AUTOMATIC SYNTHESIS OF OPTIMAL INARIANT 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 theoretic 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 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 \((L,\leq,\cup,\cap,\exists,\forall)\) a complete lattice with respect to the partial ordering \(\leq\). We use the symbols \(\cup,\cap,\exists,\forall\) for the finite and infinite lattice operations of join and meet. The infimum \(\bot\) and supremum \(\top\) of the lattice are defined by \(\bot=\forall L\) and \(\top=\exists L\), (Birkhoff[1967]).

A function \(\psi:\mathbb{L}^{\omega}\) of \(\mathbb{L}\) into \(\mathbb{L}\) is isotone (synonymously, order-preserving or monotone) if and only if \(\forall x,y\in\mathbb{L}, (x\leq y) \Rightarrow (\psi(x))\leq (\psi(y))\).

We define the limit of a chain \(x^x\in\mathbb{L}^{\omega}\) to be its least upper bound, \(\lim_{k=0} x^x = \bigcup_{k=0} x^x\).

A function \(\psi:\mathbb{L}^{\omega}\) is continuous if and only if for any chain \(x^x, k=0,1,\ldots\) we have \(\lim_{k=0} \psi(x^x) = \psi(\lim_{k=0} x^x)\). Note that a continuous function is necessarily isotone.
We denote by \( L^n \) the set of all vectors 
\( X = (X_1, \ldots, X_n) \) the components of which belong to \( L \). 
\((L^n, \leq_L, \cup, \cap, \exists_L)\) is a complete lattice with the usual "componentwise" definitions: 
\[ \{ (X \subseteq Y) \iff \{w \in \{1, \ldots, n\} : X_w \subseteq Y_w\}\} \]
\[ \{i \subseteq \{1, \ldots, n\}\}, \{\cup \subseteq \{1, \ldots, n\}\}, \{\cap \subseteq \{1, \ldots, n\}\}, \{\exists_L \subseteq \{1, \ldots, n\}\}, \cdots.\]

A function of several variable \( f : \mathbb{R}^n \rightarrow L \) is said to be isotope (continuous) in the variables jointly if and only if it is isotope (continuous) in the variables separately.

Hereafter we will consider a system of continuous equations with \( n \) variables of the form:
\[
\begin{cases}
X_1 = F_1(X_1, \ldots, X_n) \\
\vdots \\
X_n = F_n(X_1, \ldots, X_n)
\end{cases}
\]
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 \( L^n \rightarrow L^n \).

An element \( X \) of \( L^n \) is a fixpoint of \( f : L^n \rightarrow L^n \) if and only if \( f(X) = X \). The least fixpoint \( lfp(f) \) of \( f \) is such that:
\[ \{ (f(lfp(f)) \subseteq lfp(f)) \} \quad \text{and} \quad \{ \forall X \subseteq L^n, \{ f(X) = X \} \} \subseteq \{ lfp(f) \subseteq X \}. \]

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

In practice this theorem is not constructive since in general the set \( \{ X \subseteq L^n : F(X) \subseteq X \} \) of post-fixpoints of \( F \) is infinite and cannot be easily characterized. Yet, if \( F \) is continuous Kleene[1952] and Tarski[1955] suggest that the least fixpoint of \( F \) can be obtained as a limit of a sequence of successive approximations 
\[ X^0 = X, X^1 = F(X), \ldots, X^k = F(X^{k-1}), \ldots \text{that is } lfp(F) = \lim_{k \to \infty} F^k(X) \text{ where } F^k \]
denotes the \( k \)-fold composition of \( F \) with itself. This is nothing else than Jacobi's method of successive approximations:
\[
X^k = F(X^{k-1}, X^{k-2}, \ldots, X^1), \quad (k = 1, 2, \ldots, n) 
\]
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 \( L^n \rightarrow L^n \) defined by 
\[ F_J(X_1, \ldots, X_n) = (Y_1, \ldots, Y_n) \text{ where } Y_i = \begin{cases} F_i(X_i) & \text{if } i \in J \\ F_i(X_i) \text{ else } X_i \end{cases}. \]

