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
  B=N;
  while B>1 do
  begin
    J:=1; T:=0;
    while J<(B-1) do
    begin
      if K[J]>K[J+1] then
      begin
        EXCHANGE(J,J+1); (no side effects on N,B,J,T)
      end;
    end;
    while T>0 do
    begin
      if T=0 then return false;
      B:=B-1;
      od;
    end;
    T:=J;
  end;
end;

Without user provided inductive assertions nor human interaction we have automatically determined (in 1.562 seconds of C.P.U. time) that the following restraints must hold among the variables of the above procedure:

\begin{align*}
\text{(1)} & : B=N \\
\text{(2)} & : 1\leq B < N \\
\text{(3)} & : J=1, \ T=0 \\
\text{(4),(5),(6)} & : B\leq N, \ T\leq 0, \ T+1\leq J, \ J+1 \leq B \\
\text{(7)} & : B\leq J, J+1 \leq B, \ T=J \\
\text{(8)} & : B\leq N, J+1 \leq B, J+2 \leq T, T+2 \leq J+1 \\
\text{(9)} & : B\leq N, J+1 \leq B, J+2 \leq T, T+2 \leq J+1 \\
\text{(10),(11)} & : B\leq N, J+1 \leq B, J+2 \leq T, T+2 \leq J+1 \\
\text{(12)} & : J\leq N, T\leq 0, \ T+1 \leq J, \ B\leq N, B+1 \leq J+1 \\
\text{(13)} & : B\leq N, B+1 \leq N
\end{align*}

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 computions 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 equality 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 relationships 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[I] to two elements of the same array refer to different storage locations (since e.g., I\geq 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 [1976]. His approach was based on Wegbreit's [1975] 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\leq x < 2\}, \ldots, \{1\leq x < n\}, \ldots).
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 that has been realized.

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

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

3.1. Linear system of a convex polyhedron.

Let \( x^1, \ldots, x^n \) 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

\[
\sum_{i=1}^{n} (a_{i1} x^1 + \cdots + a_{in} x^n) = b_i, \quad j = 1, \ldots, m
\]

(where \( a_{ij}, b_j \in \mathbb{R} \)), 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

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

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 \( \mathcal{C} \) of \( \mathbb{R}^n \) is said to be convex if and
only if \( \forall x_1, x_2 \in \mathcal{C}, \forall \lambda \in [0,1], \lambda x_1 + (1-\lambda) x_2 \in \mathcal{C} \). 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

A line of a polyhedron $P$ is a vector $d$ such that both $d$ and $-d$ are rays of $P$: $(\forall x \in P, \forall u \in \mathbb{R}, x + ud \in P)$. 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 $\{v_1, \ldots, v_n\}$ of $P$ and of a positive combination of the extreme rays $\{r_1, \ldots, r_p\}$ of $P$:

$$x = \sum_{i=1}^{n} \alpha_i v_i + \sum_{j=1}^{p} \beta_j r_j$$

The dimension of a polyhedron $P$ is the dimension of the least 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 $\sum_{i=1}^{n} (a_i x^i) \leq b^i$ be a linear inequality defining the polyhedron $P$. Then if the intersection of the hyperplane defined by $H = \{x : \sum_{i=1}^{n} (a_i x^i) = b^i\}$ with $P$ is not empty, it is a face of $P$.

We say that a point $s$ saturates the inequality $\sum_{i=1}^{n} (a_i x^i) \leq b^i$ if and only if $\sum_{i=1}^{n} (a_i s^i) = b^i$. We say that the ray $r$ saturates this inequality if and only if $\sum_{i=1}^{n} (a_i r^i) = 0$.

