
3 we study formal representations for the particular
type of assertions that we consider. In section 4
we describe the linear restraints transformer corre-
sponding to elementary instructions of the language.
The algorithm performing the global analysis of
programs is presented in section 5 by means of
simple examples. Section 6 gives more convincing
examples and Section 7 discusses the experimental
implementation of the analysis.

2. APPROXIMATE ANALYSIS OF PROGRAM PROPERTIES,
Cousot [1976].

For purposes of exposition, a sequential program
will be represented by a connected finite flowchart
with one entry node and assignment, test, junction
and exit nodes. The evaluations of the right-hand
side of an assignment and of the boolean expression
in a test node are assumed not to affect the values
of any variables. Thus all side-effect phenomena
must be modeled as assignment statements. The
junction nodes contain no computations and represent
the merge of program execution paths.

The analysis of a program consists in attaching
an assertion $P_i (V_1, ..., V_n)$ to each arc $i$ of the
program. These predicates on the variables
$V_1, ..., V_n$ are not necessarily of the most general
form but instead are designed to model a specific
aspect of the semantic properties of the program.
The assertion on the entry arc to the program
represents what is known about the variables at
the start of execution. For each other type of program
node a transformation specifies the assertion
associated with the output arc(s) of the node in
terms of the assertions on the input arc(s) to the
node and where relevant the content of the node.
Hence we have established a system of equations

3. FORMAL REPRESENTATIONS OF LINEAR RESTRAINTS AMONG
VARIABLES OF A PROGRAM.

3.1. Linear system of a convex polyhedron.

Let $x^1, ..., x^n$ be the variables of the program.
For simplicity we assume that the values of the
variables belong to the set $\mathbb{R}$ of reals. The set of
solutions to a system of linear equations
$$\sum_{j=1}^n a_{ij} x^j = b_i \quad i = 1, ..., m$$
where $a_{ij}, b_i \in \mathbb{R}$, if such solutions exist, is a linear variety of $\mathbb{R}^n$. A
linear variety of dimension $n-1$ is an hyperplane.

The set of solutions to a linear inequality
$$\sum_{i=1}^n (a_i x^j) - b \geq 0$$
is a closed half-space of $\mathbb{R}^n$. For simplicity, strict inequalities are not considered.
By linear restraint we mean either a linear equality
or a linear inequality. In the formal reasoning we
often consider that an equation can be viewed as
two opposite inequalities.

A subset $C$ of $\mathbb{R}^n$ is said to be convex if and
only if $\{\lambda x_1, \lambda x_2 \in C, \lambda \in [0,1], \lambda x_1 + (1-\lambda) x_2 \in C\}$
for example linear varieties and half-spaces are convex.
The intersection of two convex sets is convex, but
the union of two convex sets is not necessarily
convex.

The set of solutions to a finite system of linear
inequalities can be interpreted geometrically as the
closed convex polyhedron of $\mathbb{R}^n$ defined by the
intersection of the closed halfspaces corresponding
to each inequality.

3.2. The frame of a convex polyhedron, Weyl [1950],
Klee [1958], Charnes [1953].

Let $V_1, ..., V_p$ be vectors in $\mathbb{R}^n$. A vector of
the form $\sum_{i=1}^p V_i$ where for each $i=1, ..., p$ we have
$V_i \in \mathbb{R}$ is called a linear combination of the $V_i$.

The set of all linear combinations of the $V_i$ is the
linear variety generated by the $V_i$. A basis of a
linear variety $L$ is a minimal set of vectors
A line of a polyhedron $P$ is a vector $d$ such that both $d$ and $-d$ are rays of $P$: \( \forall x \in F, x + k \cdot d \in P \). A polyhedron which contains at least one line is called a cylinder. The linear variety generated by all the lines of a cylinder is the greatest linear variety included in the cylinder. A polyhedron that contains no line has only a finite number of vertices and of extreme rays.