**DEFINITION 2.1** A chaotic iteration corresponding to the operator \( F \) and starting with a given vector \( X^0 \) such that \( X^k \leq f(X^k) \leq lfp(F) \) is a sequence \( X^k \), \( k = 0, 1, \ldots \) of vectors of \( L^n \) defined recursively by 
\[ X^k = \bigcap_{j=k-1}^k F_j(X^{j-1}) \text{ where } J^k, k = 0, 1, \ldots \text{ is a sequence of subsets of } \{1, \ldots, n\} \text{ such that } \{ \exists \text{mao} : \{w \in \{1, \ldots, n\}, \exists k \in \mathbb{N}_0, \exists \in \mathbb{N} : i \leq i \} \} \]

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

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

**Proof:** Let us first remark that whenever \( X \subseteq f(X) \subseteq \) \( lfp(F) \) we have \( \forall k (1, \ldots, n), X \subseteq F_k(X) \subseteq F(X) \subseteq \) \( lfp(F) \). Indeed \( \forall k (1, \ldots, n), X \subseteq F_k(X) \subseteq F(X) \subseteq lfp(F) \) if \( i \in J \). Then \( X_i \subseteq F_i(X_i) \subseteq F_i(X) \subseteq lfp(F) \).

Since by hypothesis \( X^0 \leq F(X^0) \subseteq \) \( lfp(F) \) this implies \( X^0 \subseteq F_g(X^0) \subseteq f(X^0) \subseteq lfp(F) \). For the induction step let us assume that \( X^{k-1} \subseteq X^k \subseteq F(X^{k-1}) \subseteq \) \( lfp(F) \) for some \( k > 0 \). If \( i \in J^{k-1} \) then \( X^k_i \subseteq F^k_i(X^k) \subseteq lfp(F) \) since \( X^{k-1} \subseteq X^k \subseteq lfp(F) \).

Otherwise \( k \notin J^{k-1} \) and then \( X^k_i \subseteq F^k_i(X^k) \subseteq \) \( F^k_i(X^{k-1}) \) by induction hypothesis and \( F_i(X^{k-1}) \subseteq lfp(F) \). By isotomy in both cases \( \forall k (1, \ldots, n), X^k_i \subseteq F^k_i(X^k) \subseteq lfp(F) \) therefore \( X^k \subseteq F(X^k) \subseteq lfp(F) \) proving that \( X^k \subseteq X^{k+1} \subseteq F(X^{k+1}) \subseteq lfp(F) \). End of Proof.

**Lemma 2.2.2** \( \exists k \geq 0, X^k \leq X^{k+q} \).

**Proof:** The proof is by reductio ad absurdum. Let us suppose that \( \forall k (1, \ldots, n), \exists k > 0 \text{ such that } F(X^k) \text{ is not comparable with } X^{k+q}. \)

**case 1:** Suppose that \( \forall k (1, \ldots, n), \exists k > 0 \text{ such that } F(X^k) \text{ is not comparable with } X^{k+q}. \)

This must be true for \( q > 0 \) which contradicts lemma 2.2.1.

**case 2:** Suppose now that \( \forall k (1, \ldots, n), \exists k > 0 \text{ such that } F(X^k) \) is not comparable with \( X^{k+q} \).

This implies that for some component \( i \subseteq \{1, \ldots, n\} \) we have \( X^k_i \nless F(X^k) \), while for the other components the inequality is not necessarily strict. By definition of
chaotic iterations \( \forall m \geq 0 : \{ W_k, \forall k, \exists \epsilon(0, \infty) : i \in C_{m+k+1} \} \), therefore \( X_{k+1}^{k+2} = F_{k}(X_{k+1}^{k}) \). But lemma 2.2.1 implies by transitivity that \( X \subseteq X_{k+1}^{k+2} \) and by isotony \( F_{k}(X) \subseteq C_{m+k+1} \) which implies \( F_{k}(X) \subseteq X_{k+1}^{k+2+1} \).

Choosing \( q \leq k+1 \) we have \( \exists \) such that \( F_{k}(X) \subseteq X_{k+q} \) and also by hypothesis such that \( X_{k+q} = F_{k}(X) \), which is impossible. This contradiction proves the truth of lemma 2.2.2. End of Proof.

\[ \]

Proof of theorem 2.3: The proof that \( (\forall k : (k \leq n) \) and \( (k \neq k \) \Rightarrow \( \operatorname{Ilfp}(F) = X \)) \) is by reductio ad absurdum.

