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 a program.

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

```
procedure BUBBLESORT(integer value N; integer array[1:N] k);
begin
  i:=N;
  while i>1 do
    if k[i]>k[i-1] then
      EXCHANGE(k,i,i-1); [no side effects on N,B,i,T]
    end;
end;
```

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 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 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[i+1] to two elements of the same array refer to different storage locations (since e.g. I>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[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 corresponding 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.

3. FORMAL REPRESENTATIONS OF LINEAR RESTRAINTS 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
\[ \sum_{j=1}^{n} \alpha_j^i x^j - b^i = 0 \quad (i=1, \ldots, m) \]
where $\alpha_j^i \in \mathbb{R}$, 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
\[ ax^1 + \cdots + ax^n \leq b \]
for $a \in \mathbb{R}$ is a closed half-space of $\mathbb{R}^n$. For
A line of a polyhedron \( P \) is a vector \( \mathbf{d} \) such that both \( \mathbf{d} \) and \(-\mathbf{d}\) are rays of \( P \): \( (\forall \mathbf{x} \in P, \forall \mu \in \mathbb{R}, \mu \mathbf{x} + \mathbf{d} \in P) \). 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

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-faces). Let \( \sum_{i=1}^{n} (a_i x_i) \leq b \) be a linear inequality defining the polyhedron \( P \). Then if the
Hence this polyhedron is characterized by a system of linear restraints in $\mathbb{R}^{n+d}$ upon the variables $x_1, x_2, \ldots, x_n, \lambda_1, \lambda_2, \ldots, \lambda_d, \mu_j, \nu_k.$ Eliminating the $\lambda_i, \mu_j, \nu_k$ we get a system of linear restraints in $\mathbb{R}^n$. This elimination can always be done by the projection operation used in Kuhn[1956].

Let us represent the system of m inequalities by $Ax \leq b$ where $A$ is an $m \times n$ real matrix and $b$ a $m$-vector. In order to eliminate the variable $x_c$, we project according to the column $c$ upon $Ax \leq b$ in order to get a simplified projected system defined as follows:

- For each row $A^T_c$ such $A^T_c \neq 0$, the restraint $A^T_c x \leq b$ is part of the projected system.

The first approximation $P_1 \{x_1\}$ is defined by the system of restraints $(x^1=1, x^2=-1, x^3=0)$. The second approximation $P_2$ is the convex-hull of $P_1$ and $\{s_2\}$ that is $(0s_1s_1, x^1-2s_1=-1, x^2+2s_2=1, x^3=0).$ Eliminating $\lambda$ we get $(x^3=0, x^1+x^2=0, -1s_1 \leq 1, -1s_2 \leq 1)$. $P_3$ is obtained by adjoining the ray $(0,0,1)$ to $P_2$ that is $(\mu \geq 0, x^1+x^2=0, -1s_1 \leq 1, -1s_2 \leq 1, x^3=0)$. Eliminating $\mu$ we get $(x^1+x^3=0, -1s_1 \leq 1, -1s_2 \leq 1, x^3 \geq 0)$. The last approximation $P_4$ is the convex-hull of $P_3$ and the line $(1,1,0)$ that is $(x^1+x^2-2v=0, -1s_1 \leq 1, -1s_2 \leq 1, x^3 \geq 0)$. Eliminating $v$ we get the final solution $(x^3 \geq 0, -2 \leq x^1, x^2 \leq 2).$
3.4.1 Basic concepts of linear programming

feasible basis corresponding to a vertex $s$, we find all extreme rays adjacent to $s$. Since each extreme...
\{Ax \leq b, \sum \alpha_i d_i \leq D, \forall k = 1, \ldots, 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:

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

Using the matrix notations for denoting systems of equations the standard form of [S1] is:

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

The system [S2] is in canonical form with respect to the infeasible basis \( A_{[4,5,6,7]} \).

2. The artificial basis method supplies the system [S3] in canonical form with respect to the basis \( A_{[4,5,6,7]} \) which is feasible:

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

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

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

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 \( P \). We thus build the system of restraints of a section \( P' \) of \( P \).

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

**Basis:** \( A_{[2,3,1,4,5]} \)

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

**Column 5 satisfies** [3.4.2.1]

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

**Adjacent feasible basis:** \( A_{[2,3,1,4,5]} \) already found and \( A_{[2,3,1,4,5]} \) can be found.

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

**Basis:** \( A_{[2,3,1,4,5]} \)

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

**No column satisfies** [3.4.2.1]

**Adjacent feasible basis:** \( A_{[2,3,1,4,5]} \) already found and \( A_{[2,3,1,4,5]} \) can be found.

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

**Basis:** \( A_{[2,3,1,4,5]} \)

**Vertex:** \( s_3 = (3, -3/2, 3/2) \)

**Column 6 satisfies** [3.4.2.1]

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

**Adjacent feasible basis:** \( A_{[2,3,1,4,5]} \) already found and \( A_{[2,3,1,4,5]} \) can be found.

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

\[
[S = \{(1, 1/2, -1/2), (1, 1/2, 1/2), (2, 3, -3/2, 3/2), \}, \]

\[
R = \{(1, 1/2, -1/2), (1, 0, 0), \}
\]

