AUTOMATIC SYNTHESIS OF OPTIMAL INVARIANT ASSERTIONS : MATHEMATICAL FOUNDATIONS

Patrick Cousot* and Radhia Cousot**

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

1. INTRODUCTION

The problem of discovering invariant assertions of programs is explored in light of the fixpoint approach in the static analysis of programs, Cousot [1977a], Cousot[1977b].

In section 2 we establish the lattice theoric foundations upon which the synthesis of invariant assertions is based. We study the resolution of a fixpoint system of equations by Jacobi's successive approximations method. Under continuity hypothesis we show that any chaotic iterative method converges to the optimal solution. In section 3 we study the deductive semantics of programs. We show that a system of logical forward equations can be associated with a program using the predicate transformer rules which define the semantics of elementary instructions. The resolution of this system of semantic equations by chaotic iterations leads to the optimal invariants which exactly define the semantics of this program. Therefore these optimal invariants can be used for total correctness proofs (section 4). Next we show that usually a system of inequations is used as a substitute for the system of equations. Hence the solutions to this system of inequations are approximate invariants which can only be used for proofs of partial correctness (section 5). In section 6 we show that symbolic execution of programs consists in fact in solving the semantic equations associated with this program. The construction of the symbolic execution tree corresponds to the chaotic successive approximations method. Therefore symbolic execution permits optimal invariant assertions to be discovered provided that one can pass to the limit, that is consider infinite paths in the symbolic execution tree. Induction principles can be used for that purpose. In section 7 we show how difference equations can be utilized to discover the general term of the sequence of successive approximations so that optimal invariants are obtained by a mere passage to the limit. In section 8 we show that an approximation of the optimal solution to a fixpoint system of equations can be obtained by strengthening the term of a chaotic iteration sequence. This formalizes the synthesis of approximate invariants by heuristic methods. Various examples provide a helpful intuitive support to the technical sections.

2. RESOLUTION OF A FIXPOINT SYSTEM OF EQUATIONS BY CHAOTIC ITERATIONS

We denote by $(L,\sqsubseteq,\bot,\top,\sqcup,\sqcap,\bigsqcup,\bigsqcap)$ a *complete lattice* with respect to the *partial ordering* $\sqsubseteq$. We use the symbols $\sqcup,\sqcap,\bigsqcup,\bigsqcap$ for the finite and infinite lattice operations of *join* and *meet*. The *infimum* $\bot$ and *supremum* $\top$ of the lattice are defined by $\bot=\bigsqcap L$ and $\top=\bigsqcup L$, (Birkhoff[1967]).

A function $\varphi \in L \to L$ of L into L is *isotone* (synonymously, *order-preserving* or *monotone*) if and only if $\{\forall x,y \in L,\ (x \sqsubseteq y) \Rightarrow (\varphi(x) \sqsubseteq \varphi(y))\}$.

We define the *limit* of a *chain* $x^0 \sqsubseteq x^1 \sqsubseteq \ldots \sqsubseteq x^k \sqsubseteq \ldots$ to be its least upper bound, $\lim_{k\to\infty} x^k = \bigsqcup_{k=0}^{\infty} x^k$.

A function $\varphi \in L \to L$ is *continuous* if and only if for any chain x^k, k=0,1,... we have $\lim_{k\to\infty} \varphi(x^k) = \varphi(\lim_{k\to\infty} x^k)$. Note that a continuous function is necessarily isotone.

---

* Attaché de Recherche au CNRS, Laboratoire Associé n°7.

** This work was supported by IRIA-SESORI under grant 76-160.

We denote by L^n the set of all vectors $X=(X_1,\ldots,X_n)$ the components of which belong to L. $(L^n,\sqsubseteq,\downarrow,\uparrow,\sqcup,\sqcap,\bigsqcup,\bigsqcap)$ is a complete lattice with the usual "componentwise" definitions : $\{\{X\sqsubseteq Y\} \Leftrightarrow \{\forall i\in[1,n],\ X_i\sqsubseteq Y_i\}\}$, $\{\downarrow=(\bot,\ldots,\bot)\}$, $\{X\sqcup Y = ((X_1\sqcup Y_1),\ldots,(X_n\sqcup Y_n))\}$, etc.

A function of several variable $f\in L^n\to L$ is said to be isotone (continuous) in the variables jointly if and only if it is isotone (continuous) in the variables separately.

Hereafter we will consider a system of continuous equations with n variables of the form :

$$\begin{cases} X_1 = F_1(X_1,\ldots,X_n) \\ \ldots \\ X_n = F_n(X_1,\ldots,X_n) \end{cases}$$

This system can be abbreviated by a *fixpoint equation* $X=F(X)$ where X is the vector $(X_1,\ldots,X_n)$ and F a continuous function of type $L^n\to L^n$.

An element X of L^n is a *fixpoint* of $F\in L^n\to L^n$ if and only if $F(X)=X$. The *least fixpoint* $lfp(F)$ of F is such that :
$\{\{F(lfp(F))=lfp(F)\}$ <u>and</u> $\{\forall X\in L^n, \{F(X)=X\} \Rightarrow \{lfp(F)\sqsubseteq X\}\}\}$.

THEOREM (Tarski[1955]) Any monotone map F of a complete lattice L^n into itself has a least fixpoint defined by : $lfp(F) = \bigsqcap\{X\in L^n : F(X)\sqsubseteq X\}$.

In practice this theorem is not constructive since in general the set $\{X\in L^n : F(X)\sqsubseteq X\}$ of *post-fixpoints* of F is infinite and cannot be easily characterized. Yet, when F is continuous Kleene[1952] and Tarski[1955] suggest that the least fixpoint of F can be obtained as the limit of a *sequence of successive approximations* $X^0=\downarrow$, $X^1=F(X^0),\ldots,$ $X^k=F(X^{k-1}),\ldots$ that is $lfp(F)=\lim_{k\to\infty} F^k(\downarrow)$ where F^k denotes the k-fold composition of F with itself. This is nothing else than Jacobi's method of successive approximations :

$$\begin{cases} X_i^k = F_i(X_1^{k-1},X_2^{k-1},\ldots,X_n^{k-1}) \quad (k=1,2,\ldots) \\ i \;= 1..n \end{cases}$$

We now generalize this result by showing that any chaotic iteration method converges to the least fixpoint of F. Otherwise stated 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).

Let J be a non-empty subset of $\{1,\ldots,n\}$. We denote by F_J the map $L^n\to L^n$ defined by $F_J(X_1,\ldots,X_n) = (Y_1,\ldots,Y_n)$ where $\forall i\in[1,n]$ we have $Y_i=$ <u>if</u> $i\in J$ <u>then</u> $F_i(X_1,\ldots,X_n)$ <u>else</u> X_i <u>fi</u>.

DEFINITION 2.1 A *chaotic iteration* corresponding to the operator F and starting with a given vector X^0 such that $X^0\sqsubseteq F(X^0)\sqsubseteq lfp(F)$ is a sequence X^k, $k=0,1,\ldots$ of vectors of L^n defined recursively by $X^k=F_{J^{k-1}}(X^{k-1})$ where J^k, $k=0,1,\ldots$ is a sequence of subsets of $\{1,\ldots,n\}$ such that $\{\exists m\geq 0 : \{\forall i\in[1,n], \forall k\geq 0, \exists \ell\in[0,m[: i\in J^{k+\ell}\}\}$.

THEOREM 2.2 The limit X^∞ of a chaotic iteration $X^0,\ldots,X^k,X^{k+1},\ldots$ is equal to $lfp(F)$.

Lemma 2.2.1 $\{\forall k\geq 0,\ X^k \sqsubseteq X^{k+1} \sqsubseteq F(X^k) \sqsubseteq lfp(F)\}$.

Proof : Let us first remark that whenever $X\sqsubseteq F(X)\sqsubseteq lfp(F)$ we have $\forall J\in\{1,\ldots,n\}$, $X\sqsubseteq F_J(X)\sqsubseteq F(X)\sqsubseteq lfp(F)$. Indeed $\forall i\in[1,n]$, $X_i\sqsubseteq F_i(X)$ therefore if $i\in J$ then $X_i\sqsubseteq F_i(X)=F_J(X)_i$ otherwise $X_i=F_J(X)_i\sqsubseteq F_i(X)$.

Since by hypothesis $X^0\sqsubseteq F(X^0)\sqsubseteq lfp(F)$ this implies $X^0\sqsubseteq F_{J^0}(X^0)=X^1\sqsubseteq F(X^0)\sqsubseteq lfp(F)$. For the induction step let us assume that $X^{k-1}\sqsubseteq X^k\sqsubseteq F(X^{k-1})\sqsubseteq lfp(F)$ for some k>0. If $i\in J^{k-1}$ then $X_i^k=F_i(X^{k-1})\sqsubseteq F_i(X^k)\sqsubseteq lfp(F)_i$ since $X^{k-1}\sqsubseteq X^k\sqsubseteq lfp(F)$ and F_i is isotone Otherwise $i\notin J^{k-1}$ and then $X_i^k=X_i^{k-1}\sqsubseteq F_i(X^{k-1})$ by induction hypothesis and $F_i(X^{k-1})\sqsubseteq F_i(X^k)\sqsubseteq lfp(F)_i$ by isotony. In both cases $\forall i\in[1,n]$, $X_i^k\sqsubseteq F_i(X^k)\sqsubseteq lfp(F)_i$ therefore $X^k\sqsubseteq F(X^k)\sqsubseteq lfp(F)$ proving that $X^k\sqsubseteq X^{k+1}= F_{J^k}(X^k)\sqsubseteq F(X^k)\sqsubseteq lfp(F)$. *End of Proof.*

Lemma 2.2.2 $\{\exists q\in[0,m] : \forall k\geq 0,\ F(X^k) \sqsubseteq X^{k+q}\}$.

