AN INTRODUCTION TO A MATHEMATICAL THEORY OF GLOBAL PROGRAM ANALYSIS

Patrick M. Cousot

Laboratoire d'Informatique
U.S.M.G., BP.53
38041 Grenoble Cedex
France

(March 1977)

1. INTRODUCTION

Performing compile-time optimization of programs (Aho & Ullman[73], Branquart et al.[76], Cocke & Schwartz[69], Hecht[75], Schaefer[73], Wulf et al.[75]) involves an *analysis of the program* (the determination and collection of information which is distributed throughout the program (Ullman [75])) followed by a *transformation of the program* (the application of those program transformation rules which according to the previous analysis can be shown to lead to an equivalent but improved transformed program).

Attaché de Recherche au C.N.R.S., Laboratoire Associé No. 7.
This work was partially supported by IRIA-SESORI grant 76-160.

This paper informally exposes and examplifies the *abstract interpretation of programs* (Cousot[77a]), a lattice theoretic model which in particular is suitable to treat all global program analysis problems.

2. DETERMINATION OF INVARIANT PROPERTIES OF PROGRAMS

Roughly speaking, global program analysis requires the determination, for each program point π of an invariant property P_π known to hold each time control reaches π during execution, independently of the path taken to reach the program point π.

Example : a fairly simple case of program analysis and optimization occurs when constant computations are evaluated at compile time (Kam & Ullman[76], Kam & Ullman[77], Kildall[73], Reif & Lewis[77]).
Consider the following skeletal program :

```
          (a := 1, b := 2, c := 3, d := 3, e := 0);
    {1}
          while ... do
    {2}
               (b := 2 * a, d := d + 1, e := e - a);
    {3}
               (a := b - a, c := e + d);
    {4}
          od;
```

If we can determine that the set P_2 of variable states at program point {2} is :

$$P_2 = \{<a = 1,\ b = 2,\ c = 3,\ d = 3+i,\ e = -i> \mid i \geq 0\}$$

we have shown that a, b and c have constant values at program point {2} during execution. *End of Example.*

3. FINDING INVARIANT PROPERTIES AS SOLUTION TO A SYSTEM OF EQUATIONS

According to the semantics of the utilized programming language, the assertion P_i associated with program point $\{i\}$ is a function f_i of the other assertions $P_1, \ldots, P_n$ associated with the various points $\{1\}, \ldots, \{n\}$ of the program. Therefore the desired properties $P_1, \ldots, P_n$ must be one of the solutions to a system of mutually recursive equations :

$$\begin{cases} X_1 = f_1(X_1, \ldots, X_n) \\ \ldots \\ X_n = f_n(X_1, \ldots, X_n) \end{cases}$$

abbreviated in $X = F(X)$.

Example : The set P_1, P_2, P_3, P_4 of variable states are related in the example program as follows :

$$\begin{cases} P_1 = \{<a = 1,\ b = 2,\ c = 3,\ d = 3,\ e = 0>\} \\ P_2 = P_1 \cup P_4 \\ P_3 = P_2(b := 2*a,\ d := d+1,\ e := e-a) \\ P_4 = P_3(a := b-a,\ c := e+d) \end{cases}$$

Since all variables have been initialized at program point $\{1\}$ their values in P_1 are numerical constants. We would give the undefined value Ω to uninitialized variables, and input variables would be initialized by a formal constant. The set union operator $\cup$ represents the effect of paths converging. Finally, if $P = \{<x = \alpha_i,\ y = \beta_i> \mid i \in I\}$ then $P(x := 2*y)$ denotes the evaluation of "$x := 2*y$" for all variable states in P which leads to $\{<x = 2*\beta_i,\ y = \beta_i> \mid i \in I\}$. *End of Example.*

4. EXISTENCE OF A LEAST SOLUTION TO THE SYSTEM OF EQUATIONS

Let L be the set of properties to which belong $P_1, \ldots, P_n$. L is partly ordered by an ordering relation $\leq$ which enables us to compare some properties in L. Moreover, the functions f_i are *order-preserving* (synonymously *monotone* or *isotone*), that is by definition if $P_j \leq P'_j$, $\forall j \in [1,n]$

then $f_i(P_1, \ldots, P_n) \leq f_i(P'_1, \ldots, P'_n)$.

For the system of equations $X = F(X)$, this implies that F is an order-preserving function from the set L^n ordered by $\leq^n$ in itself. Hence known lattice-theoretic theorems can be applied (Tarski[55]) to prove that the equations $X = F(X)$ have always a solution, (or synonymously F has always a *fixpoint*, that is there exits some $P \in L^n$ such that $P = F(P)$).

Example : $P_1, \ldots, P_4$ belong to the set L of sets of variable states. The ordering relation $\leq$ is simply set inclusion $\subseteq$. L is a complete lattice (Birkhoff[73]) which infimum is the empty set $\emptyset$, which supremum is $\{<a = \alpha,\ b = \beta,\ c = \gamma,\ d = \delta,\ e = \varepsilon> \mid \forall(\alpha, \beta, \gamma, \delta, \varepsilon) \in (\mathbb{N} \cup \Omega)^5\}$. The least upper bound operation is set union $\cup$ whereas the greatest lower bound operation is set intersection $\cap$.