A bounded polyhedron has neither lines nor rays. Each point of a bounded polyhedron is a convex combination of the vertices of the polyhedron so that a bounded polyhedron is the convex-hull of its vertices.

Each point $x$ of a polyhedron $P$ which is not a cylinder can be expressed as the sum of a convex combination of the vertices \( \{ s_1, \ldots, s_n \} \) of $P$ and of a positive combination of the extreme rays \( \{ r_1, \ldots, r_k \} \) of $P$:

\[
\begin{align*}
\lambda_1 & \cdot \ldots \cdot \lambda_n \in [0,1], \\
\mu^\prime_1 & \cdot \ldots \cdot \mu^\prime_k \in \mathbb{R}^+, \\
\lambda_1 & \cdot \ldots \cdot \lambda_n \mu^\prime_1 + \ldots + \lambda_1 \cdot \ldots \cdot \lambda_n \mu^\prime_k,
\end{align*}
\]

Let $L$ be the greatest linear variety included in the cylinder, and $L'$ be a linear variety orthogonal to $L$. Then the intersection of $L'$ with $P$ is a convex polyhedron which contains no line and which is called a section of $P$. Each point of a cylinder can be expressed as the sum of a convex combination of the vertices of a section of $P$, a positive combination of the extreme rays of this section and a linear combination of the vertices of a basis of the greatest linear variety included in $P$.

By abuse of words the vertices and rays of a cylinder will be the vertices and rays of a section of that cylinder.

A closed convex polyhedron $P$ can be characterized by three sets $S = \{ s_1, \ldots, s_n \}$, $R = \{ r_1, \ldots, r_k \}$, $D = \{ d_1, \ldots, d_l \}$ of vectors of $\mathbb{R}^m$ called the frame of the polyhedron as follows:

\[
\begin{align*}
\forall x \in F & \Rightarrow \exists \lambda_1, \ldots, \lambda_n \in [0,1], \\
\forall r & \in R \Rightarrow \lambda_1 \cdot \ldots \cdot \lambda_n \mu^\prime_1 + \ldots + \lambda_1 \cdot \ldots \cdot \lambda_n \mu^\prime_k = x \\
\text{and } s & \in S \Rightarrow \lambda_1 \cdot \ldots \cdot \lambda_n + \sum \mu^\prime_1 \cdot v_j + \ldots + \sum \mu^\prime_k \cdot v_j = x.
\end{align*}
\]

We have two equivalent representations of a closed convex polyhedron either as the set of solutions of its system of linear restraints or as the convex hull of its frame.

Example: The polyhedron defined by the following system of restraints:

\[
\begin{align*}
x_1 & \geq 2, \\
x_2 & \geq 1, \\
x_1 & \geq 0, \\
x_1 & \geq 2x_2 \geq 6, \\
x_1 & \geq 2x_2 \geq 6.
\end{align*}
\]

is spanned by the following frame (see the diagram):

- vertices: $s_1 = (1, 2), s_2 = (1, 1), s_3 = (1, 0)$
- extreme rays: $r_1 = (1, 1), r_2 = (1, 0)$, no line.

Before looking at the problem of conversion between these representations we need some definitions.

A face of a polyhedron $P$ is a convex polyhedron $F \cap P$ such that if a point $x$ lies in $F$ then each segment included in $P$ and containing $x$ is included in $F$. $\forall x \in F, p, q, \lambda \in [0,1]$ and $x + \lambda p + (1-\lambda)q \in F$ for $p, q \in F$.

The dimension of a polyhedron $P$ is the dimension of the linear variety containing $P$.

A face of dimension $k$ is called a $k$-face. An edge is a 1-face. The vertices of a polyhedron containing no line are its $0$-faces (thus a cylinder has no $0$-face). Let $\sum_{i=1}^n (a_i x') = b'$. Let $\sum_{i=1}^n (a_i x') = b'$ be a linear inequality defining the polyhedron $P$. Then if the intersection of the hyperplane defined by $H\{x : \sum_{i=1}^n (a_i x') = b'\}$ with $P$ is not empty, it is a face of $P$.