Let $\delta$ be the dimension of the greatest linear variety containing $P$. 
Hence this polyhedron is characterized by a system of linear restraints in $\mathbb{R}^{n+\sigma+\rho+\delta}$ upon the variables $\sigma, \Delta$. The first approximation $P_1^*\{s_i\}$ is defined by the system of restraints $\{x^1=1, x^2=-1, x^3=0\}$. The
3.4.1. Basic concepts of linear programming

A system of linear inequalities:
\[ \left\{ \begin{array}{c}
(a_i^1 x^1 + \ldots + a_i^m x^m) \leq b_i : j = 1, \ldots , m
\end{array} \right. \]
can be written in the equivalent form:
\[ \left\{ \begin{array}{c}
(x^1_i + x^2_i + \ldots + x^m_i) + y^1 = b^1, \quad y^1 \geq 0 : j = 1, \ldots , m.
\end{array} \right. \]

Hence a system of constraints can be written in standard form:
\[ \left\{ \begin{array}{c}
A x = B, \quad x \geq 0
\end{array} \right. \]
where \( A \) is a \( (n \times m) \) matrix, \( B \) is an \( m \)-vector \( E = (n+1, \ldots , n+m) \), \( F = \{1, \ldots , n\} \). The variables \( \{x_i : i \in F\} \) are the initial variables, \( \{x_i : i \in E\} \) are the slack variables.

A basis of the system \( [3.4.1.2] \) is a non singular \( n \times m \) submatrix \( A_I \) of \( A \). The system can be written in canonical form with respect to the basis \( A_I \):
\[ \left\{ \begin{array}{c}
x^I = A_I^{-1} A_J x^J = A_I^{-1} B, \quad x^I \geq 0 \quad \text{where} \quad J = \{1, \ldots , n+m\} - I.
\end{array} \right. \]
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 \( A_I^{-1} B \leq 0 \). Two feasible bases \( A_I \) and \( A_J \) are said to be adjacent if and only if the cardinal of the set \( I \cup J \) is equal to \( m \). If two bases are adjacent the classical pivot operation transforms the system written in canonical form with respect to \( A_I \) into an equivalent system in canonical form with respect to \( A_J \). The artificial basis method which is the initialization step of the simplex method transforms the system of constraints \( [3.4.1.2] \) into an equivalent system in canonical form \( [3.4.1.3] \) with respect to a feasible basis of \( [3.4.1.2] \) whenever such a basis exists.

3.4.2. Principles of Lannery's method.

The graph of the adjacency relation on the set \( \{A_I\} \) of feasible bases containing all initial variables (i.e. such that \( F=I \)) is connected. Hence given such a basis we can by successive pivoting operations and an exhaustive traversal technique find all feasible bases of a given system of restraints.
- If \( A_I \) is a feasible basis such that \( F=I \) then the vector \( A_I^{-1} B \) corresponds to a vertex of the convex polyhedron defined by the system of restraints. To each vertex of a polyhedron containing no line corresponds at least one feasible basis. If two vertices are adjacent then there are two adjacent feasible bases respectively corresponding to them.
- Let \( (A \times B, \ x \geq 0) \) be the canonical form of a system of restraints with respect to the feasible basis \( A_I \) containing all initial variables. Then Lannery shows that if a column \( i \in E-I \) satisfies the condition:
\[ \left\{ \begin{array}{c}
\{W(e[I, n]), \ A_i > 0\} \Rightarrow \{W(e[I, n]), \ A_i = 0\}
\end{array} \right. \]
exthen the vector \( x \in \mathbb{R}^n \) defined by \( x = -A_i^{-1} b \) where \( i \in \{1, n\} \) and \( j \) is the unique index such that \( A_i^j = 1 \) is an extreme ray of the polyhedron. Applying this result to each column \( i \) verifying \( [3.4.2.1] \) in each feasible basis corresponding to a vertex \( s \), we find all extreme rays adjacent to \( s \). Since each extreme ray of a polyhedron that contains no line is adjacent to a vertex we can find all extreme rays of such a polyhedron.