The monotony of the functions f_i reflects conservation of information. The larger is the set P_π of possible variable states at some program point π , the larger will be the set $P_{\pi'}$, at point π' immediate successor of π. *End of Example.*

In general the number of fixpoints of F is infinite. Fortunately, it can be shown that there exists a unique *least fixpoint* P of F (such that $P = F(P)$ and if $X = F(X)$ then $P \leq X$). Consequently, the set of desired properties $P_1, \ldots, P_n$ can be uniquely characterized as the least solution of a system of equations $X = F(X)$ associated with the program.

Example : The least solution of the system of equations is :

$$\left[\begin{array}{l}
P_1 = \{<a = 1,\ b = 2,\ c = 3,\ d = 3,\ e = 0>\} \\
P_2 = \{<a = 1,\ b = 2,\ c = 3,\ d = i+3,\ e = -i> \mid i \geq 0\} \\
P_3 = \{<a = 1,\ b = 2,\ c = 3,\ d = i+4,\ e = -i-1> \mid i \geq 0\} \\
P_4 = \{<a = 1,\ b = 2,\ c = 3,\ d = i+4,\ e = -i-1> \mid i \geq 0\}
\end{array}\right.$$

Another possible (but not least solution) is given by :

$$P_2 = \{\langle a=1,\ b=2,\ c=3,\ d=i+3,\ e=-i\rangle \mid i \geq 0\} \cup \{\langle a=0,\ b=0,\ c=i,\ d=i,\ e=0\rangle \mid i \in (\mathbb{N} \cup \{\Omega\})\}$$

However the least solution is preferable since it is the join over all paths solution. It can be interpreted informally as the calculation of the information P_π available at each program point π when this point is reached during execution by following any of the possible paths leading to π. *End of Example.*

5. UNDECIDABILITY OF THE PROBLEM OF COMPUTING THE LEAST SOLUTION

The problem of mechanically computing the least solution $P \in L^n$ of the equations $X = F(X)$ is in general undecidable (Kam & Ullman[77], Reif & Lewis[77]) so that there does not exist an algorithm which for arbitrary L and F will compute the least fixpoint of F. This does not rule out neither finding algorithms for particular F and L nor computing an *approximation* of the exact least solutions of unsolvable systems of equations.

Example : Since the least solution of our system of equations involves infinite sets, it can be thought that their construction might eventually be impossible by a finite process. However, we can try to determine *some* (but not necessarily *all*) constants of a program. Therefore we can approximate :

$$P_2 = \{\langle a=1,\ b=2,\ c=3,\ d=i+3,\ e=-i\rangle \mid i \geq 0\}$$

by a set of variable states such as :

$$\overline{P}_2 = \{\langle a=1,\ b=2,\ c=\gamma,\ d=\delta,\ e=\varepsilon\rangle \mid (\gamma,\delta,\varepsilon) \in (\mathbb{N} \cup \{\Omega\})^3\}$$

More generally any $\overline{P}_2$ such that $P_2 \subseteq \overline{P}_2$ would be a correct approximation of P_2, since $\overline{P}_2$ would include at least (if not at most) the set P_2 of all possible states which can occur during any execution of the program. Now $\overline{P}_2$ can be given a finite representation and computed at compile-time. *End of example.*

6. SYSTEM OF APPROXIMATE EQUATIONS

When confronted with a system of equations $X = F(X)$ in a concrete space $(L^n, \leq^n)$ which least solution P cannot be computed, we can find an asbtract space $(\overline{L}^n, \overline{\leq}^n)$ corresponding to $(L^n, \leq^n)$ and an approximate system of equations $\overline{X} = \overline{F}(\overline{X})$ corresponding to $X = F(X)$ such that the least fixpoint $\overline{P}$ of $\overline{F}$ correctly approximates P, (Cousot[77a]).

The correspondence between L and $\overline{L}$ is established by an *abstraction function* $\mathfrak{a} : L \rightarrow \overline{L}$ and a *concretization function* $\mathfrak{c} : \overline{L} \rightarrow L$ which are order-preserving and such that $\overline{X} = \mathfrak{a}(\mathfrak{c}(\overline{X}))$ and $X \leq \mathfrak{c}(\mathfrak{a}(\overline{X}))$ for any X and $\overline{X}$.

Example : We can approximate a set of variable states such as :

$$P = \{<w = 1,\ x = 0,\ y = 4,\ z = 3>,\ <w = 1,\ x = 0,\ y = 2,\ z = 5>,\ <w = 1,\ x = \Omega,\ y = \Omega,\ z = 7>\}$$

by $<w = \{1\},\ x = \{0,\Omega\},\ y = \{2,4,\Omega\},\ z = \{3,5,7\}>$. The approximation is that we have lost the information concerning the relations among variables (such as "$y+z = 7$ when y is initialized"). Again since we are interested only by the fact that w is an always initialized constant equal to 1, we can make a further abstraction and use the approximation $\overline{P} = \{w \rightarrow 1\}$ which gets rid of the exact values of non-constants (y,z) and ignores those constants (x) which may eventually be uninitialized for some program paths.
$\overline{L}$ is therefore the set of partial functions from the set of variables $V = \{a,b,c,d,e\}$ in $\mathbb{N}$. We represent elements of $\overline{L}$ by their graph, that is a set of couples (variable $\rightarrow$ value). For the sake of completeness, let us add to $\overline{L}$ an infimum denoted $\perp$ corresponding to the empty set $\emptyset$ of L, and a supremum $\top$ corresponding to $\{<a = \alpha, \ldots, e = \varepsilon> \mid (\alpha,\ldots,\varepsilon) \in (\mathbb{N} \cup \{\Omega\})^5\}$ in L.