We say that a point $s$ satisfies the inequality $\sum_{i=1}^n (a_i x') = b'$ if and only if $\sum_{i=1}^n (a_i x') = b'$.

We say that the ray $r$ satisfies this inequality if and only if $\sum_{i=1}^n (a_i r') = 0$.

Let $\delta$ be the dimension of the greatest linear variety included in the polyhedron $P$. Two vertices are said to be adjacent iff they lie on the same edge (i.e. if $\delta \neq 0$ we mean the edge of the section). It follows that the number of inequalities saturated at the same time by two adjacent vertices is at least $n-\delta-1$.

Two extreme rays are said to be adjacent iff they belong to the same 2-face. The vertices of a polyhedron saturate at least $n-\delta-2$ inequalities simultaneously.

Finally, a vertex $s$ and a ray $r$ are adjacent iff $s$ lies on an infinite edge which is parallel to $r$. Therefore $s$ must saturate all the inequalities saturated by $r$.

3.3 Conversions between the representations of a polyhedron by a system of linear restraints and by a frame.

Some operations that we have to perform on closed convex polyhedra are easy when these polyhedra are represented by systems of linear restraints while others are more simple when the frame representation is used. Hence we must be able to make conversions from one representation to the other.

3.3.1 Conversion from the frame to the linear restraints representation.

This conversion consists in finding a system of restraints of the convex-hull of the elements (points, rays, lines) of a frame. This conversion is followed by a simplification of the system of restraints.

3.3.1.1 Convex-hull of a finite frame.

Let $S = \{ s_1, \ldots, s_n \}, R = \{ r_1, \ldots, r_k \}, D = \{ d_1, \ldots, d_l \}$ be the frame of a non-empty polyhedron (so that $S \neq \emptyset$). The points $x$ of this polyhedron are characterized by the existence of $\lambda_1, \ldots, \lambda_n, \mu_1, \ldots, \mu_k, v_1, \ldots, v_{\delta}$ of $\mathbb{R}$ such that:

- $\delta \lambda_i \leq 1$ for $i=1,\ldots,\sigma$
- $\sum_{i=1}^\sigma (\lambda_i) = 1$
- $\sum_{j=1}^\delta (\lambda_i, \lambda_j) = 1$
- $\sum_{j=1}^\delta (\mu_j, r_j)$ for $j=1,\ldots,\rho$
- $x = \sum_{i=1}^\sigma (\lambda_i, s_i) + \sum_{j=1}^\delta (\mu_j, r_j) + \sum_{k=1}^\delta v_k d_k$. 

86
Hence this polyhedron is characterized by a system

The first approximation $P_1(s)$ is defined by the
3.4.1. Basic concepts of linear programming

A system of linear inequalities:

\[ \sum_{i=1}^{n} (a_{ij}^T x^T) \leq b_j : j=1..m \]

can be written in the equivalent form:

\[ \sum_{i=1}^{n} (c_{ij}^T x^T) + y^T b_j \geq 0 : j=1..m \]

Hence a system of constraints can be written in standard form:

\[ AX = B, X \geq 0 \]

where \( A \) is a \((n+m) \times m\) real matrix, \( B \) is an \( m \)-vector \( E=(n+1, ..., n+m), F=(1, ..., n) \). The variables \( \{x^i : i \in F\} \) are the initial variables, the \( \{x^i : i \in E\} \) are the slack variables.

A basis of the system [3.4.1.2] is a non-singular \( m \times m \) submatrix \( A_i \) of \( A \). The system can be written in canonical form with respect to the basis \( A_i \):

\[ A_i^T (A_i A_i^T)^{-1} A_i x = A_i B, x \geq 0 \]

\( i=1, ..., n+m \). The variables \( x^i \) such that \( i \in I \) are said to be in basis. The basis \( A_i \) is feasible if and only if \( A_i^T B \geq 0 \). Two feasible bases \( A_i \) and \( A_j \) are said to be adjacent if the \( i\text{-th} \) column of \( A_i \) that is \( j\text{-th} \) zero

feasible basis corresponding to a vertex \( s \), we find all extreme rays adjacent to \( s \). Since each extreme ray of a polyhedron contains a line that is adjacent to a vertex, we can find all extreme rays of such a polyhedron.