Proof : The proof is by reductio ad absurdum. Let us suppose that $\{\forall q\in[0,m], \exists k\geq 0 :$ <u>not</u>$(F(X^k) \sqsubseteq X^{k+q})\}$ $\Leftrightarrow \{\forall q\in[0,m], \exists k\geq 0 : (X^{k+q}\sqsubset F(X^k))$ <u>or</u> $(F(X^k)$ not comparable with $X^{k+q})\}$.

case 1 : Suppose that $\forall q\in[0,m]$, $\exists k\geq 0$ such that $F(X^k)$ is not comparable with X^{k+q}. This must be true for q=0 which contradicts lemma 2.2.1.

case 2 : Suppose now that $\forall q\in[0,m]$, $\exists k\geq 0$ such that $X^{k+q}\sqsubset F(X^k)$, that is to say by definition of the strict inequality $\sqsubset$ we have $X^{k+q}\sqsubseteq F(X^k)$ and $X^{k+q}\neq F(X^k)$. This implies that for some component $i\in[1,n]$ we have $X_i^{k+q}\sqsubset F_i(X^k)$, while for the other components the inequality is not necessarily strict. By definition of chaotic iterations $\{\exists m\geq 0 : \{\forall i, \forall k, \exists \ell\in[0,m[: i\in J^{k+\ell}\}\}$, therefore $X_i^{k+\ell+1}=F_i(X^{k+\ell})$. But lemma 2.2.1 implies by transitivity that $X^k \sqsubseteq X^{k+\ell}$ thus by isotony $F_i(X^k) \sqsubseteq F_i(X^{k+\ell})$ which implies $F_i(X^k) \sqsubseteq X_i^{k+\ell+1}$. Choosing $q=\ell+1$ we have $\exists k$ such that $F_i(X^k) \sqsubseteq X_i^{k+q}$ and also by hypothesis such that $X_i^{k+q} \sqsubset F_i(X^k)$, which is impossible. This contradiction proves the truth of lemma 2.2.2. *End of Proof.*

Proof of theorem 2.2 :
Part 1 : Let us prove that $X^\infty \sqsubseteq F(X^\infty)$. According to lemma 2.2.1 we have $\forall k\geq 0$, $X^k \sqsubseteq F(X^k)$, hence passing to the limit we get $\lim_{k\to\infty} X^k \sqsubseteq \lim_{k\to\infty} F(X^k)$. The sequence of chaotic iterations is an increasing chain (lemma 2.2.1) and F is continuous, hence $\lim_{k\to\infty} F(X^k) = F(\lim_{k\to\infty} X^k)$ hence by transivity $\lim_{k\to\infty} X^k = X^\infty \sqsubseteq F(X^\infty)$.
Part 2 : Let us now prove that $F(X^\infty) \sqsubseteq X^\infty$. By definition X^∞ is $\lim_{k\to\infty} X^k = \lim_{k\to\infty} X^{k+q}$ since X^k, $k=0,1,\ldots$ is a chain. But according to lemma 2.2.2 $\forall k\geq 0$, $F(X^k) \sqsubseteq X^{k+q}$ hence $\lim_{k\to\infty} F(X^k) \sqsubseteq \lim_{k\to\infty} X^{k+q}$ by our definition of limits as least upper bounds. F being continuous $\lim_{k\to\infty} F(X^k) = F(\lim_{k\to\infty} X^k)$ proving that $F(X^\infty) \sqsubseteq X^\infty$.
Conclusion : By antisymmetry we conclude $X^\infty = F(X^\infty)$ and X^∞ is a fixpoint of F. Also lemma 2.2.1 implies that $X^\infty = \lim_{k\to\infty} X^k \sqsubseteq lfp(F)$. Since the least fixpoint of F is unique we conclude $X^\infty = lfp(F)$. *End of Proof.*

Our definition of limits as least upper bounds imposes to take $lfp(F)$ to be the join $\bigsqcup_k X^k$ of all terms of the chaotic iteration sequence. In practice we can overcome this difficulty thanks to the following theorem :

THEOREM 2.3 Let m be the maximum number of steps which are necessary for any component to evolve in chaotic iterations (2.1). There exists an ordinal k of cardinality less than or equal to that of L^n such that $\forall \ell\geq km$ implies $lfp(F)=X^\ell$.

lemma 2.3.1 Let $k,\ell\geq 0$ such that $k\neq\ell$ then $\{(lfp(F)\neq X^{km})$ <u>or</u> $(lfp(F)\neq X^{\ell m})\} \Rightarrow \{X^{km}\neq X^{\ell m}\}$.

Proof : We prove that $X^{km}=X^{\ell m}$ implies $lfp(F)=X^{km}$ for $k<\ell$ (since the case $k>\ell$ is symmetric). According to lemma 2.2.1 and 2.2.2 we have $X^{km} \sqsubseteq F(X^{km}) \sqsubseteq X^{km+m} \sqsubseteq X^{\ell m} = X^{km}$ proving that $X^{km}=F(X^{km})$ and since $X^{km} \sqsubseteq lfp(F)$ we have $X^{km}=lfp(F)$. *End of Proof.*

Proof of theorem 2.3 : The proof that $\{\exists k : (\overline{k}\leq\overline{\overline{L}}^n)$ <u>and</u> $(\ell\geq km \Rightarrow lfp(F)=X^\ell)\}$ is by reductio ad absurdum. Indeed, suppose that $\{\forall k : (\overline{k}>\overline{\overline{L}}^n)$ <u>or</u> $(\ell\geq km$ <u>and</u> $lfp(F)\neq X^\ell)\}$. Let α be the least ordinal of cardinality strictly greater than $\overline{\overline{L}}^n$. $\forall k<\alpha$ we must have $(\ell\geq km$ <u>and</u> $lfp(F)\neq X^\ell)$, that is $lfp(F)\neq X^{km}$ when choosing ℓ to be km. Let us define $\psi \in \alpha\to L^n$ by $\psi(k)=X^{km}$. $\forall k_1,k_2\in\alpha$ such that $k_1\neq k_2$ (with eventually $(k_1=\alpha)$ exclusive or $(k_2=\alpha)$) we have $(lfp(F)\neq X^{k_1 m})$ or $(lfp(F)\neq X^{k_2 m})$ hence lemma 2.3.1 imply that $X^{k_1 m}\neq X^{k_2 m}$ proving that ψ is a one to one correspondence of α into L^n. Therefore α is of cardinality less or equal to that of L^n which is the desired contradiction. *End of Proof.*

3. DEDUCTIVE SEMANTICS OF PROGRAMS

The deductive semantics of a program defines the logical invariant assertions associated with each program point. Once determined these assertions can be used to verify the program with respect to a specification or to prove that the program is errored.

3.1. LOGICAL ASSERTIONS

We consider the set L of logical first order predicates $P(X,\overline{X})$ over the set X of program variables and the set $\overline{X}$ of initial values of these program variables. X and $\overline{X}$ are the free variables in the predicate P. The assertion $P_i(X,\overline{X})$ associated with a point i of the program describes the values of the program variables at program point i during an execution starting with an initial state $\overline{X}$ of the program variables. The set of predicates $P(X,\overline{X})$ forms a complete lattice $(\sqsubseteq,\bot,\top,\sqcup,\sqcap,\bigsqcup,\bigsqcap)$ by choosing respectively ($\Rightarrow$,<u>false</u>,<u>true</u>,<u>or</u>,<u>and</u>,<u>OR</u>,<u>AND</u>).

3.2. SYSTEM OF LOGICAL FORWARD EQUATIONS

We use the notation $\{P(X,\overline{X})\}$ S $\{Q(X,\overline{X})\}$ to mean that for every $X,\overline{X}$, if $P(X,\overline{X})$ holds prior to execution of the statement S then $Q(X,\overline{X})$ is the strongest post condition such that the statement S faultless executes and properly terminates leaving the program variables in a final state satisfying Q (Dijkstra [1976]). The deductive semantics of a programming language defines the rules which can be used to associate a system of equations with a sequential program.

Program entry point j :

$$P_j(X,\bar{X}^j) = \{(X_i=\bar{X}_i^j),\ i=1..m\}$$

The respective initial values of the variables $X=(X_1,\dots,X_m)$ are the symbols $\bar{X}^j=(\bar{X}_1^j,\dots,\bar{X}_m^j)$ (We may eventually have $(X_i=\Omega)$ when the variable X_i is not initialized).