**End of example.**

### 3.4.4. Simplification of a frame

Lamy [1968] proposes a method for eliminating all irrelevant members of a frame when a system of restraints of the polyhedron is known. This method is the dual of the one given at paragraph 3.3.1.2.
3.5. On the use of two representations for assertions.  4.2. Assignment.
The approximation is obviously very coarse since the substitution of \( x_1 \cdot x_2 \) 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 \( x_1 \geq 0 \) and \( x_2 \geq 0 \) implies \( x_1 x_2 \geq 0 \) or the assignment \( x := y \cdot x_2 \) 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 \( X^{1,2} := \sum_{i=1}^{n} (a_i x_i^2) + b \) for a 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

\[- x_1^{1,2} = (X_1^{1,2} - \sum_{i=1}^{n} (a_i x_i^2)) - b/a_{1,2} \]

Also \( (AX \leq b) \) is equivalent to \( ((MA)X \leq (B-AK)) \) which leads to the output system of restraints satisfied by \( X' \).

4.2.2.2. Non-invertible assignments

If \( a_{1,2} = 0 \) then we cannot solve \( X' \) in terms of \( X \) so that some information is lost by the assignment. Hence \( X^{1,2} \) 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 \( X^{1,2} = aX + b \) is adjoined to the resulting system.

Example:

Let \( P \) be the input assertion defined by

\[
\begin{align*}
    x_2 & \leq 1 \\
    x_1 + x_2 & \geq 5 \\
    x_2 - x_1 & \geq -1
\end{align*}
\]

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:

\[
\begin{align*}
    x_1 & \geq 2 \\
    x_2 - x_1 & = 1
\end{align*}
\]
with the input arc to a decision node testing some boolean condition C. Let \( P_r' \), \( S_r', R_r', D_r' \) and \( P_f, S_f, R_f, D_f \) be the assertions associated respectively with the true and false exits of the test. Obviously \( P_r' = P \) 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 \( ax \geq b \) where \( a \) is an \( n \)-column-vector of integers or reals, \( X \) is the \( n \)-column-vector of program variables and \( b \) is an integer or a real.

4.3.1. Non-linear tests.

If C is not a linear condition we "ignore" the test by putting \( P_t = F_r = P \). This is certainly valid, but it may not be as much information as could be gathered. As for non-linear assignments specific studies can be made of how best to handle test conditions which are not linear (for example \( \log(x) \geq 0 \) implies \( x \geq 1 \)).

4.3.2. Linear equality tests.

When C is a linear condition let \( H \) be the hyperplane \( \{ x \in \mathbb{R}^n : ax=b \} \). If C is of the form \( ax=b \) then either \( P \) is included in \( H \) in which case \( P_t = P \) and \( P_f = \emptyset \) or \( P \) is not included in \( H \) in which case \( P_t = P \) and \( P_f = P \). Note that \( \{ ax=b \} \) is approximated by \( \{ ax \leq b \} \) since the domain \( \{ x \in \mathbb{R}^n : ax \leq b \} \) is not, in general, a closed convex polyhedron.

The frame of \( P \) is found thanks to the following results:

1. Each vertex \( s \) of \( P \) lies on an edge of \( P \) according to the following two alternatives:
   - 1.1. \( s \) is a vertex \( s \) of \( P \) that belongs to \( H \).
   - 1.2. there are two adjacent vertices \( s_1 \) and \( s_2 \) of \( P \) such that \( s = \lambda s_1 + (1-\lambda) s_2 \) where \( \lambda = (b-a s_2)/(a s_1 - a s_2) \) belongs to \( [0,1] \).

2. There are a vertex \( s \) and a ray \( r \) of \( P \) such that \( s = a r \) and either \( s = r \) or \( s = t \) are adjacent to \( r \) in \( P \) and \( s = s r \) where \( \mu = (b-a s)/(a r) \) is positive.

3. Finally a vector \( d \) is a line of \( P \) if and only if there are two lines \( d_1 \) and \( d_2 \) in \( P \) such that \( d = d_1 - t d_2 \).

In order to determine a frame of \( P \) it is necessary to know the adjacent elements of the frame of \( P' \). (s1, r1), (s2, r2), (r1, r2) are adjacent. The vertex \( s' \) of \( H \) corresponds to the case 4.3.2.1.3, that is \( s' = s_2 + (b-a r_2)/(a r_2) \).

The example of \( P' \) corresponds to case 4.3.2.2.2 that is \( s = s_1 + (b-a r_1)/(a r_1) \).

A frame of \( P \) is given by \( S_t = \{ r_1, s' \}, R_t = \{ r_1, r' \} \), \( D_t = \emptyset \). A frame of \( P \) is given by \( S_t = \{ s' \} \),

4.3.3. Linear inequality tests.

If C is of the form \( ax \leq b \) then \( P_t = (P \) and \( ax \geq b \) \)

where \( P_t = (P \) and \( ax \geq b \).