Indeed, suppose that \( \forall k : (k \geq n) \) or \( (k \neq k \) \Rightarrow \( \operatorname{Ilfp}(F) = X \)). Let \( \alpha \) be the least ordinal of cardinality strictly greater than \( n \). \( W_k \alpha \) we must have \( (k \neq k \) \Rightarrow \( \operatorname{Ilfp}(F) = X \)), that is \( \operatorname{Ilfp}(F) = X \) when choosing \( \alpha \) to be \( n \). Let us define \( \psi \in \alpha \wedge n \) by \( \psi(k) = X \).

We have \( (k \neq k \) \Rightarrow (k \neq k) \) exclusive or \( (k \neq k) \Rightarrow \) we have \( (\operatorname{Ilfp}(F) = X \) or \( (\operatorname{Ilfp}(F) = X \) implies \( \operatorname{Ilfp}(F) = X \).
Program entry point $j$:

\[ P_j(x, X^j) = \{ (x_i = X^j_i), i = 1..m \} \]

The respective initial values of the variables

3.4. OPTIMAL INVARIANTS

According to Tarski's theorem the above system of equations of the form $PsF(P)$ has a least solution
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:

$$
P^{opt}_1 = (x=\overline{y}) \text{ and } (y=\overline{y})
$$

$$
P^{opt}_2 = \{ \exists j:1:(\forall k \leq 1,j], \overline{z} \overline{y}) \text{ and } (x=\overline{y}-(j-1)\overline{y}) \text{ and } (y=\overline{y}) \}
$$

$$
P^{opt}_3 = \{ \exists j:1:(\forall k \leq 1,j], \overline{z} \overline{y}) \text{ and } (x=\overline{y}-j\overline{y}) \text{ and } (y=\overline{y}) \}
$$

$$
P^{opt}_4 = \{ \exists j:1:(\forall k \leq 1,j-1], \overline{z} \overline{y}) \text{ and } (x=\overline{y}-(j-1)\overline{y}) \text{ and } (y=\overline{y}) \}
$$

4. **Proofs of Total Correctness**

5. **Systems of Implications, Approximate Invariants and Proofs of Partial Correctness**

Most program verification methods use inequalities of the form $P \leq F(P)$ 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 $P^{opt} = \text{AND}(P : P \leq F(P))$, hence $(\forall P : P \leq F(P))$ we have $P^{opt} \Rightarrow P$.

A proof of partial correctness consists in proving that $((\forall \overline{x} : \phi(\overline{x})), \forall \overline{y}, \forall \overline{y} : P^{opt}(\overline{y}, \overline{x}) \Rightarrow \psi(\overline{y}, \overline{x}))$. 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 $P \leq F(P)$ then the proof that $(\forall \overline{y}, P(\overline{y}, \overline{x}) \Rightarrow \psi(\overline{y}, \overline{x}))$ constitutes a proof of partial cor-
satisfying \( P \models F(P) \). 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 \( P_i = \bigwedge_j p_j \) where each \( p_j \) is of the form \((Q_j \land (X_1 = E_{1,j}) \land \ldots \land (X_m = E_{m,j}))\). 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 conditions 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 \( E_{k,j} \). \( E_{k,j} \) is a formal expression depending on the initial values \( X \) of the variables on program entry. No \( X_k \) can appear as a free variable neither in \( Q_j \) nor in the \( E_{k,j} \). Slightly changing the notations we will call \( P_j \) a symbolic context and rewrite it as \( P_i = \bigwedge_j p_j \) where \( p_j = Q_j \land (X_1 = E_{1,j}) \land \ldots \land (X_m = E_{m,j}) \).