Assignment statements :

$$\{P(X,\bar{X})\}\ X_i:=E(X)\ \{\exists v: P^v_{X_i}\ \underline{\text{and}}\ X_i=E^v_{X_i}\}$$

where $P^v_{X_i} = P(X_1,\dots,X_{i-1},v,X_{i+1},\dots,X_m,\bar{X}_1,\dots,\bar{X}_m)$

$E^v_{X_i} = E(X_1,\dots,X_{i-1},v,X_{i+1},\dots,X_m)$

The above rule must be enriched if one wants to take account of the fact that the execution of the assignement statement may fail, (e.g. the expression E might overflow).

Test statements :

$\{P(X,\bar{X})\}$ $\underline{\text{if}}$ $Q(X)$ $\underline{\text{then}}$ $\{P(X,\bar{X})\ \underline{\text{and}}\ Q(X)\}$ ... $\underline{\text{else}}$ $\{P(X,\bar{X})\ \underline{\text{and}}\ \underline{\text{not}}\ Q(X)\}$ ... $\underline{\text{fi}}$;

Go to statements and labels :

$$L : \left\{ \underset{i\in \underline{\text{pred}}(L)}{\underline{\text{OR}}}\ P_i(X,\bar{X}) \right\}$$

where $\underline{\text{pred}}(L)$ denotes the set of program points going to L sequentially or by jumps.

3.3. EXAMPLE

We illustrate the application of the above rules to a very simple program (over the set $\mathbb{I}$ of integers augmented by Ω).

```
{P1(x,y,x̄,ȳ)}   while x≥y do
{P2(x,y,x̄,ȳ)}      x:=x-y;
{P3(x,y,x̄,ȳ)}   od;
{P4(x,y,x̄,ȳ)}
```

Rewritting this program segment with branch and test primitives, and applying the rules of the deductive semantics we get a system of equations which can be simplified as follows :

$$\begin{cases} P_1(x,y,\bar{x},\bar{y}) = (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y}) \\ P_2(x,y,\bar{x},\bar{y}) = (P_1(x,y,\bar{x},\bar{y})\ \underline{\text{or}}\ P_3(x,y,\bar{x},\bar{y}))\ \underline{\text{and}}\ (x\ge y) \\ P_3(x,y,\bar{x},\bar{y}) = \{\exists v\in \mathbb{I}: P_2(v,y,\bar{x},\bar{y})\ \underline{\text{and}}\ (x=v-y)\} \\ P_4(x,y,\bar{x},\bar{y}) = (P_1(x,y,\bar{x},\bar{y})\ \underline{\text{or}}\ P_3(x,y,\bar{x},\bar{y}))\ \underline{\text{and}}\ (x<y) \end{cases}$$

(Hereafter we will write P_i instead of $P_i(x,y,\bar{x},\bar{y})$ in order to simplify the notations).

3.4. OPTIMAL INVARIANTS

According to Tarski's theorem the above system of equations of the form $P=F(P)$ has a least solution P^{opt} (least for ordering $\Leftarrow$ that is $\Rightarrow$). We call P^{opt} the *set of optimal invariants* since they imply any other solution to the system of equations, (for all other sets of invariants P such that $P=F(P)$, P^{opt} is a lower bound therefore $P^{opt}\Rightarrow P$). The optimal invariants are the limit of any chaotic iterations starting from the infimum $\underline{\text{false}}$ (theorem 2.2). Let us choose Gauss-Seidel's method :

$$\left[\begin{array}{ll} P_1^k = F(P_1^{k-1},\dots\dots\dots\dots,P_n^{k-1}) & \\ \dots & \\ P_i^k = F(P_1^k,\dots,P_{i-1}^k,P_i^{k-1},\dots,P_n^{k-1}) & (k=1,2,\dots) \\ \dots & \\ P_n^k = F(P_1^k,\dots\dots\dots\dots P_{n-1}^k,P_n^{k-1}) & \end{array}\right.$$

(Coming back to 2.1, this consists in choosing $J^i=\{(i \text{ modulo } n)+1)\}$ for $i=0,1,\dots$ a step k summarizing n primitive iterations).

Initialization :

$$\left[\ P_i^0 = \underline{\text{false}} \quad (i=1..4)\right.$$

Step 1 :

$$\left[\begin{array}{l} P_1^1 = (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y}) \\ P_2^1 = (\bar{x}\ge\bar{y})\ \underline{\text{and}}\ (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y}) \\ P_3^1 = (\bar{x}\ge\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-\bar{y})\ \underline{\text{and}}\ (y=\bar{y}) \\ P_4^1 = (\bar{x}<\bar{y})\ \underline{\text{and}}\ (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y}) \end{array}\right.$$

Step 2 :

$$\left[\begin{array}{l} P_1^2 = (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y}) \\ P_2^2 = [(\bar{x}\ge\bar{y})\ \underline{\text{and}}\ (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y})] \\ \quad \underline{\text{or}} \\ \quad [(\bar{x}\ge\bar{y})\ \underline{\text{and}}\ (\bar{x}\ge 2\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-\bar{y})\ \underline{\text{and}}\ (y=\bar{y})] \\ P_3^2 = [(\bar{x}\ge\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-\bar{y})\ \underline{\text{and}}\ (y=\bar{y})] \\ \quad \underline{\text{or}} \\ \quad [(\bar{x}\ge\bar{y})\ \underline{\text{and}}\ (\bar{x}\ge 2\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-2\bar{y})\ \underline{\text{and}}\ (y=\bar{y})] \\ P_4^2 = [(\bar{x}<\bar{y})\ \underline{\text{and}}\ (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y})] \\ \quad \underline{\text{or}} \\ \quad [(\bar{x}\ge\bar{y})\ \underline{\text{and}}\ (\bar{x}<2\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-\bar{y})\ \underline{\text{and}}\ (y-\bar{y})] \end{array}\right.$$

By computing these first few terms we seek to discover the general term P^i of the sequence :

Step i :

$$\left[\begin{array}{l} P_1^i = (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y}) \\ P_2^i = \underset{j=1}{\overset{i}{\underline{\text{OR}}}}\,(\underset{k=1}{\overset{j}{\underline{\text{AND}}}}\,(\bar{x}\ge k\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-(j-1)\bar{y})\ \underline{\text{and}}\ (y=\bar{y})) \\ P_3^i = \underset{j=1}{\overset{i}{\underline{\text{OR}}}}\,(\underset{k=1}{\overset{j}{\underline{\text{AND}}}}\,(\bar{x}\ge k\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-j\bar{y})\ \underline{\text{and}}\ (y=\bar{y})) \\ P_4^i = \underset{j=1}{\overset{i}{\underline{\text{OR}}}}\,(\underset{k=1}{\overset{j-1}{\underline{\text{AND}}}}\,(\bar{x}\ge k\bar{y})\ \underline{\text{and}}\ (\bar{x}< j\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-(j-1)\bar{y}) \\ \qquad\qquad \underline{\text{and}}\ (y=\bar{y})\} \end{array}\right.$$

This general term P^i of the chaotic iterations is proved to be correct by mathematical induction : *Basis* : P^1 and P^2 are of the form specified by P^i for i=1 and i=2. *Induction step* : assuming P^i to be correct and substituting in the right hand side of the equations we show that after simplifications we obtain P^{i+1}. Then according to theorem 2.3 the optimal invariants are obtained by $P^{opt} = \lim_{i\to\infty} P^i$, we get :

$$\begin{cases} P_1^{opt} = (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y}) \\ P_2^{opt} = \{\exists j\geq 1:(\forall k\in[1,j],\bar{x}\geq k\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-(j-1)\bar{y})\ \underline{\text{and}}\ (y=\bar{y})\} \\ P_3^{opt} = \{\exists j\geq 1:(\forall k\in[1,j],\bar{x}\geq k\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-j\bar{y})\ \underline{\text{and}}\ (y=\bar{y})\} \\ P_4^{opt} = \{\exists j\geq 1:(\forall k\in[1,j-1],\bar{x}\geq k\bar{y})\ \underline{\text{and}}\ (\bar{x}<j\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-(j-1)\bar{y})\ \underline{\text{and}}\ (y=\bar{y})\} \end{cases}$$

4. PROOFS OF TOTAL CORRECTNESS

If for any input values $\bar{X}$, there exists a haltpoint h where the set S_h of possible states of variables is not empty (i.e. $(\exists\bar{Y}:\bar{Y}\in S_h)$) the program must terminate at the haltpoint h (with a final state $\bar{Y}$ of the variables X). Since the set of optimal invariants describes precisely the exact domain of the variables at each program point, the termination condition is $\{\forall\bar{X},\exists h,\exists\bar{Y} : P_h^{opt}(\bar{Y},\bar{X})\}$.

