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 proofs of partial correctness (section 5). In section 6 we...
We denote by $L^n$ the set of all vectors $X=(X_1,\ldots,X_n)$ the components of which belong to $L$. The order $L^n$ is a complete lattice with the usual "componentwise" definitions: if $Y=(Y_1,\ldots,Y_n)$, $(x_i\in\{0,1\})$, $(X\cup Y = (X_1\cup Y_1,\ldots,X_n\cup Y_n))$, etc.

A function of several variables $f: L^n \to L$ 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:

\[
\begin{align*}
X_1 &= F_1(X_1,\ldots,X_n) \\
& \vdots \\
X_n &= F_n(X_1,\ldots,X_n)
\end{align*}
\]

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 $L^n\to L^n$.

If $X=F^n(X)$ is a fixpoint of $F$ then $X=F^n(X)\in L^n$.

If and only if $F(X)=X$. The least fixpoint $\text{lf}(F)$ of $F$ is such that:

\[
\{ (F(\text{lf}(F))=\text{lf}(F)) \} \text{ and } (\forall X \in L^n, (F(X)=X) \Rightarrow (\text{lf}(F) \subseteq X)).
\]

**Theorem (Tarski[55])** Any monotone map $F$ of a complete lattice $L^n$ into itself has a least fixpoint defined by: $\text{lf}(F) = \bigcap \{ X \in L^n : F(X) \subseteq X \}$.

In practice this theorem is not constructive since in general the set $\{ X \in L^n : F(X) \subseteq X \}$ of post-fixpoints of $F$ is infinite and cannot be easily characterized. Yet, when $F$ is continuous Kleene[52] and Tarski[55] suggest the least fixpoint of $F$ can be obtained as the limit of a sequence of successive approximations $X^0=\bot, X^1=F(X^0), X^2=F(X^1), \ldots$ that is $\text{lf}(F) = \lim_{k \to \infty} F(X^k)$ where $F^k$ denotes the $k$-fold composition of $F$ with itself.

This is nothing else than Jacobi’s method of successive approximations:

\[
\begin{align*}
X^1 &= F_1(X^0, X^0, \ldots, X^0) \\
& \vdots \\
X^n &= F_n(X^0, X^0, \ldots, X^0)
\end{align*}
\]

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 $\forall i \in J \cap n$ we have $Y_i=I_{ij} I_{ij} \text{ then } F_{I_{ij}}(X_1,\ldots,X_n) \neq X_i$.

**Definition 2.1** A chaotic iteration corresponding to the operator $F$ and starting with a given vector $X^0$ such that $X^0=F(X^0)\subseteq \text{lf}(F)$ is a sequence $X^k$, $k=0,1,\ldots$ of vectors of $L^n$ defined recursively by $X^{k+1}=F_{J_k}(X^k)$ where $J_k$, $k=0,1,\ldots$ is a sequence of subsets of $\{1,\ldots,n\}$ such that $(\exists m_0 : (\forall i \in J_{m_0}, j \in \{0,1,\ldots,m_0\} \Rightarrow I_{ij})$.

**Theorem 2.2** The limit $X^\omega$ of a chaotic iteration $X^0,\ldots,X^k,\ldots$ is equal to $\text{lf}(F)$.

**Lemma 2.1** $\{ \forall k \geq 0, X^k \subseteq X^{k+1} \subseteq F(X^k) \subseteq \text{lf}(F) \}.$

**Proof** - Let us first remark that whenever $X^k=F(X^k)$.

If $F(X)=X$. The least fixpoint $\text{lf}(F)$ of $F$ is such that:

\[
\{ (F(\text{lf}(F))=\text{lf}(F)) \} \text{ and } (\forall X \in L^n, (F(X)=X) \Rightarrow (\text{lf}(F) \subseteq X)).
\]

Indeed $\forall i \in \{0,1,\ldots,n\}$, $X_i \subseteq F_i(X)$ therefore if $i \in J$ then $X_i \subseteq F_i(X)\subseteq F_J(X)$ otherwise $X_i \subset F_i(X)\subseteq F_F(X)$.