For polyhedra containing a line there is no feasible basis containing all initial variables.

Hence let \( (AX=B, X \geq 0) \) be the canonical form with respect to any feasible basis \( A_i \). Let \( i_0 \in (F-I) \) be a column satisfying the condition:

\[ A_j^T B \neq 0 \text{ for all } j \neq i_0 \]

Let \( d(i_0) \) be the vector of \( \mathbb{R}^n \) the \( i\text{-th} \) component of which is defined by:

\[ d(i_0)^j = \begin{cases} 1 & \text{if } i = i_0 \\ 0 & \text{else} \end{cases} \]

Then Farkas shows that \( d(i_0) \) corresponds to a line of the polyhedron.

Also if the basis \( A_i \) is such that each \( i_0 \in (F-I) \) verifies the property (3.4.2.2) then the set...
\{AX=0, \sum_{j=1}^{n} a(i,j)x^j = 0, \forall k=1..6\} defines a new polyhedron \(P'\) which is a section of the initial polyhedron. Applying the algorithm to this new system of restraints case 4.1 is applicable since \(P'\) contains no lines so that the algorithm terminates.

Example: Let us compute a frame of the polyhedron \(P\) corresponding to the following system of restraints:

\[
\begin{align*}
-x_1 x_2 - x_3 & \leq 0 \\
-x_1 & \leq -1 \\
-x_1 x_2 + x_3 & \leq 0 \\
-x_2^2 + x_3 & \leq 3
\end{align*}
\]

\([S1]\)

1-Using the matrix notations for denoting systems of equations the standard form of \([S1]\) is:

\[
\begin{array}{cccccccc}
 x^1 & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 \\
\hline
-1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\
-1 & -1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & 0 & 1 & 3
\end{array}
\]

\([S2]\)

The system \([S2]\) is in canonical form with respect to and \(A_{[2,3,4,5,6]}\) is a stable basis:

<table>
<thead>
<tr>
<th>(x^1)</th>
<th>(x^2)</th>
<th>(x^3)</th>
<th>(x^4)</th>
<th>(x^5)</th>
<th>(x^6)</th>
<th>(x^7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>-1/2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1/2</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\([S6]\)

Basis: \(A_{[2,3,4,5,6]}\)
Vertex: \(s_1 = (1,1/2,-1/2)\)
Column 5 satisfies \([3.4.2.1]\)
Row: \(r_1 = (1,1/2,-1/2)\)
Adjacent feasible basis: \(A_{[2,3,4,5,6]}\)

<table>
<thead>
<tr>
<th>(x^1)</th>
<th>(x^2)</th>
<th>(x^3)</th>
<th>(x^4)</th>
<th>(x^5)</th>
<th>(x^6)</th>
<th>(x^7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>-1/2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1/2</td>
<td>1/2</td>
<td>0</td>
<td>1/2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

\([S7]\)

Basis: \(A_{[2,3,4,5,6]}\)
Vertex: \(s_2 = (1,-1/2,1/2)\)
No column satisfies \([3.4.2.1]\)
Adjacent feasible basis: \(A_{[2,3,4,5,6]}\) already found
3.5. On the use of two representations for assertions.