Assume now that the intended behaviour of the program is specified by means of an input specification ϕ and an output specification ψ. The intention is that for any initial values $\bar{X}$ of the variables satisfying the input specification $\phi(\bar{X})$ the program terminates with final values $\bar{Y}$ of the variables satisfying $\psi(\bar{Y},\bar{X})$. Therefore the partial correctness condition is $\{(\forall\bar{X}:\phi(\bar{X})),\ \forall h,\ \exists\bar{Y} : P_h^{opt}(\bar{Y},\bar{X}) \Rightarrow \psi(\bar{Y},\bar{X})\}$.

Since $\{P_h^{opt}(\bar{Y},\bar{X})\ \underline{\text{and}}\ (P_h^{opt}(\bar{Y},\bar{X}) \Rightarrow \psi(\bar{Y},\bar{X}))\}$ is equivalent to $\{P_h^{opt}(\bar{Y},\bar{X})\ \underline{\text{and}}\ \psi(\bar{Y},\bar{X})\}$ the total correctness condition is : $\{(\forall\bar{X}:\phi(\bar{X})),\ \exists h,\ \exists\bar{Y} : P_h^{opt}(\bar{Y},\bar{X})\ \underline{\text{and}}\ \psi(\bar{Y},\bar{X})\}$.

Example : The input condition guaranteeing the termination of the simple program 3.3 is : $\phi(\bar{x},\bar{y}) = \{\exists x,y : P_4^{opt}(x,y,\bar{x},\bar{y})\} = \{\exists j\geq 1 : (\forall k\in[1,j-1],\bar{x}\geq k\bar{y})\ \underline{\text{and}}\ (\bar{x}<j\bar{y})\} = \{(0<\bar{y})\ \underline{\text{or}}\ (\bar{x}<\bar{y}\leq 0)\}$. *End of Example.*

5. SYSTEMS OF IMPLICATIONS, APPROXIMATE INVARIANTS AND PROOFS OF PARTIAL CORRECTNESS

Most program verification methods use inequalities of the form $P \Leftarrow F(P)$ whereas we used equalities $P=F(P)$. For example instead of {x>0} x:=x+2 {x>2} one can legally write less precise assertions such as {x>0} x:=x+2 {x>1} since the strongest post-condition resulting from the pre-condition {x>0} is {x>2} which implies {x>1}.

According to Tarski's theorem $P^{opt} = \underline{\text{AND}}\{P:P \Leftarrow F(P)\}$, hence $\{\forall P:P \Leftarrow F(P)\}$ we have $P^{opt} \Rightarrow P$. A proof of partial correctness consists in proving that $\{(\forall\bar{X}:\phi(\bar{X})),\ \forall h,\ \forall\bar{Y} : P_h^{opt}(\bar{Y},\bar{X}) \Rightarrow \psi(\bar{Y},\bar{X})\}$. If the programmer can provide a set of approximate invariants P (eventually the loop invariants only since the remaining can be deduced by a simple propagation in the recursive equations) and if it can be verified that $P \Leftarrow F(P)$ then the proof that $\{\forall h, P_h(\bar{Y},\bar{X}) \Rightarrow \psi(\bar{Y},\bar{X})\}$ constitutes a proof of partial correctness since we have shown that $P_h^{opt}(\bar{Y},\bar{X}) \Rightarrow P_h(\bar{Y},\bar{X})$. Hence the program is partially correct with respect to ϕ and ψ if and only if :
$\{(\exists P:P \Leftarrow F(P)),\ (\forall\bar{X}:\phi(\bar{X})),\ \forall h,\ \forall\bar{Y},\ P_h(\bar{Y},\bar{X}) \Rightarrow (\bar{Y},\bar{X})\}$
which is the condition given in Katz & Manna[1976].

Example : Suppose we want to prove the partial correctness of the program 3.3 with respect to the input specification $\phi(\bar{x},\bar{y})=\{(\bar{x}\geq 0)\ \underline{\text{and}}\ (\bar{y}\geq 0)\}$ and the output specification $\psi(x,y,\bar{x},\bar{y})=\{y>x\geq 0\}$. Choosing the loop invariant P_2 to be :
$P_2 = (x\geq y)\ \underline{\text{and}}\ (y=\bar{y})$
and propagating in the equations we get :
$P_3 = (x\geq 0)\ \underline{\text{and}}\ (y=\bar{y})$
$P_4 = (x<y)\ \underline{\text{and}}\ (y=\bar{y})\ \underline{\text{and}}\ [(x=\bar{x})\ \underline{\text{or}}\ (x\geq 0)]$
It is easy to verify that $\{(P_1\ \underline{\text{or}}\ P_3)\ \underline{\text{and}}\ (x\geq y)\} \Rightarrow P_2$ so that P_2 is a correct approximate loop invariant. Finally $(\forall\bar{x}\geq 0)$ we have $P_4(x,y,\bar{x},\bar{y}) \Rightarrow \psi(x,y,\bar{x},\bar{y})$, (although the program does not terminate for y=0). *End of Example.*

The termination condition $\{(\forall\bar{X}:\phi(\bar{X})),\exists h,\exists\bar{Y} : P_h^{opt}(\bar{Y},\bar{X})\}$ can be expressed using approximate invariants. Since $P^{opt} = \underline{\text{AND}}\{P : P \Leftarrow F(P)\}$ we can write :

$$\{(\forall\bar{X}:\phi(\bar{X})),\exists h,\exists\bar{Y} : \underline{\text{AND}}\{P_h(\bar{Y},\bar{X}) : P \Leftarrow F(P)\}\}$$
$$= \{(\forall P : P \Leftarrow F(P)),(\forall\bar{X}:\phi(\bar{X})),\exists h,\exists\bar{Y} : P_h(\bar{Y},\bar{X})\}$$

which is the termination condition of Katz & Manna [1976]. They observed that this condition is not utilizable in practice since it is expressed in terms of the infinitely many approximate invariants satisfying $P \stackrel{\leftarrow}{=} F(P)$. This is not surprising since this condition is based on Tarski's theorem which does not provide an algorithmic construction of the least fixpoint of F.

6. SYMBOLIC EXECUTION

The purpose of this section is to show that symbolic execution of a program consists in solving the semantic equations associated with this program by chaotic iterations.

6.1. SYMBOLIC CONTEXTS

Observe (3.4) that the invariant P_i associated with a program point i can be expressed in the normal form $P_i = \underline{\mathrm{OR}}_{j\in\Delta}\ p_j$ where each p_j is of the form $(Q_j\ \underline{\mathrm{and}}\ (X_1=E_{1j})\ \underline{\mathrm{and}}\ \ldots\ \underline{\mathrm{and}}\ (X_m=E_{mj}))$. Each p_j describes a program path which may lead to the point i. For each program path p_j an assertion Q_j states the condition which had to be satisfied in order for that path to be executed. At point i on that path the value of the program variable X_k, (k=1..m) is given by E_{kj}. E_{kj} is a formal expression depending on the initial values $\overline{X}$ of the variables on program entry. No X_k can appear as a free variable neither in Q_j nor in the E_{kj}. Slightly changing the notations we will call P_i a *symbolic context* and rewrite it as $P_i=\{p_j : j\in\Delta\}$ where $p_j=\langle Q_j, E_{1j}, \ldots, E_{mj}\rangle$.

6.2. SYSTEM OF SYMBOLIC FORWARD EQUATIONS

Using now the notation of symbolic contexts the rules of the deductive semantics (3.2) must be adapted so that they transform an input predicate in normal form into an output predicate in normal form. For example, the output predicate corresponding to the input predicate

$$\{\langle Q_j, E_{1j}, \ldots, E_{mj}\rangle : j\in\Delta\}$$
$$= \{\underline{\mathrm{OR}}_{j\in\Delta}(Q_j\ \underline{\mathrm{and}}\ (X_1=E_{1j})\ \underline{\mathrm{and}}\ \ldots\ \underline{\mathrm{and}}\ (X_m=E_{mj}))\}$$

after the assignment statement $X_k:=E(X_1,\ldots,X_m)$ is :

$$\{\exists v: \underline{\mathrm{OR}}_{j\in\Delta}(Q_j\ \underline{\mathrm{and}}\ (X_1=E_{1j})\ \underline{\mathrm{and}}\ \ldots\ \underline{\mathrm{and}}\ (v=E_{kj})\ \underline{\mathrm{and}}\ \ldots\ \underline{\mathrm{and}}\ (X_m=E_{mj}))\ \underline{\mathrm{and}}\ X_k = E(X_1,\ldots,v,\ldots X_m)\}$$

Eliminating the free variables in E as well as the intermediate bound variable v we get :

$$\{\underline{\mathrm{OR}}_{j\in\Delta}(Q_j\ \underline{\mathrm{and}}\ \ldots\ \underline{\mathrm{and}}\ (X_k=E(E_{1j},\ldots,E_{kj},\ldots,E_{mj}))\ \underline{\mathrm{and}}\ \ldots\ \underline{\mathrm{and}}\ (X_m=E_{mj}))\}$$
$$= \{\langle Q_j, E_{1j}, \ldots, E(E_{1j},\ldots,E_{mj}), \ldots, E_{mj}\rangle : j\in\Delta\}$$

We will denote this rule by the shorthand notation :

$$\{P\}\ X_k:=E(X_1,\ldots,X_m)\ \{P(X_k \leftarrow E(P(X_1),\ldots,P(X_m)))\}$$

However, when there is no ambiguity on which context must be used to evaluate the expression E we will write more simply :

$$\{P\}\ X_k:=E(X_i,\ldots,X_m)\ \{P(X_k \leftarrow E(X_1,\ldots,X_m))\}.$$

The other rules can be deduced in the same way. In particular, the operation $\sqcup$ describes the union $P \sqcup Q$ of two symbolic contexts $P=\{p_1,\ldots,p_r\}$ and $Q=\{q_1,\ldots,q_s\}$ that is the set $\{p_1,\ldots,p_r,q_1,\ldots,q_s\}$ where superfluous equivalent program paths are eliminated whereas inaccessible paths (the path condition of which is false) are removed.