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

Hence let \( (A \times B, \ x \geq 0) \) be the canonical form with respect to any feasible basis \( A_I \). Let \( i \in E-I \) be a column satisfying the condition:
\[ \left\{ \begin{array}{c}
\{W(e[I, n]), \ W_k e[I, n], \ A_k > 0\} \Rightarrow \{A_k > 0 \ \text{or} \ \text{keF}\}
\end{array} \right. \]
Let \( d(i) \) be the vector of \( \mathbb{R}^n \) the \( i \)-th component of which is defined as:
\[ d(i)_i = \begin{cases} 1 & \text{if } i = i_0 \\ 0 & \text{otherwise} \end{cases} \]
\[ \text{else if } i \neq i_0 \text{ then } A_i \text{ is the unique index such that } A_i^j = 1 \]
\[ \end{cases} \]
Then it is shown that \( d(i) \) corresponds to a line of the polyhedron.

3.4.3. Algorithm for finding the frame of a convex polyhedron

Let \( A \times B \) be a system of \( m \) linear restraints among the variables \( x \in \mathbb{R}^n \).
- 1. Build the standard form \( A \{0\} x - B \{0\}, \ x \geq 0 \), where \( \mathbb{R}^n \).
- 2. Apply the first step of the simplex method. If there is no feasible basis, the polyhedron is empty. Otherwise we get the system \( A \{1\} x - B \{1\}, \ x \geq 0 \) in canonical form with respect to the feasible basis \( A_I \) with \( B \{1\} \geq 0 \).
- 3. While there exists an initial variable staying out of the basis and satisfying \( [3.4.2.2] \) perform a pivot operation which puts this variable in the basis by removing a slack variable from the basis.
- 4. We get a system \( A \{2\} x - B \{2\}, \ x \geq 0 \) in canonical form with respect to the basis \( I_2 \). Two subcases must be considered:
- 4.1. If all initial variables are in the basis \( (F=I_2) \), then the polyhedron contains no line and a vertex has been found. Then traverse exhaustively all feasible bases of the system and note at each step the vertex and the ray that are found. An efficient algorithm for this travel is given by Dyer-Pratt[1977].
- 4.2. The initial variables \( x^1, \ldots , x^l \) possibly remaining out of the basis verify \( [3.4.2.2] \). So \( \{d(i), \ldots , d(i_0)\} \) is a basis of the greatest linear variety contained in the polyhedron. The following system of restraints
\( \{x \leq 0, y \leq 0, z \leq 0\} \) defines a new polyhedron \( P' \) which is a section of the initial polyhedron. Applying the algorithm to this new system of constraints 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 constraints:

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

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

\[
\begin{array}{cccccc}
 x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
 1 & -1 & -1 & 0 & 0 & 0 \\
 1 & 0 & 0 & 1 & 0 & 0 \\
 1 & -1 & 1 & 0 & 0 & 1 \\
 -1 & 1 & 1 & 0 & 0 & 1
\end{array}
\]

The system \([S2]\) is in canonical form with respect to the infeasible basis \( A_{\{1,5,6,7\}} \).

2. The artificial basis method supplies the system \([S3]\) in canonical form with respect to the basis \( A_{\{1,5,6,7\}} \) which is feasible:

\[
\begin{array}{cccccc}
 x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
 0 & 1 & -1 & 1 & -1 & 0 \\
 0 & 1 & 0 & 0 & 1 & 0 \\
 0 & 1 & 1 & 0 & 0 & 1 \\
 1 & 0 & 0 & 0 & 0 & 0
\end{array}
\]

3. Then we put into the basis as many initial variables as we can, we get the basis \( A_{\{2,5,6,7\}} \):

\[
\begin{array}{cccccc}
 x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
 0 & 1 & -1 & 1 & -1 & 0 \\
 0 & 0 & 0 & 1 & 0 & 0 \\
 0 & 0 & 1 & 0 & 1 & 0 \\
 1 & 0 & 0 & 0 & 0 & 0
\end{array}
\]

4. The initial variable \( x_3 \) remains out of the basis and cannot be put into the basis without getting \( x_2 \) out. The third column verifies [3.4.2.2]. Hence the vector \( d = (0, 1, 1) \) is the only line of the polyhedron \( P' \). Let us build the system of constraints of a section \( P'' \) of \( P' \).