We have :

$$\mathfrak{a}(\emptyset) = \perp$$
$$\mathfrak{a}(\{<x = 1,\ y = 2>,\ <x = 1,\ y = 3>\}) = \{x \rightarrow 1\}$$
$$\mathfrak{a}(\{<x = 1,\ y = 2>,\ <x = 0,\ y = 3>\}) = \top$$
$$\mathfrak{c}(\perp) = \emptyset$$
$$\mathfrak{c}(\{x \rightarrow 1\}) = \{<x = 1,\ y = \beta> \mid \beta \in (\mathbb{N} \cup \{\Omega\})\}$$
$$\mathfrak{c}(\top) = \{<x = \alpha,\ y = \beta> \mid (\alpha,\beta) \in (\mathbb{N} \cup \{\Omega\})^2\}$$

The ordering $\overline{\leq}$ of $\overline{L}$ is the inclusion $\supseteq$ of function graphs, extended by $\bot \mathrel{\overline{\leq}} \bot \mathrel{\overline{\leq}} \overline{X} \mathrel{\overline{\leq}} \overline{X} \mathrel{\overline{\leq}} \top \mathrel{\overline{\leq}} \top$, $\forall \overline{X} \in \overline{L}$. For example, corresponding to the following inequalities in L :

$$
\begin{array}{l}
\quad\emptyset \\
\subseteq \\
\quad\{<x=1,\ y=2,\ z=0>,\ <x=1,\ y=2,\ z=1>\} \\
\subseteq \\
\quad\{<x=1,\ y=2,\ z=0>,\ <x=1,\ y=2,\ z=1>,\ <x=1,\ y=3,\ z=1>\} \\
\subseteq \\
\quad\{<x=1,\ y=2,\ z=0>,\ <x=1,\ y=2,\ z=1>, \\
\qquad\qquad <x=1,\ y=3,\ z=1>,\ <x=0,\ y=2,\ z=1>\}
\end{array}
$$

we would have in $\overline{L}$:

$$\bot \mathrel{\overline{\leq}} \{x \to 1,\ y \to 2\} \mathrel{\overline{\leq}} \{x \to 1\} \mathrel{\overline{\leq}} \top$$

End of example.

In order for the abstract system of equations $\overline{X} = \overline{F}(\overline{X})$ to correctly simulate the unsolvable concrete system $X = F(X)$, $\overline{F}$ must be chosen so that its least fixpoint $\overline{P}$ is a correct approximation of P the least fixpoint of F. Such conditions for the choice of $\overline{F}$ are given in Cousot[77a].
In order that $P \leq \mathfrak{C}(\overline{P})$ it is sufficient to choose the $\overline{f}_i$ such that for any $(P_1,\ldots,P_n)$ of L^n we have :

$$f_i(P_1,\ldots,P_n) \leq \mathfrak{C}(\overline{f}_i(\mathfrak{A}(P_1),\ldots,\mathfrak{A}(P_n))).$$

Intuitively, instead of computing $f_i(P_1,\ldots,P_n)$ on concrete properties $P_1,\ldots,P_n$, one can as well apply the corresponding function $\overline{f}_i$ on the abstract properties $\mathfrak{A}(P_1),\ldots,\mathfrak{A}(P_n)$, take the concrete form of the result $\mathfrak{C}(\overline{f}_i(\mathfrak{A}(P_1),\ldots,\mathfrak{A}(P_n)))$ which leads to a correct approximation of the exact computations $f_i(P_1,\ldots,P_n)$.

Example : The computation of :

$$
\begin{aligned}
P &= \{<x=1,\ y=\alpha> \mid \alpha > 0\} \cup \{<x=1,\ y=\beta> \mid \beta < 0\} \\
&= \{<x=1,\ y=\alpha> \mid \alpha \in (\mathbb{N} - \{0\})\}
\end{aligned}
$$

is correctly approximated by P' such that $P \subseteq P'$:

$$
\begin{aligned}
P' &= \mathfrak{C}(\mathfrak{A}(\{<x=1,\ y=\alpha> \mid \alpha > 0\}) \cap \mathfrak{A}(\{<x=1,\ y=\beta> \mid \beta < 0\})) \\
&= \mathfrak{C}(\{x \to 1\} \cap \{x \to 1\}) \\
&= \mathfrak{C}(\{x \to 1\}) \\
&= \{<x=1,\ y=\alpha> \mid \alpha \in (\mathbb{N} \cup \{\Omega\})\}
\end{aligned}
$$