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

\[
\{Q_j \land (X_1 = E_{1,j}) \land \ldots \land (X_m = E_{m,j})\} : j \in \alpha
\]

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

\[
\{Q_j \land (X_1 = E_{1,j}) \land \ldots \land (X_m = E_{m,j})\} \land (X_k = E(X_1, \ldots, X_m))
\]

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

\[
\{Q_j \land \ldots \land (X_k = E(X_1, \ldots, X_m))\} : j \in \alpha
\]

We will denote this rule by the shorthand notation:

\[
\{p\} X_k := E(X_1, \ldots, X_m) \{P(X_k = E(P(X_1, \ldots, X_m)))\}
\]

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

\[
\{p\} X_k := E(X_1, \ldots, X_m) \{P(X_k = E(X_1, \ldots, X_m))\}
\]

The other rules can be deduced in the same way. In particular, the operation \( \cup \) describes the union of two symbolic contexts \( P = \{p_1, \ldots, p_n\} \) and \( Q = \{q_1, \ldots, q_m\} \) that is the set \( \{p_1, \ldots, p_n, q_1, \ldots, q_m\} \) 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:

\[
\begin{align*}
\text{loop:} & \quad (1) \text{ if } x = y \text{ then } \\
& \quad (2) \quad x := x - y; \\
& \quad (3) \quad \text{go to loop;} \\
& \quad (4) \quad \text{fi;}
\end{align*}
\]

we have the system of equations:

\[
\begin{align*}
P_0 &= \{<true, x, y>\} \\
P_1 &= P_0 \cup P_2 \\
P_2 &= P_1 \land (x = y) \\
P_3 &= P_2 \land (x \neq y)
\end{align*}
\]

6.4. SYMBOLIC EXECUTION TREE

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

Initialization:

\[
P^0_1 = \varnothing \quad \text{ (i=0..4)}
\]

Step 1:

\[
\begin{align*}
P^1_0 &= \{<true, x, y>\} \\
P^1_1 &= P^1_0 \cup P^0_3 = \{<true, x, y>\} \\
P^1_2 &= P^1_1 \land (x \neq y) = \{<x \neq y>, x, y\} \\
P^1_3 &= P^1_2 \land (x = y) = \{<x = y>, x, y\}
\end{align*}
\]

6
Step 2:

- \( P_0^2 = \langle \text{true}, x, y \rangle \)
- \( P_1^2 = \{ \langle \text{true}, x, y \rangle, \langle (R \geq y), x, y \rangle \} \)
- \( P_2^2 = \{ \langle (R \geq y), x, y \rangle, \langle (R \geq y) \text{ and } (R \geq 2y), x, y \rangle \} \)
- \( P_3^2 = \{ \langle (R < y), x, y \rangle, \langle (R < y) \text{ and } (R < 2y), x, y \rangle \} \)

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

\[
\begin{array}{c}
\langle \text{true}, x, y \rangle \\
\langle \text{true}, x, y \rangle \\
\langle (R \geq y), x, y \rangle \\
\langle (R \geq y) \text{ and } (R \geq 2y), x, y \rangle \\
\langle (R \geq y) \text{ and } (R \geq 2y), x, y \rangle \\
\langle (R < y), x, y \rangle \\
\langle (R < y) \text{ and } (R < 2y), x, y \rangle \\
\end{array}
\]

\[\{(P(x^0) \text{ or } P(x^m)) \text{ and } (\forall k, P(x^km) \Rightarrow P(x^{(k+1)m}))\} \Rightarrow P(\lim x^m)\]

(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 \( \{\forall a[1, (R \geq x)]/y], R \geq y \} \) at point 3 of program 6.3.

Base: For the single path of \( P_1 \) we have \( \{\forall a[1, 1], R \geq y \} \).

Induction step: We assume that at step \( k \) the symbo-
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:

\[ P_1 = P_0 \cup (P_1 \text{ and } x \Rightarrow y) \]

they are of the form:

\[ P_1 = f_1(P_0) \cup f_2(P_1) \]

The resolution by successive approximations was:

\[ P^1_1 = f_1(P_0) \cup f_2(\emptyset) = f_1(P_0) \]

\[ P^2_1 = f_1(P_0) \cup f_2 f_1(P_0) \]

\[ P^3_1 = f_1(P_0) \cup f_2 f_1(P_0) \cup f_2 f_1 f_1(P_0) \]

Since \( f_2 \) is distributive over \( \cup \), (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 OR and AND.