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

Applying (1) and (2) to [S5] we get the canonical form with respect to the feasible basis \( A_{\{2,3,5,6,7\}} \) which contains all initial variables. Thus we have found a first vertex of \( P'' \) and we start the traversal of the graph of vertices.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>X7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>-1/2</td>
<td>0</td>
<td>0</td>
<td>1/2</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>
</tbody>
</table>

Basis: \( A_{\{2,3,5,6,7\}} \)

Column 5 satisfies [3.4.2.1]

Ray: \( r_1 = (1, 1/2, -1/2) \)

Adjacent feasible basis: \( A_{\{2,3,5,6,7\}} \)

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>X7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</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>1</td>
<td>-1/2</td>
<td>1/2</td>
<td>0</td>
<td>1/2</td>
</tr>
</tbody>
</table>

Basis: \( A_{\{2,3,5,7\}} \)

Vertex: \( s_1 = (1, 1, -1/2) \)

No column satisfies [3.4.2.1]

Adjacent feasible bases: \( A_{\{2,3,5,6,7\}} \) already found and \( A_{\{2,3,5,7\}} \)

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>X7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1/2</td>
<td>0</td>
<td>-1/2</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>
</tbody>
</table>

Basis: \( A_{\{2,3,6,7\}} \)

Vertex: \( s_2 = (1, -1/2, 0) \)

No column satisfies [3.4.2.1]

Adjacent feasible bases: \( A_{\{2,3,5,6,7\}} \) already found and \( A_{\{2,3,6,7\}} \)

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>X7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1/2</td>
<td>1/2</td>
<td>0</td>
</tr>
</tbody>
</table>

Basis: \( A_{\{2,3,7\}} \)

Vertex: \( s_3 = (0, 3/2, 2/3) \)

Column 6 satisfies [3.4.2.1]

Ray: \( r_2 = (1, 0, 0) \)

Adjacent feasible basis: \( A_{\{2,3,5,6,7\}} \) already found

All feasible bases that contain initial variables have been found, the algorithm terminates with the following frame for \( P' \):

\[
\begin{align*}
S &= \{(1, 1/2, -1/2), (1, -1/2, 1/2), (3, -3/2, 3/2)\}, \\
R &= \{(1, 1/2, -1/2), (1, 1, 0)\}, \\
D &= \{(0, 1, 1)\}
\end{align*}
\]

End of example.

### 3.4.4. Simplification of a frame

Lenary [1986] proposes a method for eliminating all irrelevant members of a frame when a system of constraints of the polyhedron is known. This method is the dual of the one given at paragraph 3.3.1.2. For simplifying a system of inequalities, Lenary's method is based on the following results:

- A vertex or a ray saturating no inequality is irrelevant.
- A ray saturating all constraints corresponds to a line.
- If \( e_1 \) and \( e_2 \) are two vertices or two rays (which are not lines) in the frame the quasi-ordering \( \leq \) is defined by:

  \( e_1 \leq e_2 \) if every inequality satisfied by \( e_1 \) is satisfied by \( e_2 \).

- If \( e_1 \leq e_2 \) and \( e_2 \not\leq e_1 \), then:

  - \( e_1 \leq e_2 \) and \( e_2 \not\leq e_1 \) imply that \( e_1 \) is irrelevant.
  - \( e_1 \leq e_2 \) and \( e_2 \not\leq e_1 \) imply that one but only one of the two elements may be eliminated.
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], Klei[1984]. 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 \( P_1 \subset P_2 \) if and only if each element of the frame of \( P_1 \) satisfies the restraint 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 simplications. 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 correctness of these transformations apply to the specific case analyzed here.