The same way, the computation of :

$$P(x \leftarrow 2*x,\ y \leftarrow y+x)$$

where

$$P = \{<x = 1,\ y = \alpha> \mid \alpha \geq 0\}$$

is correctly approximated by :

$$\begin{aligned}
&\mathfrak{C}(\mathfrak{A}(P)(x \leftarrow 2 \circledast x,\ y \leftarrow y \boxplus x))\\
&= \mathfrak{C}(\{x \rightarrow 1\}(x \leftarrow 2 \circledast x,\ y \leftarrow y \boxplus x)\}\\
&= \mathfrak{C}(\{x \rightarrow 2 \circledast 1,\ y \rightarrow \Omega \boxplus x\})\\
&= \mathfrak{C}(\{x \rightarrow 2\})\\
&= \{<x = 2,\ y = \alpha> \mid \alpha \in (\mathbb{N} \cup \{\Omega\})\}\\
&\supseteq \{<x = 2,\ y = \alpha> \mid \alpha > 0\}
\end{aligned}$$

The system of abstract equations will be :

$$\begin{cases}
\overline{P}_1 = \{a \rightarrow 1,\ b \rightarrow 2,\ c \rightarrow 3,\ d \rightarrow 3,\ e \rightarrow 0\}\\
\overline{P}_2 = \overline{P}_1 \cap \overline{P}_4\\
\overline{P}_3 = \overline{P}_2(b := 2 \circledast a,\ d := d \boxplus 1,\ e := e \boxminus a)\\
\overline{P}_4 = \overline{P}_3(a := b \boxminus a,\ c := e \boxplus d)
\end{cases}$$

The operator $\cap$ is the intersection of function graphs extended by $\perp \cap \overline{X} = \overline{X}$ and $\top \cap \overline{X} = \top$, $\forall \overline{X} \in \overline{L}$. The operators $\circledast$, $\boxplus$, $\boxminus$ are extensions of the usual arithmetic operators $*$, $+$, $-$ which result is undefined (Ω) whenever one of their arguments is undefined.

The least solution to the above system of abstract equations is :

$$\left[\begin{array}{l}
\overline{P}_1 = \{a \rightarrow 1,\ b \rightarrow 2,\ c \rightarrow 3,\ d \rightarrow 3,\ e \rightarrow 0\}\\
\overline{P}_2 = \overline{P}_3 = \overline{P}_4 = \{a \rightarrow 1,\ b \rightarrow 2\}
\end{array}\right.$$

$\overline{P}_2$, $\overline{P}_3$, $\overline{P}_4$ correspond to the concrete property :

$$\{<a = 1,\ b = 2,\ c = \gamma,\ d = \delta,\ e = \varepsilon> \mid \forall(\gamma,\delta,\varepsilon) \in (\mathbb{N} \cup \Omega)^3\}$$

which includes the concrete properties P_2, P_3, P_4.

End of example.

7. COMPUTING THE LEAST SOLUTION TO THE SYSTEM OF EQUATIONS

7.1 SUCCESSIVE APPROXIMATIONS

The interest of reasoning on the abstract space of properties $\overline{L}$ and on the system $\overline{X} = \overline{F}(\overline{X})$ is that $\overline{L}$ and $\overline{F}$ can be chosen so that the least fixpoint of $\overline{F}$ can be algorithmicly computed. The known algorithms are of two types : "iterative" algorithms typified for constant propagation by (Kildall[73], Kam & Ullman[76][77]) and "elimination" algorithms typified by (Reif & Lewis[77]). The most general of these two approaches are the "iterative" methods. Intuitively they are akin to Jacobi's method for solving systems of numerical equations by successive approximations.

The least solution P to the system of equations X = F(X) is computed as the limit $\lim_{k\to\infty} F^k(X_0)$ of a sequence X_0, $X_1 = F(X_0)$, $X_2 = F(X_1) = F^2(X_0)$, ..., $X_k = F(X_{k-1}) = F^k(X_0)$,... of successive approximations. The initial approximation X_0 must be chosen such that $X_0 \leq F(X_0)$ and $X_0 \leq P$. Therefore the infimum $\perp^n$ of L^n is always a convenient choice. (Hypothesis on L and F ensuring the existence of the limit, and the proof (related to Kleene [52]'s first recursion theorem) that this limit is the least fixpoint may be found in a.o., Scott[72]).

Example : For solving the equations of paragraph 6 the initial approximation is chosen to be the infimum :

Initialization 1 :

$$\overline{P}_{1,1} = \overline{P}_{2,1} = \overline{P}_{3,1} = \overline{P}_{4,1} = \perp$$

The sequence of approximations is then constructed by successively replacing the current values of $\overline{P}_1$, $\overline{P}_2$, $\overline{P}_3$, $\overline{P}_4$ in the right hand side of the equations, until stabilization.

Step 2 :

$$\overline{P}_{1,2} = \{a \to 1,\ b \to 2,\ c \to 3,\ d \to 3,\ e \to 0\}$$

$$\overline{P}_{2,2} = \overline{P}_{1,2} \cap \overline{P}_{4,1} = \overline{P}_{1,2} \cap \perp = \overline{P}_{1,2}$$

$$= \{a \to 1,\ b \to 2,\ c \to 3,\ d \to 3,\ e \to 0\}$$

$\overline{P}_{3,2} = \overline{P}_{2,2}(b := 2 \circledast a,\ d := d \boxplus 1,\ e := e \boxminus a)$

$= \{a \to 1,\ b \to 2 \circledast 1,\ c \to 3,\ d \to 3 \boxplus 1,\ e \to 0 \boxminus 1\}$