6.3. EXAMPLE

Using again the example 3.3 :

```
      {0}
loop: {1}  if x≥y then
      {2}     x:=x-y;
      {3}     go to loop;
      {4}  fi;
```

we have the system of equations :

$$\begin{cases} P_0 = \{\langle \underline{\mathrm{true}}, \overline{x}, \overline{y}\rangle\} \\ P_1 = P_0 \sqcup P_3 \\ P_2 = P_1\ \underline{\mathrm{and}}\ (x\geq y) \\ P_3 = P_2(x \leftarrow (x-y)) \\ P_4 = P_1\ \underline{\mathrm{and}}\ (x<y) \end{cases}$$

6.4. SYMBOLIC EXECUTION TREE

As in 3.4 we solve these equations using chaotic iterations with a Gauss-Seidel policy :

Initialization :

$$\left[\ P_i^0 = \emptyset \qquad (i=0..4)\right.$$

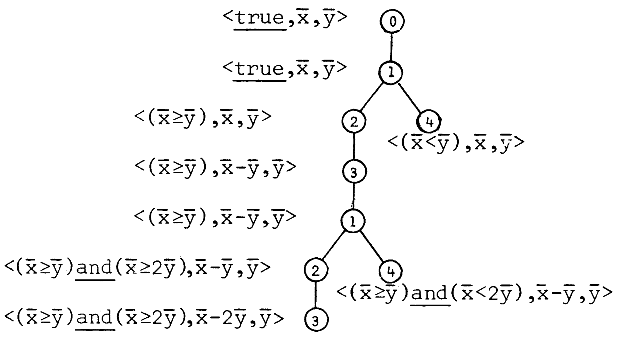
Step 1 :

$$\left[\begin{array}{l} P_0^1 = \{\langle \underline{\mathrm{true}}, \overline{x}, \overline{y}\rangle\} \\ P_1^1 = P_0^1 \sqcup P_3^0 = \{\langle \underline{\mathrm{true}}, \overline{x}, \overline{y}\rangle\} \\ P_2^1 = P_1^1\ \underline{\mathrm{and}}\ (x\geq y) \quad = \{\langle (\overline{x}\geq\overline{y}), \overline{x}, \overline{y}\rangle\} \\ P_3^1 = P_2^1(x \leftarrow (x-y)) = \{\langle (\overline{x}\geq\overline{y}), \overline{x}-\overline{y}, \overline{y}\rangle\} \\ P_4^1 = P_1^1\ \underline{\mathrm{and}}\ (x<y) \quad = \{\langle (\overline{x}<\overline{y}), \overline{x}, \overline{y}\rangle\} \end{array}\right.$$

Step 2 :

$$\left[\begin{array}{l} P_0^2 = \{<\underline{true},\bar{x},\bar{y}>\} \\ P_1^2 = \{<\underline{true},\bar{x},\bar{y}>,<(\bar{x}\geq\bar{y}),\bar{x}-\bar{y},\bar{y}>\} \\ P_2^2 = \{<(\bar{x}\geq\bar{y}),\bar{x},\bar{y}>,<(\bar{x}\geq\bar{y})\ \underline{and}\ (\bar{x}\geq 2\bar{y}),\bar{x}-\bar{y},\bar{y}>\} \\ P_3^2 = \{<(\bar{x}\geq\bar{y}),\bar{x}-\bar{y},\bar{y}>,<(\bar{x}\geq\bar{y})\ \underline{and}\ (\bar{x}\geq 2\bar{y}),\bar{x}-2\bar{y},\bar{y}>\} \\ P_4^2 = \{<(\bar{x}<\bar{y}),\bar{x},\bar{y}>,<(\bar{x}\geq\bar{y})\ \underline{and}\ (\bar{x}<2\bar{y}),\bar{x}-\bar{y},\bar{y}>\} \end{array}\right.$$

So that at iteration 2 we have built the following symbolic execution tree (Hantler & King[1976]) :



We have represented the symbolic context P_i associated with program point i by the set of paths associated with each of the nodes labelled i in the above execution tree. Equivalently we could have represented the symbolic context associated with program point i by the maximal subtree (of the above symbolic tree) the leaves of which are labelled i. Then the union $\sqcup$ of symbolic contexts performed at junction of program paths would be the merging of symbolic execution trees.

It is clear that the computation of the next terms of the sequence of chaotic iterations would cause the symbolic execution tree to grow. We can make the tree to grow in whatever direction we want, the result will be the same (2.2). Without particular hypothesis on $\bar{x}$ and $\bar{y}$ this process would converge to the optimal invariants in infinitely many steps (2.3). Therefore we must be able either to reason about the limit without knowing it (6.4) or to directly pass to the limit (7).

6.5. VERIFICATION OF PROPERTIES OF OPTIMAL SYMBOLIC CONTEXTS

Coming back to the notations of §2.3 in order to prove a property $P(X^\infty)$ of the solution to the system of equations X=F(X) we can prove by induction that all terms X^{km} of a chaotic iteration sequence have this property :

$$\{\{P(X^0)\ \underline{or}\ P(X^m)\}\ \underline{and}\ \{\forall k, P(X^{km}) \Rightarrow P(X^{(k+1)m})\}\} \Rightarrow \{P(\lim_{k\to\infty} X^{km})\}$$

(Yet P must be an admissible predicate chosen in order to remain true when passing to the limit. Rigorously we should apply the second principle of transfinite induction).

Example : Let us prove the trivial fact that $\{\forall a\in[1,(\bar{x}-x)/y], \bar{x}\geq ay\}$ at point 3 of program 6.3.
Basis : For the single path of P_3^1 we have $\{\forall a\in[1,1],\bar{x}\geq ay\}$.
Induction step : We assume that at step k the symbolic context P_3^k is equal to $\{<p_i,x_i,y_i> : i\in D\}$ with the induction hypothesis $\{\forall i\in D,\{\forall a\in[1,(\bar{x}-x_i)/y_i], \bar{x}\geq ay_i\}\}$. The equations 6.3 allow the computation of P_3^{k+1} :

$$P_3^k = \{<p_i,x_i,y_i>:i\in D\}$$
$$P_1^{k+1} = \{<\underline{true},\bar{x},\bar{y}>,<p_i,x_i,y_i>:i\in D\}$$
$$P_2^{k+1} = \{<(\bar{x}\geq\bar{y}),\bar{x},\bar{y}>,<(p_i\ \underline{and}\ x_i\geq y_i),x_i,y_i>:i\in D\}$$
$$P_3^{k+1} = \{<(\bar{x}\geq\bar{y}),\bar{x}-\bar{y},\bar{y}>,<(p_i\ \underline{and}\ x_i\geq y_i),x_i-y_i,y_i>:i\in D\}$$

We must show that the hypothesis holds for all paths of P_3^{k+1}. This is trivial for the path $<(\bar{x}\geq\bar{y}),\bar{x}-\bar{y},\bar{y}>$. Otherwise we must show that $\{\forall i\in D,\{\forall a\in[1,(\bar{x}-x_i)/y_i+1], \bar{x}\geq ay_i\}\}$. According to the induction hypothesis, this condition is true $\forall a\in[1,(\bar{x}-x_i)/y_i]$. Finally for $a=(\bar{x}-x_i)/y_i+1$, the path condition $x_i\geq y_i$ implies $\bar{x}\geq ay_i$.

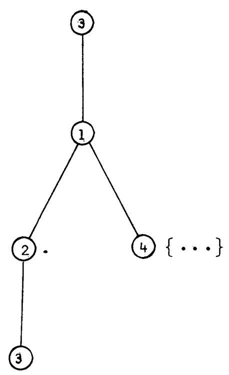
This approach for reasoning about the limit of chaotic iterations is implicitly used in the technique of "cut-trees" of Hantler & King[1976]. Indeed the induction step can be understood as consisting in reasoning on the cut tree for {3} :

$\{<p_i,x_i,y_i>:i\in D\}$

$\{<\underline{true},\bar{x},\bar{y}>,$
$<p_i,x_i,y_i>:i\in D\}$

$\{<(\bar{x}\geq\bar{y}),\bar{x},\bar{y}>,$
$<p_i\ \underline{and}\ (x_i\geq y_i),x_i,y_i>:i\in D\}$

$\{<(\bar{x}\geq\bar{y}),\bar{x}-\bar{y},\bar{y}>$
$<p_i\ \underline{and}\ (x_i\geq y_i),x_i-y_i,y_i>:i\in D\}$



End of Example.