4.1. Program entry node.

In some cases (parameters of procedures) the restraints on the input variables may be known. In this case they constitute the assertion associated with the input arc. Otherwise the variables are assumed not to be initialized so that they satisfy no restraints. Hence the vector of their values may be any point of \( \mathbb{R}^n \) where \( n \) is the number of variables involved in the program analysis. The corresponding polyhedron is given by a system with no restraints and equivalently by the frame consisting in:

- The vertex which is the origin of \( \mathbb{R}^n \)
- No ray
- The lines \( d_1, \ldots, d_m \) such that for every \( i=1, \ldots, n \), \( d_1^i=1 \) and \( d_j^i=0 \) for \( j=1, \ldots, n \) and \( j \neq i \).

4.2. Assignment.

Let \( \{A,B,S,R,D\} \) be the representation of the input assertion \( P \) corresponding to the polyhedron \( AX \subset B \) the frame of which is \( S=(S_1 : i=1 \ldots c), R=(r_j : j=1 \ldots p), D=(d_k : k=1 \ldots q) \). Then the representation of the output assertion \( P' \) after the assignment \( X' = E(X) \) is given by

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

4.2.1. Assignment of a non linear expression.

If the expression \( E(X) \) is not linear it is assumed that any value of \( X \) can be assigned to \( X' \).

Hence after the assignment \( X' = E(X) \) nothing is known about the value of \( X' \). Therefore \( X' \) is eliminated from the system of restraints \( AX \subset B \) by a projection along the column \( l_0 \) (§ 3.3.1.1). For the frame representation this consists in adding the line \( d \) (such that \( d_1^i = 1 \) and \( d_j^i = 0 \) for every \( j=1 \ldots n \) different from \( l_0 \)) to the line of restraints. 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:

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

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

The output assertion \( P' \) after the assignment \( X' = X_1 \cdot X_2 \) is \( (X_1 \geq 0, X_1 \geq 2) \), the frame of which is \( S' = \{(2,3),(4,1)\}, R' = \{(1,1),(1,0)\}, D' = \{(0,1)\} \). Simplifying we get \( (X_1 \geq 2), S' = \{(2,3)\}, R' = \{(1,0)\}, D' = \{(0,1)\} \) that is:

```
  x1
       /
      /  
     /    
    /     
   /     
 x2
```

90
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:

\[
\begin{align*}
  x_2 & \leq x_1^2 x_2^2 \\
  x_2 & \geq x_1 \\
  x_2 & \geq 5x_1 - x_1^2
\end{align*}
\]

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 $x_2 > 0$ and $x_2 x > 0$ imply $x_1 x_2 > 0$ or the assignment $x := x_2$ 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 $x_2 := \sum_{i=1}^{n} (a_i x_i^2) + b$ where $a$ 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:

- $S' = \{s'_1, \ldots, s'_0\}$ where $s'_i$ is defined by $s'_1 = a_1 x_1^2 + b$ and $s'_i = a_i x_i$ where $i = 1..n$ with $i = 1..l_1$.
- $R' = \{r'_1, \ldots, r'\}$ where $r'_i$ is the vector defined by $r'_i = a_i x_i$ and $r'_i = r_1$ for $i = 1..n$ and $i = 1..l_0$.
- $D' = \{d'_1, \ldots, d'\}$ where $d'_i$ is the vector defined by $d'_i = a_i x_i$ and $d'_i = a_i x_i^2$ for $i = 1..n$ and $i = 1..l_1$.

leads to $\{x_2 \geq 0\}$ so that the output system of restraints is:

\[
\begin{align*}
  x_2 & \geq 2 \\
  x_2 x_1 & = 1
\end{align*}
\]

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

\[
\begin{align*}
  2x_2 - 2x_2 & \geq 3 x_2
\end{align*}
\]
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 \( P_f = P \) and \( \neg C \).

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 \( P_f = (P \text{ and } aX \geq b) \).(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 \( \cap H \cap P \) with the hyperplane \( H = \{ x \in \mathbb{R}^n : aX = b \} \).

The set \( S_t \) of vertices of \( P_t \) is \( \{ s \in S : as < b \} \cup S' \).