$= \{a \to 1,\ b \to 2,\ c \to 3,\ d \to 4,\ e \to -1\}$

$\overline{P}_{4,2} = \overline{P}_{3,2}(a := b \boxminus a,\ c := e \boxplus d)$

$= \{a \to 1,\ b \to 2,\ c \to 3,\ d \to 4,\ e \to -1\}$

Step 3 :

$\overline{P}_{1,3} = \{a \to 1,\ b \to 2,\ c \to 3,\ d \to 3,\ e \to 0\}$

$\overline{P}_{2,3} = \overline{P}_{1,3} \cap \overline{P}_{4,2}$

$= \{a \to 1,\ b \to 2,\ c \to 3,\ d \to 3,\ e \to 0\} \cap$
$\{a \to 1,\ b \to 2,\ c \to 3,\ d \to 4,\ e \to -1\}$

$= \{a \to 1,\ b \to 2,\ c \to 3\}$

$\overline{P}_{3,3} = \overline{P}_{2,3}(b := 2 \circledast a,\ d := d \boxplus 1,\ e := e \boxminus a)$

$= \{a \to 1,\ b \to 2 \circledast 1,\ c \to 3,\ d \to \Omega \boxplus 1,\ e \to \Omega \boxminus 1\}$

$= \{a \to 1,\ b \to 2,\ c \to 3,\ d \to \Omega,\ e \to \Omega\}$

$= \{a \to 1,\ b \to 2,\ c \to 3\}$

$\overline{P}_{4,3} = \overline{P}_{3,3}(a := b \boxminus a,\ c := e \boxplus d)$

$= \{a \to 2 \boxminus 1,\ b \to 2,\ c \to \Omega \boxplus \Omega\}$

$= \{a \to 1,\ b \to 2\}$

Step 4 :

$\overline{P}_{1,4} = \{a \to 1,\ b \to 2,\ c \to 3,\ d \to 3,\ e \to 0\}$

$\overline{P}_{2,4} = \overline{P}_{1,4} \cap \overline{P}_{4,3}$

$= \{a \to 1,\ b \to 2,\ c \to 3,\ d \to 3,\ e \to 0\} \cap \{a \to 1,\ b \to 2\}$

$= \{a \to 1,\ b \to 2\}$

$\overline{P}_{3,4} = \overline{P}_{2,4}(b := 2 \circledast a,\ d := d \boxplus 1,\ e := e \boxminus a)$

$= \{a \to 1,\ b \to 2\}$

$\overline{P}_{4,4} = \overline{P}_{3,4}(a := b \boxminus a,\ c := e \boxplus d)$

$= \{a \to 1,\ b \to 2\}$

A last step 5, would prove stabilization, that is $\overline{P}_{1,5} = \overline{P}_{1,4}$, $\overline{P}_{2,5} = \overline{P}_{2,4}$, $\overline{P}_{3,5} = \overline{P}_{3,4}$, $\overline{P}_{4,5} = \overline{P}_{4,4}$.

The final result is that "a" and 'b" have been found to be constants, whereas "c" has not been discovered. Recall this is a consequence of our choice to use the approximate equations. This choice was motivated by our feeling that we could not solve the system of exact equations given at paragraph 3. For example, solving iteratively by successive approximations, we would try to built the infinite sets of the least solution given at paragraph 4, by successively adding an element to a set initially empty. This process would converge only after infinitely many steps. *End of example.*

7.2 CONVERGENCE OF THE ITERATES

Notice that the initial approximation X_0 in the sequence X_0, X_1, ..., X_k, ... satisfies $X_0 \leq F(X_0) = X_1$. Since F is monotone this implies that $F(X_0) \leq F(X_1)$ that is $X_1 \leq X_2$. By recurrence on k, we have in general $X_k \leq X_{k+1}$ and by transitivity $X_0 \leq X_1 \leq \ldots \leq X_k \leq \ldots$ so that the sequence of successive approximations is an increasing chain. The iteration process eventually converges after m steps if $X_m = X_{m-1}$. On the contrary it diverges when the sequence of successive approximations is an infinite strictly increasing chain. Therefore the most widely used hypothesis to insure convergence of iterative methods is that L must satisfy the ascending chain condition (every strictly increasing chain in L is finite).

Example : In our example, any strictly increasing chain in $\overline{L}$ is of the form of the following one :

$$\bot < \{a \to \alpha,\ b \to \beta, \ldots, e \to \varepsilon\} < \ldots < \{a \to \alpha,\ b \to \beta\} < \{a \to \alpha\} < \top$$

For a program with m variables, the maximal length of a strictly increasing chain in $\overline{L}$ is m+2. Let n be the number of equations, the maximal length of strictly increasing chains in $\overline{L}^n$ is $n * (m+2)$. Hence the trivial constants of the program are found in at most $n * (m+2) + 1$ steps. (This worst case analysis is given to prove convergence, but is largely bigger that the average case. For example we converged in 5 steps whereas the maximum is $4 * (5+2) + 1 = 29$). *End of example.*

When the system of equations $X = F(X)$, in L^n cannot be solved iteratively, one can approximate its least solution. We illustrated "*structural approximation*" which consists in simulating the iteratively unsolvable system of equations in a space satisfying the ascending chain condition. Alternatively, "*computational approximation*" can be used either to truncate the infinite sequence of successive approximations which leads to a lower approximation of the limit or to compute an upper approximation of the limit in a finite number of steps, (Cousot[76], Cousot[77a]).