The general term of the approximation sequence is:

\[ P^1_1 = \bigcup_{i=0}^{k-1} f^i_2 f_1(P_0) \]

since:

\[ P^{k+1}_1 = f_1(P_0) \cup f_2 \bigcup_{i=0}^{k-1} f^i_2 f_1 f_1(P_0) \]

Passing to the limit we get:

\[ P_1 = \bigcup_{i=0}^{\infty} f^i_2 f_1 P_0 \]

This result can be obtained directly since the graph of dependence between \( P_1 \) and \( P_0 \):

\[ f_1 \rightarrow P_1 \]

\[ f_2 \rightarrow P_0 \]

can be considered as (in general a non-deterministic) loop which involves the AND gate.

\[ f^{i+1}_2 f^i_1(P_0) = f_2 \{ <p_{ij}, x_{ij}, y_{ij}> : j \in D_1 \} \]

\[ = \{ <p_{ij}, x_{ij}, y_{ij}, x_{ij}, y_{ij}> : j \in D_1 \} \]

First of all, we can determine the domain of \( j \) which indexes the possible paths. Since \( D_0 = \{ 1 \} \) and \( D_{i+1} = D_1 \) we have \( D_1 = \{ 1 \} \) (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):

\[ y_0 = y, \quad y_{i+1} = y_1 \]

\[ x_0 = x, \quad x_{i+1} = x_1, y_0 \]

\[ p_0 = \text{true}, \quad p_{i+1} = p_1 \text{ and } (x_i \geq y_i) \]

These recurrence relations can be solved directly (e.g. Cohen & Katcoff [1976]) yielding \( y_j = y_0, \quad x_j = x_0 - i y_0 \), and since \( p_0 = \text{true} \) and \( p_{i+1} = p_1 \) and \( x_0 (i+1) y_0 \) we get

\[ p_{i+1} = \text{AND}(x_i \geq y_0) \]

Finally the optimal loop invariant of program 6.3 is:

\[ I_1 = \bigcup_{i=0}^{\infty} \{ <p_{ij}, i, x_0 > : x_0 \geq y_0, y_0 > \} \]

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 & Townshend [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 can express $P_1^\omega$ by the following equivalent forms:

$$P_1^\omega = \bigcup_{i>0} (f_4 \circ f_3 \circ (f_6 \circ f_5 \circ f_1)^i \circ f_2)^i \circ f_1(P_0)$$

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

$$P_1^\omega = \bigcup_{k>0} (f_1(P_0) \cup f_4 \circ f_3 \circ (f_6 \circ f_5 \circ f_1)^k \circ f_2 \circ f_1(P_0))$$

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

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

$$P_1 = P_0 \cup (P_1 \text{ and } y \neq 0 \text{ and even}(P_1(y))) \cup (P_1 \text{ and } z \neq P_1(z))$$

$$P_1 = P_0 \cup (P_1 \text{ and } y \neq 0 \text{ and odd}(P_1(y))) \cup (P_1 \text{ and } z \neq P_1(z))$$

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

$$P_1 = P_0 \cup (P_1 \text{ and } y \neq 0)(x^2 + y^2 + z^2 + z \ast (\text{even}(y) + 1))$$

$$= P_0 \cup f(P_1)$$

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

- $\text{true} \lor v \lor (Q \lor v') = v \lor (Q \lor v')$
- $\text{false} \lor v \lor (Q \lor v') = (Q \lor v')$
- $Q \lor Q' = Q \text{ or } Q'$
- $Q \lor Q' = Q \land Q'$
- $\text{true}(Q \lor v) = (Q \lor v)'$
- $\text{true}(Q \lor v) = Q \lor v'$
- $Q \lor (Q' \lor v) = (Q \lor Q') \lor v$
- $f(v \lor (Q' \lor v)) = \text{true}(f(v \lor v))$
Finally the optimal output invariant of the program is:

\[ P_7 = P_1 \land (y = 0) \]

\[ = \bigcup_{i=0}^{\infty} < q_i \land (y_i = 0), x_i, y_i, z_i > \]

\[ = \bigcup \left\{ \begin{array}{l}
q(b=0, x=a, y=b, z=1) \\
q((\exists i > 0 : (b \cdot 2^i - 1 \neq 0) \land (b \cdot 2^i = 0)),
\quad x = a^{2^i} ,
\quad y = b^{2^i} ,
\quad z = \prod_{j=0}^{i} \left( \text{even}(b \cdot 2^j) \rightarrow 1 \lor \text{odd}(b \cdot 2^j) \rightarrow a^{2^j} \right) \end{array} \right. \]