If \( S_t \) is empty then \( P_t \) is the empty polyhedron whereas \( P_f \) equals \( P \).

Otherwise \( S_t \) is not empty and the set \( R_t \) of extreme rays of \( P_t \) is \[ \{ r \in R : ar < 0 \} \cup \{ d \in D : ad < 0 \} \cup \{ -d \in D : ad > 0 \} \cup R' \]. The set of
4.4. Simple junction nodes.

Simple junction nodes correspond for example to the merge of the "then" and "else" paths in a conditional statement. More generally let $P_1, P_2, ..., P_p$ be the assertions associated with the input arcs to the simple junction node and $P$ the output assertion:

![Diagram of a simple junction node]

Then the output assertion $OK_{i=1}^p (P_i)$ does not necessarily correspond to a convex polyhedron so that it must be approximated by the convex-hull of $P_1, ..., P_p$ which is the least polyhedron containing $P_1, ..., P_p$.

Since the convex-hull operation is associative we can assume without loss of generality that there are two input arcs ($p=2$). Given $P_1(A_1, B_1, S_1, R_1, D_1)$ and $P_2(A_2, B_2, S_2, R_2, D_2)$ the frame of $P$ is $convex-hull (P_1, P_2)$ is $S_1S_2S_3$, $R=R_1R_2$ and $D=D_1D_2$. The system of restraints $AXSB$ describing $P$ is obtained by the convex-hull of the frame $(S,R,D)$ which can be computed by successive approximations (§ 3.3.1.1). As soon as the system of restraints $(AXSB)$ of $P$ is known the frame $(S,R,D)$ of $P$ can be simplified (§ 3.4.4).

Notice that $convex-hull (P_1, P_2)$ cannot be computed directly from the systems of restraints $A_iXB$, and $A_jXB$. In order to avoid a costly conversion (§ 3.4) the redundant frame representation has been kept along with the restraints representation. For the output assertion $P$ the conversion from frame $(S,R,D)$ to restraints $AXSB$ representation is less expensive (§ 3.3). Since the system of restraints of $P_1$ (or $P_2$) is known this conversion is optimized as follows:

$convex-hull (convex-hull (A_1XB, \{S_1, R_1, D_1\}), A_2XB, \{S_2, R_2, D_2\}) = convex-hull (convex-hull (S_1S_2S_3), A_1XB, \{S_2, R_2, D_2\})$

so that starting from $A_1XB$ we can successively incorporate in the convex-hull the elements of the frame of $P_2$.

Example:

$P_1 = \{(x_1 \geq 0, x_2 \geq 0, x_1 + x_2 \leq 1), S_1 = \{(0,0), (1,0), (0,1)\}, R_1 = D_1 = \emptyset\}$

$P_2 = \{(x_1 \leq 2, x_2 = 2), S_2 = \{(1,2)\}, R_2 = \{(1,0)\}, D_2 = \emptyset\}$

The convex-hull of $P_1$ (given by $A_1XB$) and the vertex $v = \{(1,2)\}$ of $P_2$ is obtained (§ 3.3.1.1) by elimination of $\lambda$ in $\{0Sx1, A_1x1-A_1s1, A_1x2-A_1s2\}$. Eliminating $\lambda$ in $\{0Sx1, x_1 + x_2 \leq 2, x_1 \geq 2\}$ we get the approximation $A'XSB$ given by $\{x_2 \geq s1, 0Sx1, x_2 \geq 0\}$.

Incorporating the ray $r = (1,0)$ of $P_2$ in $A'XSB$ by elimination of $\mu$ in $\{\mu s0, A'x-\mu A'r=\emptyset\}$ that is $\{\mu s0, x_2+x_1-\mu s, 0Sx_1-\mu s, 1, x_2=0\}$ we get the convex-hull of $P_1$ and $P_2$ given by $\{0Sx_2s2, x_1 \geq 0, x_2 \geq x_1s1\}$.

End of example.