7.3 ACCELERATING THE CONVERGENCE OF THE ITERATES

We defined the sequence $X_0, X_1, \ldots, X_k, \ldots$ of successive approximations by $X^{k+1} = F(X^k)$ $(k = 0,1,2,\ldots)$ which can be detailed as :

$$\left\{\begin{array}{l} X_i^{k+1} = f_i(X_1^k, X_2^k, \ldots, X_n^k) \\ \\ i = 1,2,\ldots,n \end{array}\right. \qquad (k = 0,1,2,\ldots)$$

However (under our hypothesis on F and L) any chaotic iteration method would converge to the least solution, this signifies that one can arbitrarily determine at each step which are the components of the system of equations which will evolve and in what order (as long as no component is forgotten indefinitely).

Example : When solving the equations we used the Gauss-Seidel iteration method :

$$\left\{\begin{array}{l} X_1^{k+1} = f_1(X_1^k, X_2^k, \ldots\ldots\ldots\ldots\ldots, X_n^k) \\ \ldots \\ X_i^{k+1} = f_i(X_1^{k+1}, \ldots, X_{i-1}^{k+1}, X_i^k, \ldots, X_n^k) \\ \ldots \\ X_n^{k+1} = f_n(X_1^{k+1}, \ldots\ldots\ldots\ldots, X_{n-1}^{k+1}, X_n^k) \end{array}\right.$$

which consists in continually reinjecting in the computations the results of the computations themselves. This reduces the memory congestion and accelerates the convergence. *End of example.*

Among the possible iterating order which can be used to solve the equations, some converge more rapidly than others. The question of optimal order of iteration has not yet received a conceptual answer. (Such an order has been shown to exist for a particular class of equations (Kennedy[75], Tarjan[76]) and can sometimes be algorithmicly constructed (Aho & Ullman[75])).

Note : When the iterating order which is used to solve the equations corresponds to the program control graph the successive approximations can be intuitively understood as a symbolic execution of the program. In this symbolic execution local abstract properties are used in place of the actual execution environment and operations of the language are interpreted as specified by the equations which define the transformation of a property when passing through an elementary instruction. Each step in this symbolic execution process corresponds to the evaluation of an equation. Yet, all possible paths are followed pseudo-parallely and eventually merged together at junction points. This was the way iterative methods were first understood (e.g. Kildall[73], Schwartz[75], Sintzoff[72], Urschler [74], Wegbreit[75b]). *End of note.*

8. CONCLUSION AND HISTORICAL SURVEY

We informally exposed a mathematical theory of global analysis of programs using a simple example concerning the compile time determination of constant computations. The model is typical of the "fixpoint approach" to analysis of programs.

By "fixpoint approach" we refer to the whole of techniques for determining properties of programs which take as starting point the fact that these properties can be defined as the least fixpoint of a system of equations which is associated in a rather natural way with the program. This approach has been recently recognized (Cousot[77a], Cousot[77c]) to provide a unified understanding of apparently unrelated works such as global data flow analysis, type checking (Cousot[77b], Jones & Muchnick[76], Sintzoff[72], Tenenbaum[74]), denotational semantics of programming languages (references in Scott[76]), program proving (e.g. Manna et al.[73]), determination of weak properties of programs (Cousot[76], Karr[75], Karr[76], Sintzoff[72], Wegbreit[75b]), evaluation of program performance (e.g. Kennedy & Zucconi[77], Wegbreit[75a]), etc.

In the domain of global data flow analysis the use of lattices, systems of equations and fixpoint computations remained for a long time implicit, in particular reasonings about the system of equations where replaced by tracing the program flow graph. However all classical algorithms can be understood in light of the fixpoint approach.

The early methods used to solve the equations of data analysis problems where akin to Gaussian elimination, (Allen[70], Allen[71], Allen & Cocke[72], Cocke[70]). The technique is limited to a restricted class of recursive equations (corresponding to "reducible" programs (Hecht & Ullman[72], Hecht & Ullman[74], Kasvanov[73], Tarjan[74]) which are a frequent but not general case) and to a restricted class of data flow problems. This so called "interval analysis" approach was further extended to deal with wider classes of program graphs and data flow problems (e.g. Graham & Wegman[76], Kennedy[71]). However, in general direct methods for solving the equations (Fong & al.[75], Fong[77], Kennedy & Zucconi[77], Reif & Lewis[77]) are application dependent and cannot be easily generalized to arbitrary data flow analysis problems.

More recently iterative methods akin to Jacobi's successive approximations appeared in the literature (Backhouse[76], Cousot[76], Hecht & Ullman[75], Kam & Ullman[76], Kam & Ullman[77], Kildall[73], Morel & Renvoise[74], Schwartz[75], Ullman[73], Urschler[74], Wegbreit[75b]). They are more general than direct methods since even when convergence is not naturally guaranteed it can be enforced by using "computational" approximation techniques (Cousot[77a]). It is often argued that iterative approaches are more expensive than direct methods. On the contrary the comparisons for given problems are inconclusive (Hecht & Ullman[75], Kennedy[76]) because the hypothesis which are necessary to allow the use of direct methods also imply that the number of iterates will be small (Kam & Ullman[76], Graham & Wegman[76], Tarjan[75], Ullman[75]).