Note that the inequality is not strict since \( P_t \) must be a closed polyhedron and that we can write \( ax \geq b + 1 \) for integers.

As above, the determination of the frames of \( P_t \) and \( P_f \) uses use of a frame \( (S', R', D') \) of the intersection \( P \) of \( P \) with the hyperplane \( H = \{ x : ax = b \} \).

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

If \( S_t' \) is empty then \( P_t \) is the empty polyhedron whereas \( P_t \) equals \( P \). Otherwise \( S_t' \) is not empty and the set \( R_t \) of extreme rays of \( P_t \) is \( \{ \{ r \in R : ar = b \} \} \).

The set lines \( D_t \) of \( P_t \) is \( D' \). A symmetric reasoning is used to determine a frame of \( P_f \).

The resulting frames \( (S_t', R_t, D_t) \) and \( (S_f', R_f, D_f) \) are not necessarily minimal and must be simplified (4.3.4).

Example: Let \( P \) be an input assertion defined by \( \{ (x_1, x_2) : x_1 + x_2 \leq 5, x_1 - x_2 = 1 \} \), \( S = \{ s_1 = (4, 1), s_2 = (2, 3) \} \), \( R' = \{ r_1 = (1, 1), r_2 = (1, 1) \} \), \( D = \emptyset \). Let \( P_t \) and \( P_f \) be the output assertions associated respectively with the true and false exits of a test on the condition \( x_1 + 2x_2 = 1 \) that is \( x \leq 3 \) where \( a = (1, 2) \).

The geometric interpretation of the above:
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 junctions containing that loop junction node. Also the definition of the widening which we have given is a tentative one. The experimentations that have been carried out seem 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:

\[
\begin{align*}
\{P_0\} & \quad I := 2; J := 0; \\
\{P_1\} & \quad L_1; \\
\{P_2\} & \quad \text{if } \ldots \text{ then}
\end{align*}
\]

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

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

\[
(2J+2I, I+2J \leq 6, 0 \leq J), S_0 = \{(2,0), (10,0), (8,1)\}, R = \emptyset, O = \emptyset
\]

\[
P^3 = P^2 \vee \text{convex-hull}(P^2_1, P^2_2)
\]

\[
(2J+2I, I+2J \leq 10, 0 \leq J), S_0 = \{(6,2), (6,1), (4,1)\}, R = \emptyset, O = \emptyset
\]

\[
P^4 = \text{assign}(\text{assign}(P^2_3, J := J+1), I := I+2)
\]

\[
(2J+2I, I+2J \leq 10, 0 \leq J), S_0 = \{(6,2), (6,1), (4,1)\}, R = \emptyset, O = \emptyset
\]

\[
P^5 = \text{assign}(\text{assign}(P^2_3, J := J+1), I := I+2)
\]

\[
(2J+2I, I+2J \leq 10, 0 \leq J), S_0 = \{(6,2), (6,1), (4,1)\}, R = \emptyset, O = \emptyset
\]

We have \(P^2 = (2J+2I, I+2J \leq 6, 0 \leq J)\) and \(\text{convex-hull}(P^2_1, P^2_2) = (2J+2I, I+2J \leq 10, 0 \leq J)\) with \(S_0 = \{(2,0), (10,0), (6,2)\}, R = \emptyset, O = \emptyset\), so that I+2J \leq 6 which is the only constraint of \(P^2_2\) not verified by every element of the frame of \(\text{convex-hull}(P^2_1, P^2_2)\) is eliminated by the widening operation.

\[
J \quad \text{convex-hull}(P^1_1, P^1_2)
\]
Then \( \text{convex-hull}(P_1, P_2) \) 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 explicitly in the program text and often escape the notice of anyone studying this simple example:

\[
\begin{align*}
(0) & : \text{no information} \\
(1) & : 1 \geq 2, J = 0 \\
(2), (3), (5) & : 2J \geq 2I, J \geq 0 \\
(4) & : 2J \geq 6I, J \geq 0 \\
(6) & : 2J \geq 2I, J \geq 1 \\
(7) & : 2J \geq 2I, 6I \geq 2J, J \geq 0
\end{align*}
\]

\[
\begin{align*}
(15) & : J \geq 2J, R, 2J \geq 2I+2I+1, R \geq 22N, 1 \leq L, \\
(16) & : 2J \geq 2I+1, R \geq 22N, 1 \leq L, \\
(17) & : \{15\}, L \geq 2 \\
(19) & : \{15\}, L \geq 2 \\
(21) & : R \geq 1, 2I \leq 2N, R \geq 2N, 2L + 2I + 3 \leq 3N, L \geq 1, \\
& \text{RsN}
\end{align*}
\]

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.

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

7. NOTES ON THE EXPERIMENTAL IMPLEMENTATION.
Acknowledgements. We thank Radhia Cousot for helpful discussions on linear programming and Mrs H. Diaz for typing the manuscript.

8. REFERENCES

M.L. Balinski, An algorithm for finding all vertices

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


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