4.5. Loop junction nodes.

Each cycle in the program graph contains a loop junction node:

![Diagram of a loop junction node]

The corresponding transformation $P = P\text{\textasciitilde}{\text{loop-hull}}(P_1, P_2, ..., P_p)$ is defined by the widening operation $\text{\textasciitilde}$. Let $Q_1(A_1, B_1, S_1, R_1, D_1)$ and $Q_2(A_2, B_2, S_2, R_2, D_2)$ be two convex polyhedra. Then $Q_1 \text{\textasciitilde} Q_2$ is the convex polyhedron consisting in the linear restraints of $Q_1$ verified by every element of the frame $(S_2, R_2, D_2)$ of $Q_2$.

Example:

$Q_1 = \{(-x_1+2x_2 \leq -2, x_1+2x_2 \leq 0, x_2 \geq 0)\}$

$S = \{(2,0), (6,0), (4,1)\}, R = D = \emptyset$  

$Q_2 = \{(-x_1+2x_2 \leq -2, x_1+2x_2 \leq 0, x_2 \geq 0)\}$

$S = \{(2,0), (10,0), (6,2)\}, R = D = \emptyset$  

$Q_1 \text{\textasciitilde} Q_2 = \{(-x_1+2x_2 \leq -2, x_2 \geq 0)\}$

End of example.

In order to compute the frame of $Q_1 \text{\textasciitilde} Q_2$ it is necessary to convert from the linear restraint representation. Laney's method (§ 3.4) is known to be expensive. However when applied after a widening operation the cost remains reasonable because of the following arguments:

- In the widening $Q_1 \text{\textasciitilde} Q_2$ of $Q_1$ by $Q_2$ a certain number of vertices of $Q_1$ have been replaced by extreme rays. Hence the widening operation eliminates vertices, so that $Q_1 \text{\textasciitilde} Q_2$ can be assumed to have a small number of vertices. Now it is the discovery of vertices which is the most costly in Laney's method.

- A frame of $Q_1$ is known and $Q_1$ is included in $Q_1 \text{\textasciitilde} Q_2$. Therefore the initialization of the simplex method needs not to be used. Moreover since any restraint of $Q_1 \text{\textasciitilde} Q_2$ is a restraint of $Q_1$ it is highly probable that the frames of $Q_1$ and $Q_1 \text{\textasciitilde} Q_2$ have numerous common elements.

The correctness criterion of Cousot[1977] recalled at paragraph 2 is satisfied since $Q_1 \text{\textasciitilde} Q_2$, $Q_2 \text{\textasciitilde} Q_3 \text{\textasciitilde} Q_2$, and for every chain $C_1 C_2 \cdots C_n < C_n \cdots$ the chain $S_1 S_2 \cdots S_{n-1}$, $S_n < S_{n-1} \text{\textasciitilde} C_n$, ... is not an infinite strictly increasing chain since at each step $n$ the number of restraints describing $S_n$ is finite and less than or equal to the number of restraints describing $S_{n-1}$.
The definition of the widening operation must be \( w^2 = \text{some}\ \text{operation}\ \alpha^1 \alpha^2 \).
Then \( \text{convex-hull}(P_i, P_j) \) is included in \( P_2 \) so that the program analysis has converged.

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

\[
\begin{align*}
[0] & : \text{no information} \\
[1] & : I=2, J=0 \\
[2],[3],[5] & : 2J+2<R, J>0 \\
[4] & : 2I+6<1, J=0 \\
[6] & : 2J+2<R, J>1 \\
[7] & : 2J+2<R, 6I+2J, J>0 \\
[8] & : J=0, 2R+1<sN \\
[9] & : J=2, 2<4N, 5+2R+3<sN, L=1, \\
[10] & : 15, 7<sN \\
[12] & : J=2, 2<sN, 4<s2N, 5+2R+3<sN, L=1, \\
& \quad L<sN
\end{align*}
\]