7. SYNTHESIS OF OPTIMAL INVARIANT ASSERTIONS : THE USE OF DIFFERENCE EQUATIONS

7.1. DISCOVERY OF OPTIMAL SYMBOLIC CONTEXTS

The equations of example 6.3 can be written as :

$$P_1 = P_0 \sqcup (P_1 \; \underline{\text{and}} \; x \geq y)(x \leftarrow x-y)$$

they are of the form :

$$P_1 = f_1(P_0) \sqcup f_2(P_1)$$

The resolution by successive approximations was :

$$P_1^0 = \emptyset$$

$$P_1^1 = f_1(P_0) \sqcup f_2(\emptyset) = f_1(P_0)$$

$$P_1^2 = f_1(P_0) \sqcup f_2 f_1(P_0)$$

$$P_1^3 = f_1(P_0) \sqcup f_2(f_1(P_0) \sqcup f_2 f_1(P_0))$$

$$= f_1(P_0) \sqcup f_2 f_1(P_0) \sqcup f_2^2 f_1(P_0)$$

since f_2 is distributive over $\sqcup$. (This comes from the fact that f_2 is the composition of elementary functions as defined in 6.2. Setting apart the difference in notations they are similar to the predicate transformers of the deductive semantics (3.2). Since the set of predicates form a complete boolean lattice the infinite distributive laws holds for $\underline{\text{OR}}$ and $\underline{\text{AND}}$).

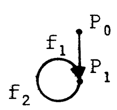
The general term of the approximation sequence is

$$P_1^k = \bigsqcup_{i=0}^{k-1} f_2^i f_1(P_0)$$

since

$$P_1^{k+1} = f_1(P_0) \sqcup f_2\Big(\bigsqcup_{i=0}^{k-1} f_2^i f_1(P_0)\Big) = \bigsqcup_{i=0}^{k} f_2^i f_1(P_0)$$

Passing to the limit we get $P_1^\infty = \bigsqcup_{i=0}^{\infty} f_2^i f_1(P_0)$. This result can be obtained directly since the graph of dependence between P_1 and P_0 :



can be considered as (in general a non-deterministic) finite automaton which recognizes the Kleene's language $f_1 f_2^*$. We have obtained a symbolic formulation of the desired solution, but it remains to give the explicit formulation of the function $f_2^i \circ f_1$. This can be done by solving the difference equations :

$$\begin{cases} f_2^0 \circ f_1(P_0) \doteq P_0 \\ f_2^{i+1} \circ f_1(P_0) = f_2(f_2^i \circ f_1(P_0)) \end{cases}$$

We have :

$$f_2^0 \circ f_1(P_0) = \{<\underline{\text{true}}, \bar{x}, \bar{y}>\}$$

$$= \{<p_{0j}, x_{0j}, y_{0j}>: j \in \{1\}\}$$

Let $f_2^i \circ f_1(P_0)$ be $\{<p_{ij}, x_{ij}, y_{ij}>: j \in D_i\}$, we have

$$f_2^{i+1} \circ f_1(P_0) = f_2(\{<p_{ij}, x_{ij}, y_{ij}>: j \in D_i\})$$

$$= \{<p_{ij} \; \underline{\text{and}} \; x_{ij} \geq y_{ij}, x_{ij}-y_{ij}, y_{ij}>: j \in D_i\}$$

First of all, we can determine the domain of j which indexes the possible paths. Since $D_0=\{1\}$ and $D_{i+1}=D_i$ we have $D_i=\{1\}$ (because there is a single path within the loop). Therefore we can simply ignore the path index and solve the difference equations (in the order of dependence) :

$$\begin{cases} y_0=\bar{y}, & y_{i+1}=y_i \\ x_0=\bar{x}, & x_{i+1}=x_i-y_i \\ p_0=\underline{\text{true}}, & p_{i+1}=p_i \; \underline{\text{and}} \; (x_i \geq y_i) \end{cases}$$

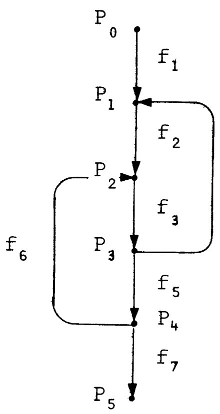
These recurrence relations can be solved directly (e.g. Cohen & Katcoff[1976]) yielding $y_i=y_0$, $x_i=x_0-iy_0$, and since $p_0=\underline{\text{true}}$ and $p_{i+1}=p_i \; \underline{\text{and}} \; x_0 \geq (i+1)y_0$ we get $p_i = \underline{\text{AND}}_{j=1}^{i}(x_0-jy_0) = \{\forall j \in [1,i], (x_0 \geq jy_0)\}$. Finally the optimal loop invariant of program 6.3 is :

$$P_1^\infty = \bigsqcup_{i=0}^{\infty} \{<(\forall j \in [1,i], x_0 \geq jy_0), x_0-iy_0, y_0>\}$$

The "difference equations method" was introduced by Elspas, Green, Levitt & Waldinger[1972], Elspas[1974]. It is further used in Greif & Waldinger [1975], Katz & Manna[1976] (algorithmic approach), Cheatham & Townley[1976]. It has also been used in determining symbolic expression of program complexity (Wegbreit[1975]). However, the technique was understood with respect to an operational semantics, i.e. by reasoning on (dummy) loop counters and on approximate invariants. In fact a reasoning in terms of denotational semantics clearly shows that (at least in theory) optimal invariants can be discovered and consequently program termination can be proved or disproved.

7.2. REMARK

We have proved that in theory all chaotic iteration sequences lead to the optimal solution to the system of equations. However, in practice a major difficulty consists in finding a form of the general term of the approximation sequence which is suitable for establishing the difference equations. Consider for example a system of equations with the following dependence graph :



We can express P_1^∞ by the following equivalent forms :

$$P_1^\infty = \bigsqcup_{\substack{i\geq 0\\ j\geq 0}} (f_4 \circ f_3 \circ (f_6 \circ f_5 \circ f_3)^i \circ f_2)^j \circ f_1(P_0)$$

(i and j correspond to individual counters for the loops)

$$P_1^\infty = \bigsqcup_{k\geq 0} (f_1(P_0) \sqcup f_4 \circ f_3 \circ ((f_2 \circ f_4 \sqcup f_6 \circ f_5) \circ f_3)^k \circ f_2 \circ f_1(P_0)$$

(k is a common counter for the two loops).

Although these two forms of P_1^∞ are equivalent one of them will be more suitable for generating the difference equations and the choice depends on the semantics of the considered program.

7.3. EXAMPLE

Let us consider the following program (taken from King[1969]) : (÷ is integer division with truncation)

```
              z:=1;
       {0}
loop:  {1}
       {2}    if y≠0 then
       {3}       if odd(y) then
       {4}          z:=z*x;
       {5}       fi;
       {6}       (y,x):=(y÷2,x²);
                 go to loop;
       {7}    fi;
```

The system of equations associated with this program is :

$$\begin{cases} P_0 = \{<\underline{true}, x=a, y=b, z=1>\} \\ P_1 = P_0 \sqcup P_6 \\ P_2 = P_1 \ \underline{and}\ (P_1(y)\neq 0) \\ P_3 = P_2 \ \underline{and}\ \underline{odd}(P_2(y)) \\ P_4 = P_3(z \leftarrow P_3(z) * P_3(x)) \\ P_5 = (P_2 \ \underline{and}\ \underline{even}(P_2(y))) \sqcup P_4 \\ P_6 = P_5(y \leftarrow P_5(y) \div 2, x \leftarrow (P_5(x))^2) \\ P_7 = P_1 \ \underline{and}\ (P_1(y)=0) \end{cases}$$

Since $P_2(y)=P_1(y)$, $P_3(z)=P_1(z)$, $P_3(x)=P_1(x)$, $P_5(y)=P_1(y)$ and $P_5(x)=P_1(x)$ we can simplify as follows :

$$\begin{aligned} P_1 = {} & P_0 \\ & \sqcup \\ & (P_1 \ \underline{and}\ P_1(y)\neq 0 \ \underline{and}\ \underline{even}(P_1(y)))(y \leftarrow P_5(y)\div 2, x \leftarrow (P_5(x))^2, z \leftarrow P_1(z)) \\ & \sqcup \\ & (P_1 \ \underline{and}\ P_1(y)\neq 0 \ \underline{and}\ \underline{odd}(P_1(y)))(y \leftarrow P_1(y)\div 2, x \leftarrow (P_5(x))^2, z \leftarrow P_1(z)*P_1(x)) \end{aligned}$$

Since the two alternatives differ only with respect to z we can factorize as follows :

$$\begin{aligned} P_1 &= P_0 \sqcup (P_1 \ \underline{and}\ y\neq 0)(x \leftarrow x^2, y \leftarrow y\div 2, z \leftarrow z*(\underline{even}(y) \rightarrow 1 \sqcup \underline{odd}(y) \rightarrow x)) \\ &= P_0 \sqcup f(P_1) \end{aligned}$$