The search for the frame of a polyhedron as described above may become very expensive although many important optimizations are possible. The number of vertices of a polyhedron generated in \( \mathbb{R}^n \) by \( m \) inequalities is known to be bounded only by functions increasing very quickly with \( m \) and \( n \), Saaty[1955], Klef[1964]. So the use of this method should be avoided except when applied to polyhedra which are thought to have very few vertices. However the use of two representations is necessary for the following reasons:
- Some operations like the convex-hull of two polyhedra can only be performed on the frame representation, others like widening require the restraints representation.
- In order to define the inclusion relation between two polyhedra it is very useful to know both representations. Indeed \( P_2 \cap P_1 \) if and only if each element of the frame of \( P_1 \) satisfies the restraints of \( P_2 \).
- It appears that it is difficult to simplify one representation without knowing the other one and neither can be used efficiently without simplifications. So we shall build at the same time and in a consistent way the systems of restraints and the frames of the polyhedra. Both representations will be simultaneously used to represent the assertions. Experience shows that this redundant representation is much less expensive than the frequent use of conversions.

4. TRANSFORMATION OF LINEAR ASSERTIONS BY ELEMENTARY LANGUAGE CONSTRUCTS.

Given the flowchart representation of programs we now give for each type of program node a transformation specifying the assertion associated with the output arc(s) of the node in terms of the assertions associated with the input arc(s) of the node and where relevant the content of this node. The conditions of Cousot[1977] guaranteeing the soundness of these transformations are:

4.2. Assignment.

Let \( \{A',B',S',R',D'\} \) be the representation of the input assertion \( P' \) corresponding to the polyhedron \( AX < B \) the frame of which is \( S_0 = \{s_i^j : i = 1..\sigma\} \), \( R_0 = \{r_j^i : j = 1..\rho\}, D_0 = \{d_k^i : k = 1..\delta\} \). Then the representation of the output assertion \( P \) after the assignment \( X^1 = E(X) \) is given by:

\[
\text{assign } (P, X^1 = E(X)) = \{A', B', S', R', D'\}.
\]

4.2.1. Assignment of a non linear expression.

If the expression \( E(X) \) is not linear it is assumed that any value of \( R \) can be assigned to \( X^1 \).
Hence after the assignment \( X^1 = E(X) \) nothing is known about the value of \( X^1 \). Therefore \( X^1 \) is eliminated from the system of restraints \( AX < B \) by a projection along the column \( 1_0 \) (§ 3.3.1.1). For the frame representation this consists in adding the line \( d \) (such that \( d_j = 1 \) and \( d_j = 0 \) for every \( j = 1..\delta \) different from \( 1_0 \)) to the set of lines \( D \). Then in general the representation \( \{A', B', S', R', D'\} \) of \( P' \) is not minimal and must be simplified (§3.3.1.2 and §3.4.4).

Example: Let \( n = 2 \) and \( P \) be the input assertion represented by the system of restraints:

\[
\begin{align*}
X_1 - X_2 & \geq -1 \\
X_2 & \geq 1 \\
X_1 + X_2 & \geq 5
\end{align*}
\]

The frame representation is \( S = \{(2, 3), (4, 1)\}, R = \{(1, 1), (1, 0)\}, D = \emptyset \). The geometrical interpretation is:

\[
\text{assign } (P, X^1 = E(X)) = \{A', B', S', R', D'\}.
\]
The approximation is obviously very coarse since the substitution of $x_1 x_2$ for $x_2$ in the input system of constraints would lead to:

$$
\begin{align*}
    x_2 &\leq x_1 \cdot (x_2)^2 \\
    x_2 &\geq x_1 \\
    x_2 &\geq 5x_1 - (x_1)^2
\end{align*}
$$

However, the corresponding domain is not a polyhedron and this situation is hardly manageable:

Note however that the exact domain is covered by the approximate domain (Cousot [1977]). Also, a more precise analysis is feasible. For example $\{x_1 \geq 0, x_2 \geq 0\}$ implies $\{x_1 x_2 \geq 0\}$ or the assignment $x := y_{x \geq 2}$ implies that $x$ is greater than or equal to zero.