We hope to have clearly shown that the central problem in global program analysis is to solve a system of equations in a space appropriately chosen for modelling the properties to be gathered about the program. Mathematicians have spend centuries in studying the resolution of systems of real equations, very efficient methods have been discovered. Only very few work has been done on systems of equations defined on discrete domains. Hence considerable progress could be made in the near future.

9. BIBLIOGRAPHY

Aho & Ullman[73]
A.V. Aho, and J.D. Ullman, The theory of parsing, translation and compiling, vol. II : compiling, Prentice-Hall, Englewood Cliffs, N.J., 1973.

Aho & Ullman[75]
A.V. Aho, and J.D. Ullman, Node listings for reducible flow graphs, Proc. 7th Annual ACM Symp. on Theory of Computing, May'1975, 177-185.

Allen[70]
F.E. Allen, Control flow analysis, SIGPLAN Notices, vol. 5, 1970, 1-9.

Allen[71]
F.E. Allen, A basis for program optimization, Proc. IFIP Cong. 71, vol. 1, North-Holland Pub. Co., Amsterdam, 1971, 381-390.

Allen & Cocke[72]
F.E. Allen, and J. Cocke, Graph theoretic constructs for program control flow analysis, IBM. Res. Rep. RC3933, T.J. Watson Res. Center, Yorktown Heights, N.J., July 1972.

Backhouse[76]
R.C. Backhouse, An improved iterative algorithm for global data flow analysis, Tech. Rep. no. 3, Dept. of computer Sci., Heriot-Watt University, Edinburgh, May 1976.

Birkhoff[73]
G. Birkhoff, Lattice theory, AMS Coll. Pub., Vol. XXV, 3rd ed., Providence, R.I., 1973.

Branquart et al.[76]
P. Branquart, J.P. Cardinal, J. Lewi, J.P. Delescaille, M. Vanbegin, An optimized translation process and its application to ALGOL 68, Springer-Verlag, 1976.

Cocke[70]
J. Cocke, Global common subexpression elimination, SIGPLAN Notices, vol. 5, no. 7, July 1970, 20-24.

Cocke & Schwartz[69]
J. Cocke, and J.T. Schwartz, Programming Languages and their compilers, New York University, N.Y., 1969.

Cousot[76]
P.M. Cousot, and R. Cousot, Static determination of dynamic properties of programs, Proc. of the 2nd Int. Symp. on Programming, B. Robinet (Ed.), Dunod, Paris, April 1976. [Also in MOL-Bulletin, no. 5, P.M. Cousot (Ed.), IRIA, Rocquencourt, France, Sept. 1976, 27-52].

Cousot[77a]
P.M. Cousot, and R. Cousot, Abstract interpretation : a unified lattice model for static analysis of programs by construction or approximation of fixpoints, Conf. Rec. of the 4th ACM Symp. on Principles of Programming Languages, Los Angeles, Calif., Jan. 1977, 238-252.

Cousot[77b]
P.M. Cousot, and R. Cousot, Static determination of dynamic properties of generalized type unions, Proc. of the ACM Conf. on Language Design for Reliable Software, Raleigh, North-Carolina, March 1977.

Cousot[77c]
P.M. Cousot, and R. Cousot, Towards a universal model for static analysis of programs, Res. Rep. Submitted for Publication, March 1977.

Fong[77]
A. Fong, Generalized common subexpressions in very high level languages, Conf. Rec. of the 4th ACM Symp. on Principles of Programming Languages, Los Angeles, Jan. 1977, 48-57.

Fong et al.[75]
A. Fong, J. Kam, and J.D. Ullman, Application of lattice algebra to loop optimization, Conf. Rec. of the 2nd ACM Symp. on Principles of Programming Languages, Palo Alto, Calif., Janv. 1975, 1-9.

Graham & Wegman[76]
S.L. Graham, and M. Wegman, A fast and usually linear algorithm for global flow analysis, Journal of the ACM, vol. 23, no. 1, Jan. 1976, 172-202.

Hecht[75]
M.S. Hecht, A theoretical foundation for global program improvement, American Elsevier, 1975.

Hecht & Ullman[72]
M.S. Hecht, and J.D. Ullman, Flow graph reducibility, SIAM J. Comput., vol. 1, no. 2, June 1972, 188-202.

Hecht & Ullman[74]
M.S. Hecht, and J.D. Ullman, Characterizations of reducible flow graphs, Journal of the ACM, vol. 21, no. 3, July 1974, 367-375.

Hecht & Ullman[75]
M.S. Hecht, and J.D. Ullman, A simple algorithm for global flow analysis problems, SIAM J. Computing, vol. 4, 1975, 519-532.

Jones & Muchnick[76]
N.D. Jones, and S.S. Muchnick, Binding time optimization in programming languages : some thoughts toward the design of an ideal language, Conf. Rec. of the 3rd ACM Symp. on Principles of Programming Languages, Atlanta, Jan. 1976, 77-94.

Kam & Ullman[76]
J.B. Kam, and J.D. Ullman, Global data flow analysis and iterative algorithms, Journal of the ACM, vol. 23, no. 1, Jan. 1976, 158-171.

Kam & Ullman[77]
J.B. Kam, and J.D. Ullman, Monotone data flow analysis frameworks, Acta Informatica, vol. 7, 1977, 305-317.