Since the path condition at the haltpoint {7} : (b=0) or (\exists i > 0 : (b \cdot 2^i - 1 \neq 0) and (b \cdot 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 \( b = a \cdot 2^i \).

\[ \frac{\mathbb{P}(X^k) \_1 = x^k \_1}{\text{else}} \]

\[ x^k \_1 \cup \exists \_1(X^k) \in \mathbb{P}(X^k) \_1 \]

And \( J^k \), \( k = 0, 1, \ldots \) is a sequence of subsets of \( \{1, \ldots, n\} \) such that \( \{ \exists i \geq 0 : (\forall i \in [1, n], \forall k \geq 0, \exists l \in [0, m] : i \in J^k \_l) \} \).

**DEFINITION 8.2** A strengthened chaotic iteration is said to **stabilize after s steps** if and only if \( \{ \exists s \geq 0 : (x^{s+m} = x^s) \land (\forall s, x^{s+m} = x^s) \} \).

Notice that stabilization can always be enforced. Proof: take \( x^i = \dagger \) (but this choice is of no practical interest).

**THEOREM 8.3** The limit \( x^S \) of a chaotic iteration sequence \( x^1, x^2, \ldots \) which stabilizes after s steps is such that \( \mu_{p}(F) \subseteq x^S \).
approximation sequence at §3.4 :
\[ p_2^g = \text{false} \]
\[ p_2^1 = (x \geq y) \land (x = R) \land (y = \bar{y}) \]
\[ p_2^2 = (\forall k \in \{1, 2\}, (x \geq k y) \land (x \in [0, 1] \land (y = \bar{y})) \]
\[ p_2^3 = (\forall k \in \{1, 2\}, (x \geq k y) \land (x \in [0, 1] \land (y = \bar{y})) \]
\[ p_2^4 = (\forall k \in \{1, 2\}, (x \geq k y) \land (x \in [0, 1] \land (y = \bar{y})) \]

An easy guess is that \( p_2^5 \Rightarrow [\exists k \in [0, 2] : (x = R - k y) \land (y = \bar{y})] \). Therefore taking this assertion as a strengthened version of \( p_2^6 \) we compute \( p_2^6 \)

\[ p_2^6 = (\forall k \in \{1, 2\}, (x = R - k y) \land (y = \bar{y}) \land (x = v - y)) \]
\[ p_2^7 = (\forall k \in \{1, 2\}, (x = R - k y) \land (y = \bar{y}) \land (x = v - y)) \]
\[ p_2^8 = (\forall k \in \{1, 2\}, (x = R - k y) \land (y = \bar{y}) \land (x = v - y)) \]

Since \( p_2^8 \) does not imply \( p_2^9 \) we strengthen \( p_2^9 \) to get :
\[ p_2^9 = [\exists k \in [0, 2] : (x = R - k y) \land (y = \bar{y})] \]

Iterating again we obtain :
\[ p_2^{10} = (\forall k \in [0, 1], (x \geq k y) \land (y = \bar{y}) \land (x = v - y)) \]

which implies \( p_2^{11} \) so that \( p_2^{11} \) can be chosen to be an approximate invariant at program point \( 2 \).

End of Example.

\( p_2^{12} \) of \( F \) such that \( F(x^{12}) = x^{12} \) can further be improved to get a better approximation of \( I_\mathcal{F}p(F) \), see Cousot[1977a]).

9. CONCLUSION

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 automating the resolution of these equations is the one of approximation. Compile time checks, global dataflow 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

Cousot[1977b]. Cousot, P., and Cousot, R. Static de-


Special Issue

Proceedings of the Symposium on
Artificial Intelligence and Programming Languages
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.