**End of example.**

### 4.2.2. Assignment of a linear expression

The assignment is of the form $X^{l_0} = \sum_{i=1}^{n} (a_i X^{l_i}) + b$ where $a_i$ is a $n$-row vector of integers or reals and $b$ is an integer or a real. The transformation consists in an alteration of the basis of the space $\mathbb{R}^n$. The output assertion $P'$ is defined by the frame $\{S', R', D'\}$ computed as follows:

1. **$S'$** is $\{s'_1, ..., s'_l\}$ where $s'_i$ is defined by $s^{l_0}_i = a_i \cdot b_{i}^{l_0}$ and $s^{l_1}_i = a_i$ where $l = 1..n$ with $l \neq 0$.

2. **$R'$** is $\{r'_1, ..., r'_l\}$ where $r'_j$ is the vector defined by $r^{l_0}_j = c_{j}^{l_0}$ and $r^{l_1}_j = c_{j}$ for $l = 1..n$ and $l \neq 0$.

3. **$D'$** is $\{d'_1, ..., d'_l\}$ where $d'_i$ is the vector defined by $d^{l_0}_k = a_{i} d_{i}^{l_0}$ and $d^{l_1}_k = a_{i}$ for $l = 1..n$ and $l \neq 0$.

The system of restraints corresponding to $P'$ can be obtained as the convex-hull of the frame $\{S', R', D'\}$. However, following Karr [1976] this system of restraints can be computed directly.

#### 4.2.2.1. Invertible assignments

The assignment is of the form $X^{l_0} = \sum_{i=1}^{n} (a_i X^{l_i}) + b$, $a_i \neq 0$. The fact that $a_i \neq 0$ allows us to carry over our knowledge of the previous value of $X^{l_0}$ to the new value of $X^{l_0}$. To see this, denote the values of the variables by $X$ before and by $X'$ after the invertible assignment statement. Then for $l = 1..n$ and $l \neq 0$ we have $X^{l_0} = X^{l_1}$ whereas $X^{l_1} = a X + b$. Therefore $X = M X' + K$ as defined by

$$
X^{l_0} = X^{l_1} = M X' + b
$$

Also $(AX \leq b)$ is equivalent to $((MA)^t \leq [B-AK])$ which leads to the output system of restraints satisfied by $X'$.

#### 4.2.2.2. Non-invertible assignments

If $a_i = 0$ then we cannot solve $X'$ in terms of $X$ so that some information is lost by the assignment. Hence $X^{l_0}$ is eliminated from the input restraints by a projection operation. The case is similar to the one of the assignment of a non-linear expression except that the restraint $X^{l_0} = a X + b$ is adjoined to the resulting system.

**Example:**

Let $P$ be the input assertion defined by

$$
\begin{align*}
    x_2 &\geq 1 \\
    x_1 x_2 &\geq 5 \\
    x_1 - x_2 &\geq -1
\end{align*}
$$

The assignment $x_2 := x_1 + 1$ is not invertible. The elimination of $x_2$ in the input system of restraints leads to $\{x_1 \geq 2\}$ so that the output system of restraints is:

$$
\begin{align*}
    x_1 &\geq 2 \\
    x_2 - x_1 &\geq 1
\end{align*}
$$

The assignment $x_2 := x_1 x_2 / 2 + 1$ is invertible so that $x_1 x_1' + x_2 = 2x_1' - 2x_1^2$. Substituting in the input system of restraints we get the output system:

$$
\begin{align*}
    2x_1' - 2x_1^2 &\geq 3 \\
    2x_2' - x_1^2 &\geq 7 \\
    -2x_1' + 3x_1^2 &\geq -3
\end{align*}
$$

**End of example.**

### 4.3. Test Nodes

Let $P(A,B,S,R,D)$ be the assertion associated
with the input arc to a decision node testing some boolean condition C. Let \( P_t (A_t, B_t, S_t, R_t, D_t) \) and \( P_f (A_f, B_f, S_f, R_f, D_f) \) be the assertions associated respectively with the true and false exits of the test. Obviously \( P_t = P \) and \( C \) and \( P_f = \neg C \).

The condition \( C \) is said to be linear if and only if it is of the form \( aX \leq b \) or \( aX = b \) where \( a \) is an \( n \)-row-vector of integers or reals, \( X \) is the \( n \)-column-vector of program variables and \( b \) is an integer or a real.