Karr[75]
M. Karr, Gathering information about programs, Mass. Computer Associates, Inc., CAID-7501-0611, July 1975.

Karr[76]
M. Karr, Affine relationship among variables of a program, Acta Informatica, vol. 6, 1976, 133-151.

Kasvanov[73]
V.N. Kasvanov, Some properties of fully reducible graphs, Inf. Proc. Letters, vol. 2, no. 4, 1973, 113-117.

Kennedy[71]
K.W. Kennedy, A global flow analysis algorithm, Int. J. of Computer Math., vol. 3, Dec. 1971, 5-15.

Kennedy[75]
K.W. Kennedy, Node listings applied to data flow analysis, Conf. Rec. of the 2nd ACM Symp. on Principles of Programming Languages, Palo Alto, Calif., Jan. 1975, 10-21.

Kennedy[76]
K.W. Kennedy, A comparison of two algorithms for global data flow analysis, SIAM J. Computing, Vol. 1, March 1976, 158-180.

Kennedy & Zucconi[77]
K.W. Kennedy, and L. Zucconi, Applications of a graph grammar for program control flow analysis, Conf. Rec. of the 4th ACM Symp. on Principles of Programming Languages, Los Angeles, Calif., Jan. 1977, 72-85.

Kildall[73]
G.A. Kildall, A unified approach to global program optimization. Conf. Rec. of the ACM Symp. on Principles of Programming Languages, Boston, Mass., Oct. 1973, 194-206.

Kleene[52]
S.C. Kleene, Introduction to metamathematics, North-Holland Pub. Co., Amsterdam, 1952.

Manna et al.[73]
Z. Manna, S. Ness, and J. Vuillemin, Inductive methods for proving properties of programs, Communications of the ACM, vol. 16, 491-502.

Morel & Renvoise[74]
E. Morel, and C. Renvoise, Etude et réalisation d'un optimizeur global, Th. de 3ième cycle, U. of Paris VI, June 1974.

Reif & Lewis[77]
J.H. Reif, and H.R. Lewis, Symbolic evaluation and the global value graph, Conf. Rec. of the 4th ACM Symp. on Principles of Programming Languages, Los Angeles, Calif., Jan. 1977, 104-118.

Rosen[77]
B.K. Rosen, Applications of high level control flow, Conf. Rec. of the 4th ACM Symp. on Programming Languages, Los Angeles, Calif., Jan. 1977, 38-47.

Schaefer[73]
M. Schaefer, A mathematical theory of global program optimization, Prentice-Hall, Englewood Cliffs, N.J., 1973.

Schwartz[75]
J.T. Schwartz, Automatic data structure choice in a language of very high level, Communications of the ACM, vol. 18, no. 12, Dec. 1975, 722-728.

Scott[72]
D. Scott, Continuous lattices, Proc. 1971 Dalhousie Conf., Lecture notes in Math., vol. 274, Springer-Verlag, New-York, 1972, 97-136.

Scott[76]
D. Scott, Data types as lattices, SIAM J. Computing, vol. 5, no. 3, Sept. 1976, 522-587.

Sintzoff[72]
M. Sintzoff, Calculating properties by valuations on specific models, Proc. ACM Conf. on Proving Assertions about Programs, SIGPLAN Notices, vol. 7, no. 1, 1972, 203-207.

Tarjan[74]
R. Tarjan, Testing flow graph reducibility, J. Comp. Sys. Sciences, vol. 9, 1974, 355-365.

Tarjan[75]
R.E. Tarjan, Solving path problems on directed graphs, STAN-CS-75-528, Computer Sci. Dept., Stanford U., 1975.

Tarjan[76]
R.E. Tarjan, Iterative algorithms for global flow analysis, Tech. Report CS 76-545, Computer Science Dept., Stanford U., Feb. 1976.

Tarski[55]
A. Tarski, A lattice-theoretical fixpoint theorem and its applications, Pacific J. Math., vol. 5, 1955, 285-309.

Tenenbaum[74]
A.M. Tenenbaum, Type determination for very high level languages, NSO-3, Courant Inst. of Math. Sci., New York U., Oct. 1974.

Ullman[73]
J.D. Ullman, Fast algorithms for the elimination of common subexpressions. Acta Informatica, vol. 2, no. 3, 1973, 191-213.

Ullman[75]
J.D. Ullman, Data flow analysis, Second USA-Japon Computer Conference, Montvale, (N.J.) : AFIPS Press, 1975.

Urschler[74]
G. Urschler, Complete redundant expression elimination in flow diagrams, IBM Research Report RC 4965, T.J. Watson Research Center, Yorktown Heights, N.Y., Aug. 1974.

Wegbreit[75a]
B. Wegbreit, Mechanical program analysis, Communications of the ACM, vol. 18, no. 9, Sept. 1975, 528-539.

Wegbreit[75b]
B. Wegbreit, Property extraction in well-founded property sets, IEEE Trans. on Soft. Eng., vol. SE-1, no. 3, Sept. 1975, 270-285.

Wulf et al.[75]
W.A. Wulf, R.K. Johnson, C.C. Weinstock, S.O. Hobbs, and C.M. Geschke, The design of an optimizing compiler, American Elsevier, New York, 1975.