This formulation uses the conditional expressions of Sintzoff[1975] :

- $(\underline{true} \rightarrow v \ \sqcup\ Q \rightarrow v') = v \sqcup (Q \rightarrow v')$
- $(\underline{false} \rightarrow v \ \sqcup\ Q \rightarrow v') = (Q \rightarrow v')$
- $Q \sqcup Q' = Q \ \underline{or}\ Q'$
- $Q \rightarrow Q' = Q \ \underline{and}\ Q'$
- $\bigsqcup_i (Q_i \rightarrow v) = (\underline{OR}_i\, Q_i \rightarrow v)$
- $\bigsqcup_i (Q \rightarrow v_i) = Q \rightarrow \bigsqcup_i v_i$
- $Q \rightarrow (Q' \rightarrow v) = (Q \rightarrow Q') \rightarrow v$
- $f(v_1,\ldots,\bigsqcup_i w_i,\ldots,v_n) = \bigsqcup_i f(v_1,\ldots,w_i,\ldots,v_n)$
- $f(v_1,\ldots,(Q \rightarrow w),\ldots,v_n) = (Q \rightarrow f(v_1,\ldots,w,\ldots,v_n))$
- etc.

The general form of the solution for the equation defining P_1 is $P_1 = \bigsqcup_{i\geq 0} f^i(P_0)$. For determining $f^i(P_0)$ we use the difference equations :

$$\begin{cases} f^0(P_0) = <\underline{true}, a, b, 1> \\ \qquad\quad = <q_0, x_0, y_0, z_0> \\ f^{i+1}(P_0) = f_2(<q_i, x_i, y_i, z_i>) \\ \qquad\quad = <q_i \ \underline{and}\ (y_i\neq 0), x_i^2, y_i\div 2, z_i*(\underline{even}(y_i) \rightarrow 1 \sqcup \underline{odd}(y_i) \rightarrow x_i)> \\ \qquad\quad = <q_{i+1}, x_{i+1}, y_{i+1}, z_{i+1}> \end{cases}$$

The formal resolution proceeds as follows :

$$\begin{cases} x_0 = a \\ x_{i+1} = x_i^2 \end{cases} \Rightarrow x_i = a^{2^i}$$

$$\begin{cases} y_0 = b \\ y_{i+1} = y_i \div 2 \end{cases} \Rightarrow y_i = b \div 2^i$$

$$\begin{cases} z_0 = 1 \\ z_{i+1} = z_i*(\underline{even}(b\div 2^i) \rightarrow 1 \sqcup \underline{odd}(b\div 2^i) \rightarrow a^{2^i}) \end{cases}$$

$$\Rightarrow z_i = \prod_{j=0}^{i-1} (\underline{even}(b\div 2^j) \rightarrow 1 \ \sqcup\ \underline{odd}(b\div 2^j) \rightarrow a^{2^j})$$

$$\begin{cases} q_0 = \underline{true} \\ q_{i+1} = q_i \ \underline{and}\ (b\div 2^i)\neq 0 \end{cases}$$

$$\Rightarrow q_i = \underline{AND}_{j=0}^{i-1} (b\div 2^j \neq 0)$$

$$\Rightarrow q_i = (b\div 2^{i-1} \neq 0) \quad \text{when} \quad (i>0)$$

Finally the optimal output invariant of the program is :

$$P_7 = P_1 \ \underline{\text{and}}\ (y=0)$$
$$= \bigsqcup_{i=0}^{\infty} \langle q_i \ \underline{\text{and}}\ (y_i=0), x_i, y_i, z_i \rangle$$
$$= \langle b=0, x=a, y=b, z=1 \rangle$$
$$\sqcup$$
$$\langle (\exists i>0 : (b \div 2^{i-1} \neq 0)\ \underline{\text{and}}\ (b \div 2^i = 0)),$$
$$x = a^{2^i},$$
$$y = (b \div 2^i),$$
$$z = \prod_{j=0}^{i} (\underline{\text{even}}(b \div 2^j) \to 1 \sqcup \underline{\text{odd}}(b \div 2^j) \to a^{2^j}) \rangle$$

Since the path condition at the haltpoint {7} :
$(b=0)\ \underline{\text{or}}\ (\exists i>0 : (b \div 2^{i-1} \neq 0)\ \underline{\text{and}}\ (b \div 2^i = 0)$
is true for any input value b of y, the program always terminates.

It remains to show that $P_7(z)=a^{|b|}$ to prove total correctness. In the non-obvious case $b \neq 0$, we have to know that every number b can be expressed in binary form :

$$b = ((\exists i : (b \div 2^{i-1} \neq 0)\ \underline{\text{and}}\ (b \div 2^i = 0)) \to \underline{\text{sign}}(b) * \sum_{j=0}^{i-1} (\underline{\text{even}}(b \div 2^j) \to 0 \sqcup \underline{\text{odd}}(b \div 2^j) \to 2^j))$$

Evaluating $a^{|b|}$ using the property that :

$$a^{\sum_i \alpha_i} = \prod_i a^{\alpha_i}$$

we get :

$$a^{|b|} = ((\exists i : (b \div 2^{i-1} \neq 0)\ \underline{\text{and}}\ (b \div 2^i = 0)) \to \prod_{j=0}^{i-1} (\underline{\text{even}}(b \div 2^j) \to a^0 \sqcup \underline{\text{odd}}(b \div 2^j) \to a^{2^j}))$$

which is the desired result.

8. SYNTHESIS OF APPROXIMATE INVARIANTS

Let us consider a system of equations X=F(X) associated with a given program.We have seen that a set P of approximate invariants must satisfy $\mathit{lfp}(F) \Rightarrow P$, that is $\mathit{lfp}(F) \sqsubseteq P$ with lattice notations. We can obtain such a P by "strengthening" the terms of a chaotic iteration sequence.

DEFINITION 8.1 *A strengthened chaotic iteration* corresponding to the continuous operator $F \in L^n \to L^n$ and starting with a given vector X^0 such that $X^0 \sqsubseteq F(X^0)$ is a sequence X^k, k=0,1,... of vectors of L^n defined recursively by $X^{k+1} = \bar{\bar{F}}^k(X^k)$ where $\{\forall k \geq 0, \forall i \in [1,n],$

$\quad \underline{\text{if}}\ (i \in J^k\ \ \underline{\text{and}}\ F_i(X^k) \sqsubseteq X_i^k)\ \underline{\text{or}}\ (i \notin J^k)\ \underline{\text{then}}$

$\qquad \bar{\bar{F}}^k(X^k)_i = X_i^k$

$\quad \underline{\text{else}}$

$\qquad X_i^k \sqcup F_i(X^k) \sqsubseteq \bar{\bar{F}}^k(X^k)_i$

$\quad \underline{\text{fi}}\}$

and J^k, k=0,1,... is a sequence of subsets of {1,...,n} such that $\{\exists m \geq 0 : \{\forall i \in [1,n], \forall k \geq 0, \exists \ell \in [0,m[: i \in J^{k+\ell}\}\}$.

DEFINITION 8.2 A strengthened chaotic iteration is said to *stabilize after* s *steps* if and only if $\{\exists s \geq 0 : (X^{s+m}=X^s)\ \underline{\text{and}}\ (\forall r<s, X^{s+m} \neq X^s)\}$.

Notice that stabilization can always be enforced. Proof : take $X^1=\top$ (but this choice is of no practical interest).

THEOREM 8.3 The limit X^s of a chaotic iteration sequence $X^0,\ldots,X^k,\ldots$ which stabilizes after s steps is such that $\mathit{lfp}(F) \sqsubseteq X^s$.