4.3.3. Linear inequality tests

If \( C \) is of the form \( aX \leq b \) then \( P_t = (P \text{ and } aX \leq b) \) whereas \( P_f = (P \text{ and } aX > b) \). (Note that the inequality is not strict since \( P_f \) must be a closed polyhedron and that we can write \( aX \leq b + 1 \) for integers).

As above, the determination of the frames of \( P_t \) and \( P_f \) makes use of a frame \( (S', R', D') \) of the intersection \( \text{PnH of } P \) with the hyperplane \( H = \{X \in \mathbb{R}^n : aX = b\} \).

The set \( S_t \) of vertices of \( P_t \) is \( \{s \in S : s \leq X \} \cap S' \).

If \( S_t \) is empty then \( P_t \) is the empty polyhedron whereas \( P_f \) equals \( P \). Otherwise \( S_t \) is not empty and
$R_s = \{x, r\}, D_s = \emptyset$.

End of example

Incorporating the ray $r = (1,0)$ of $P_2$ in $A'X < B'$ by elimination of $\mu$ in $\{\mu > 0, A'x - \mu A'r < B'\}$ that is $\mu > 0, v = x, 1 = x_1, \alpha = x_2 > 0$ we get the convex-
The definition of the widening operation must be a balance between compelling the convergence of the global analysis of the program (by throwing away the restraints that do not quickly stabilize in the program cycles) and discovering as much information as possible about the program. Hence it is wise not to perform widening operations at loop junction nodes before gathering the information along the program cycles containing that loop junction node. Also the definition of the widening which we have given is a tentative one. The experiments that have been carried out seems to corroborate our choice but further studies are necessary to give a definite conclusion.

5. GLOBAL ANALYSIS OF PROGRAMS.

We illustrate the global analysis of programs on the following ad-hoc skeletal program which is simple enough to allow hand computations:

\[
\begin{align*}
\{P_0\} & \quad I:=2; \quad J:=0; \\
\{P_1\} & \quad L:\nonumber \\
\{P_2\} & \quad \text{if} \; J < 2 \; \text{then} \\
\{P_3\} & \quad I:=I+4; \\
\{P_4\} & \quad \text{else} \\
\{P_5\} & \quad J:=J+1; \quad I:=I+2; \\
\{P_6\} & \quad f; \\
\{P_7\} & \quad \text{go to} \; L;
\end{align*}
\]

The test involving some non-linear condition is not taken into account. Each assertion \(P^i\), \(i=0..7\) is initially the empty polyhedron \(\emptyset\) and the input assertion is propagated through the program graph:

\[
\begin{align*}
P^0_1 &= \mathbf{R}^2, \quad S=\{(0,0), R=\emptyset, D=\{(1,0),(0,1)\} \\
P^1_1 &= \mathbf{assign}(P^0_1), I=2; \quad J=0) \\
&= \{(I=2, J=0), S=\{(2,0)\}, R=\emptyset, D=\emptyset \\
P^2_2 &= \mathbf{assign}(P^1_1, P^0_2) = \mathbf{convex-hull}(P^1_1, P^0_2) = P^1_1 \\
P^3_1 &= P^1_1 = P^1_2 \\
P^4_1 &= \mathbf{assign}(P^3_1, I=I+4) \\
&= \{(I=6, J=0), S=\{(6,0)\}, R=\emptyset, D=\emptyset \\
P^5_1 &= \mathbf{assign}(P^4_1, J=J+1, I=I+2) \\
&= \{(I=4, J=1), S=\{(4,1)\}, R=\emptyset, D=\emptyset \\
P^6_1 &= \mathbf{convex-hull}(P^5_1, P^4_1) \\
&= \{(1+2J, 6, 4, 6, 0, 0)\}, S=\{(6,0),(4,1)\}, R=\emptyset, D=\emptyset \\
P^1_2 &= \mathbf{assign}(P^6_1, I=I+4) \\
&= \{(2+J, 2I, 0)\}, S=\{(2,0)\}, R=\{(1,0),(2,1)\}, D=\emptyset \\
P^3_2 &= P^1_2 = P^2_2 \\
P^4_2 &= \mathbf{assign}(P^3_2, I=I+4) \\
&= \{(2+J, 2I, 0)\}, S=\{(4,1)\}, R=\{(1,0),(2,1)\}, D=\emptyset \\
P^5_2 &= \mathbf{assign}(P^4_2, J=J+1, I=I+2) \\
&= \{(2+J, 2I, 0)\}, S=\{(4,1)\}, R=\{(1,0),(2,1)\}, D=\emptyset \\
P^6_2 &= \mathbf{convex-hull}(P^5_2, P^4_2) \\
&= \{(2+J, 2I, 0)\}, S=\{(4,1)\}, R=\{(1,0),(2,1)\}, D=\emptyset
\end{align*}
\]