6. EXAMPLE.

On the next example (HEAPSRT, 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:

\[
\begin{align*}
\text{procedure HEAPSRT}(\text{integer value } N; \text{real array } [1..N] \ T;)
\end{align*}
\]

\[
\begin{align*}
& \begin{align*}
& \{1\} \quad \text{begin integer } L, R, I, J, \text{ real } K; \\
& \{2\} \quad L:= \text{N dlo2} + 1; \ R:=N; \\
& \{3\} \quad \text{if } (L>2) \text{ then } \\
& \{4\} \quad L:= L-1; \ K:=T[L]; \\
& \{5\} \quad \text{else } \\
& \{6\} \quad \text{if } (L>R) \text{ do } \\
& \{7\} \quad \text{while } (R>2) \text{ do } \\
& \{8\} \quad I:=L; \ J:=2*I; \\
& \{9\} \quad \text{while } (3>R) \text{ do } \\
& \{10\} \quad \text{if } (3>R-1) \text{ then } \\
& \{11\} \quad \text{if } (T[I] < T[J+1]) \text{ then } J:=J+1 \ f: f; \\
& \{12\} \quad \text{if } (K>T[J]) \text{ then } \\
& \{13\} \quad \text{exit (of the inner loop f); \\
& \{14\} \quad I:=I+1; \ J:=2*J; \\
& \{15\} \quad \text{odl; } \\
& \{16\} \quad \text{if } L>2 \text{ then } \\
& \{17\} \quad L:=L-1; \ K:=T[L]; \\
& \{18\} \quad \text{else } \\
& \{19\} \quad \text{if } (T[R] < T[I]) \text{ then } R:=R-1; \\
& \{20\} \quad f, \\
& \{21\} \quad (T[J]+K); \\
& \{22\} \quad od; \\
& \{23\} \quad end;
\end{align*}
\end{align*}
\]

The procedure is analyzed with the input specification Nx2 (see Knuth[1973], p.146). This analysis does not take account of the statements involving operations on arrays, these statements such as \( \{K:=T[L]\} \) have been bracketed in the text of the

\[
\begin{align*}
\{15\} & : J=2, 2<sN, 4<s2N, 5+2R+3<sN, L=1, \\
& \quad L<sN, 5+2R+3<s2N, 5+2R+3<sN, L<sN
\end{align*}
\]

Once the above invariant assertions have been discovered it is very easy using projections \((3.3.1.1)\) to check statically that all array accesses are correct. Notice that some relationships among the variables of the procedure are not obvious and cannot be discovered by hand without deep understanding of the program.

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 inequalities have been represented for simplicity by real matrices and this sometimes results in a loss of precision. It seems very difficult to write the program so that this loss of precision is acceptable that is so that the relationships which have been found correspond to a domain including any value that each real variable can take during any execution of the analyzed program. However the main applications we have in mind (such as array bound checking) deal with integer variables. In this case the coefficients of the linear restraints are rationals, which can be represented as fractions \(p/q\) where \(p\) and \(q\) are integers. Then the operations which are performed on these coefficients \(\pm, \times, /\) are more costly but introduce no loss of precision.

We have noticed that programs involving numerical constants are better handled when these numerical constants are replaced by the declaration of a symbolic constant. It is often the case that the convergence of the analysis is faster although the systems of restraints are bigger.

It seems to be very difficult to evaluate the cost of the analysis of a program. The cost of the analysis does not only depend on the length (number of lines) of the program but mainly on the complexity of the program graph (number of loops, degree of loop combination, etc...). It seems that the cost of an analysis is almost linear in the length of the program but exponential in the number of variables involved in the analysis. From this point of view nested static scopes (such as \( A[J]\))
array bounds. Taking account of such facts we could propagate this information backward to the loop junction nodes so that we would have a guideline for the widening operation. This would enable us to combine the discovery and verification approaches.

Acknowledgements. We thank Radha Cousot for helpful discussions on linear programming and Mrs H. Diaz for typing the manuscript.

8. REFERENCES


M. Simonnard, Programmation Linéaire, Dunod, Paris, [1973]