Proof : Let us first prove that a chaotic iteration sequence is an increasing chain.
Basis : since $X^0 \sqsubseteq F(X^0)$, we have $X^0 \sqsubseteq F_{J^0}(X^0) \sqsubseteq \bar{\bar{F}}^0(X^0) = X^1$.
Induction step : suppose that $X^k \sqsubseteq X^{k+1}$. We have $X^{k+2} = \bar{\bar{F}}^{k+1}(X^{k+1})$. $\forall i \in [1,n]$, if $(i \in J^{k+1}\ \underline{\text{and}}\ F_i(X^{k+1}) \sqsubseteq X_i^{k+1})\ \underline{\text{or}}\ (i \notin J^{k+1})$ then $X_i^{k+2} = X_i^{k+1}$ else $X_i^{k+1} \sqsubseteq X_i^{k+1} \sqcup F_i(X^{k+1}) \sqsubseteq X_i^{k+2}$ proving that $X^{k+1} \sqsubseteq X^{k+2}$. Hence by recurrence on k : $\{\forall k \geq 0, X^k \sqsubseteq X^{k+1}\}$. According to the definition of a chaotic iteration which stabilizes we have $X^s = X^{s+m}$ and since $X^s \sqsubseteq X^{s+1} \sqsubseteq \ldots \sqsubseteq X^{s+m}$ this implies $X^s = X^{s+1} = \ldots = X^{s+m}$. Let us now prove that $F(X^s) \sqsubseteq X^s$. $\forall i \in [1,n]$, $\exists \ell \in [0,m[$ such that $i \in J^{s+\ell}$. Therefore if $F_i(X^{s+\ell}) \sqsubseteq X_i^{s+\ell}$ then $F_i(X^s) \sqsubseteq X_i^s$ since $X^s = X^{s+\ell}$ else $X_i^{s+\ell} \sqcup F_i(X^{s+\ell}) \sqsubseteq \bar{\bar{F}}^{s+\ell}(X^{s+\ell})_i = X_i^{s+\ell+1}$ which implies $X_i^s \sqcup F_i(X^s) \sqsubseteq X_i^s$ since $X^s = X^{s+\ell} = X^{s+\ell+1}$. Hence in both cases $\forall i \in [1,n]$, $F_i(X^s) \sqsubseteq X^s$ proving that $F(X^s) = X^s$. Also, according to Tarski's theorem $\mathit{lfp}(F)$ exists and is equal to $\sqcap S$ where $S = \{X \in L^n : F(X) \sqsubseteq X\}$ therefore $X^s \in S$ proving that $\mathit{lfp}(F) \sqsubseteq X^s$.
End of Proof.

In practice the definition of $\bar{\bar{F}}^k$ models the use of heuristics for generating invariants (e.g. Katz & Manna[1976]).

Example : Coming back to the program of §3.3, we have given the first terms of the corresponding approximation sequence at §3.4 :

$$P_2^0 = \underline{\text{false}}$$

$$P_2^1 = (\bar{x}\geq\bar{y})\ \underline{\text{and}}\ (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y})$$

$$\begin{aligned} P_2^2 = {} & [(\bar{x}\geq\bar{y})\ \underline{\text{and}}\ (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y})] \\ & \underline{\text{or}} \\ & [(\forall k\in[1,2]:\bar{x}\geq k\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-\bar{y})\ \underline{\text{and}}\ (y=\bar{y})] \end{aligned}$$

$$\begin{aligned} P_2^3 = {} & [(\bar{x}\geq\bar{y})\ \underline{\text{and}}\ (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y})] \\ & \underline{\text{or}} \\ & [(\forall k\in[1,2]:\bar{x}\geq k\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-\bar{y})\ \underline{\text{and}}\ (y=\bar{y})] \\ & \underline{\text{or}} \\ & [(\forall k\in[1,3]:\bar{x}\geq k\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-2\bar{y})\ \underline{\text{and}}\ (y=\bar{y})] \end{aligned}$$

An easy guess is that $P_2^3 \Rightarrow [\exists k\in[0,2] : (x=\bar{x}-k\bar{y})\ \underline{\text{and}}\ (y=\bar{y})]$. Therefore taking this assertion as a strengthened version of P_2^3 we compute P_2^4 :

$$\begin{aligned} P_3^3 & = \{\exists v : [\exists k\in[0,2] : (v=\bar{x}-k\bar{y})\ \underline{\text{and}}\ (y=\bar{y})\ \underline{\text{and}}\ (x=v-y)]\} \\ & = [\exists k\in[1,3] : (x=\bar{x}-k\bar{y})\ \underline{\text{and}}\ (y=\bar{y})] \end{aligned}$$

$$\begin{aligned} P_2^4 = {} & [(\bar{x}\geq\bar{y})\ \underline{\text{and}}\ (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y})] \\ & \underline{\text{or}} \\ & [\exists k\in[1,3]:(\bar{x}\geq(k+1)\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-k\bar{y})\ \underline{\text{and}}\ (y=\bar{y})] \end{aligned}$$

Since P_2^4 does not imply P_2^3 we strengthen P_2^4 to get :

$$P_2^4 = [\exists k\geq 0 : (x=\bar{x}-k\bar{y})\ \underline{\text{and}}\ (y=\bar{y})]$$

Iterating again we obtain :

$$\begin{aligned} P_2^5 = {} & [(\bar{x}\geq\bar{y})\ \underline{\text{and}}\ (x=\bar{x})\ \underline{\text{and}}\ (y=\bar{y})] \\ & \underline{\text{or}} \\ & [\exists k\geq 1:(\bar{x}\geq(k+1)\bar{y})\ \underline{\text{and}}\ (x=\bar{x}-k\bar{y})\ \underline{\text{and}}\ (y=\bar{y})] \end{aligned}$$

which implies P_2^4 so that P_2^4 can be chosen to be an approximate invariant at program point {2}.

End of Example.

(A post fixed point X^S of F (such that $F(X^S) \sqsubseteq X^S$) can further be improved to get a better approximation of *lfp*(F), see Cousot[1977a]).

9. CONCLUSION

The numerous techniques which are used for discovering properties of programs often appear to be rather different. We think that the fixpoint approach to the semantic analysis of programs (Cousot[1977a], Cousot[1977b]) provides the convenient framework for expressing the deep unity underlying these apparently unrelated techniques. By fixpoint approach in the semantic analysis of programs we refer to the whole of techniques for determining properties of programs which take as starting point the fact that properties of a program can be viewed as the least solution to a system of equations which is associated in a rather natural way with this program. The idea of solving these equations by means of chaotic successive approximations is so natural that it was first understood as a program execution on symbolic values. We think that even a semi-automatic resolution of the semantic equations is a terrific task which result can hardly be controlled by the programmer. Therefore an important idea for automatizing the resolution of these equations is the one of approximation. Compile time checks, global data flow analysis, program performance prediction or proofs of partial correctness are examples of approximate analyses of programs. It is our experience that the exact or total properties of a program can often be determined by addition of several approximate analyses of specific properties. The determination of a range of complementary logical properties of program should allow the analysis of program properties by successive approximations. Each of these specific properties should be simple enough to allow the automatic resolution of the corresponding equations. The spectrum of these properties should be wide enough to cover the total semantics of programs. Interaction with the programmer for guiding the choice and the order of determination of these specific properties might be a natural way for human intervention.

Acknowledgements : We were very lucky to have Mrs. C. Puech do the typing for us.

10. REFERENCES

Birkhoff[1967]. Birkhoff,G. *Lattice Theory.* 3rd ed., Colloquium Publications, Vol.XXV, AMS, Providence, R.I., 1967.

Cheatham & Townley[1976]. Cheatham, T.E., and Townley, J.A. *Symbolic evaluation of programs : a look at loop analysis.* Proc. of the 1976 ACM Symp. on Symbolic and Algebraic Computation, New York, Aug.1976.

Cohen & Katcoff[1976]. Cohen,J., and Katcoff,J. *Symbolic solution of finite difference equations.* Research Report, Physics Dept., Brandeis U., Waltham, Mass., July 1976.

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

Cousot[1977b]. Cousot,P., and Cousot,R. *Static determination of dynamic properties of recursive procedures*. IFIP W.G.2.2. Working Conference on Formal Description of Programming Concepts, St. Andrews, New Brunswick, Canada, Aug.1977.

Dijkstra[1976]. Dijkstra,E.W. *A discipline of programming*. Prentice Hall, New Jersey, 1976.

Elspas[1974]. Elspas,B. *The semiautomatic generation of inductive assertions for proving program correctness*. Research Rep. SRI, Menlo Park, Calif., July 1974.

Elspas et al.[1972]. Elspas,B., Green,M., Levitt,K., and Waldinger,R. *Research in interactive program proving techniques*. SRI, Menlo Park, Calif., May 1972.

Hantler & King[1976]. Hantler,S.L., and King,J. *An introduction ot proving the correctness of programs*. Computing Surveys 8, 2(Sept.1976), 331-353.

Katz & Manna[1976]. Katz,S., and Manna,Z. *Logical analysis of programs*. Comm. ACM 19, 4(April 1976), 188-206.

Kleene[1952]. Kleene,S.C. *Introduction to metamathematics*. North-Holland Pub. Co., Amsterdam, 1952, 348-349.

King[1969]. King,J. *A program verifier*. Ph.D. Th., Dept. of Computer Sci., Carnegie-Mellon U., Pittsburgh, Pa., 1969.

Sintzoff[1975]. Sintzoff,M. *Vérification d'assertions pour des fonctions utilisables comme valeurs et affectant des variables extérieures*. Proc. of Inter. Symp. on Proving and Improving Programs, Arcs et Senans, France, July 1975, 11-27.

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

Wegbreit[1975]. Wegbreit,B. *Mechanical program analysis*. Comm. ACM 18, 9(Sept.1975), 528-539.

SIGPLAN Notices
Vol. 12, No. 8 / August 1977

SIGART Newsletter
No. 64 / August 1977

Special Issue

Proceedings of the Symposium on Artificial Intelligence and Programming Languages

Association for Computing Machinery

The papers in this volume were presented at the ACM Symposium on Artificial Intelligence and Programming Languages, sponsored jointly by SIGART and SIGPLAN, and held at the University of Rochester, August 15-17, 1977.

The Program Committee wishes to thank all those who submitted papers for consideration.