When the loop body has been analyzed a widening operation takes place at the loop junction node \(L:\)

\[
P^1_2 = P^2_2 \vee \mathbf{convex-hull}(P^1_1, P^2_1)
\]

We have \(P^2_2 = \{(2+J, 2I, 0)\}, S=\{(2,0)\}, R=\{(1,0),(2,1)\}, D=\emptyset\) and \(\mathbf{convex-hull}(P^1_1, P^2_1) = \{(2+J, 2I, 0)\}\) with \(S=\{(2,0)\}, R=\{(1,0),(2,1)\}, D=\emptyset\), so that \(I=2J\leq 0\) which is the only constraint of \(P^2_2\) not verified by every element of the frame of \(\mathbf{convex-hull}(P^1_1, P^2_1)\) is eliminated by the widening operation.
Then \textit{convex-hull}(P_1,P_2) is included in P_2 so that
the program analysis has converged.

The final result shows up linear restraints among
the variables of the program that never appear ex-
plicitly in the program text and often escape the
notice of anyone studying this simple example:

\begin{itemize}
\item \{0\} : no information
\item \{1\} : i=2, j=0
\item \{2,\{3\},\{5\} : 2j+2s1, j\geq 0
\item \{4\} : 2j+6s1, j\geq 0
\item \{6\} : 2j+2s1, j\geq 1
\item \{7\} : 2j+2s1, 6s1+2j, j\geq 0
\end{itemize}

\section*{6. Example.}

On the next example (HEAPSORT, Knuth[1973,p.148])
it is not possible to trace the details of the
analysis so that we directly provide the results
produced by our experimental implementation:

\begin{verbatim}
procedure HEAPSORT(integer value N;
real array[1..N] T);
\end{verbatim}

\section*{7. Notes on the Experimental Implementation.}

We have produced an experimental implementation
written in PASCAL on the CII-IRIS 80 computer. The
length of the program is about 2500 lines.
The systems of equations and inequations have
array bounds. Taking account of such facts we
could propagate this information backward to the
loop junction nodes so that we would have a guide-
line for the widening operation. This would enable
us to combine the discovery and verification ap-
proaches.

Acknowledgements. We thank Radia COUSOT for
helpful discussions on linear programming and

M. SIMONNARD, Progammation Lininaire, Dunod, Paris,
(1973)

N. SUZUKI and K. ISHIHATA, Implementation of an array
bound checker, Conf. Record of the 4th ACM
Symposium on Principles of programming languages.
(Jan. 1977)
Conference Record
of the
FIFTH ANNUAL ACM SYMPOSIUM ON
PRINCIPLES OF PROGRAMMING LANGUAGES

Papers Presented at the Symposium
Tucson, Arizona
January 23-25, 1978

Sponsored by the
ASSOCIATION FOR COMPUTING MACHINERY
SPECIAL INTEREST GROUP ON AUTOMATA AND COMPUTABILITY THEORY
SPECIAL INTEREST GROUP ON PROGRAMMING LANGUAGES