Since by hypothesis $X^0=F(X^0)\subseteq \text{lf}(F)$ this implies $X^0\subseteq F_J(X^0)=F_J(X)$ otherwise $X_i \subset F_i(X)\subseteq F_J(X)$.

For the induction step let us assume that $X^{k-1} \subseteq F_{J_k}(X^{k-1}) \subseteq \text{lf}(F)$ for some $k>0$. If $i \in J_k$ then $X_i^{k-1} \subseteq F_i(X^{k-1}) \subseteq F_{J_k}(X^{k-1}) \subseteq \text{lf}(F)$ and $F_{J_k}(X^{k-1})$ is isotone otherwise $F_{J_k}(X^{k-1}) \subseteq \text{lf}(F)$ by isotony. In both cases $\forall i \in \{0,1,\ldots,n\}$, $X_i \subseteq F_i(X^{k-1}) \subseteq \text{lf}(F)$ and therefore $X^k=F(X^k)\subseteq \text{lf}(F)$ proving that $X^k=F_{J_k}(X^{k-1}) \subseteq \text{lf}(F)$.

End of Proof.

**Lemma 2.2** $\{ \exists q \in \{0,\ldots,m\} : \forall k \geq 0, F(X^k) \subseteq X^{k+q} \}.$

**Proof** - The proof is by reductio ad absurdum. Let us suppose that $\{ \exists q \in \{0,\ldots,m\}, \exists k \geq 0: \not(F(X^k) \subseteq X^{k+q}) \}$

$\Rightarrow \{ \forall k \in \{0,\ldots,m\}, \exists k \geq 0 : (X^{k+q}) \not(F(X^k) \subseteq X^{k+q}) \}$

By definition of the strict inequality $\subseteq$ we have $X^{k+q} \subseteq F(X^k)$ and $X^{k+q} \not(F(X^k)$.

Imply that for some component $i \in \{0,1,\ldots,m\}$ we have $X_i^{k+q} \subseteq F_i(X^k)$, while for the other components the inequality is not necessarily strict. By definition of
chaotic iterations \( \forall k \geq 0 : \{ V_1, V_k, \exists \iota \in (0, m[ : \ i \in k^0) \} \), therefore \( X^{k+1} = F_1(X^{k+1}) \). But lemma 2.2.1 implies by transitivity that \( X^k \subseteq X^{k+1} \) thus by isotony \( F_1(X^k) \subseteq F_1(X^{k+1}) \) which implies \( F_1(X^k) \subseteq X^{k+1} \).

Choosing \( q = k+1 \) we have \( \exists k \) such that \( F_1(X^k) \subseteq X^{k+q} \) and also by hypothesis such that \( X^{k+q} = F_1(X^k) \), which is impossible. This contradiction proves the truth of lemma 2.2.2. End of Proof.

Proof of theorem 2.3: The proof that \( (k_1 : (k_1 \triangleright k_2)) \) and \( (k_2 \triangleright k_3 \Rightarrow \text{lf}(x_1 \triangleright x_2)) \) is by reductio ad absurdum. Indeed, suppose that \( (k_1 : (k_1 \triangleright k_2)) \) or \( (k_2 \triangleright k_3 \Rightarrow \text{lf}(x_1 \triangleright x_2)) \). Let \( \alpha \) be the least ordinal of cardinality strictly greater than \( \omega \). \( \forall k < \alpha \) we must have \( (k_2 \triangleright k_3 \Rightarrow \text{lf}(x_1 \triangleright x_2)) \), that is \( \text{lf}(x_1 \triangleright x_2) \) when choosing \( \lambda \) to be \( k_2 \). Let us define \( \psi \in \alpha \triangleright \omega \) by \( \psi(k) = x_1 \triangleright x_2 \).

\( \psi_1 \), \( \psi_2 \), \( \psi_3 \) are such that \( k_1 \neq k_2 \) (with eventually \( k_1 = \alpha \) exclusive or \( k_2 = \alpha \)) we have \( \text{lf}(x_1 \triangleright x_2) \) or \( \text{lf}(x_1 \triangleright x_2) \) hence lemma 2.3.1 imply that \( x_1 \triangleright x_2 \) proving that \( \psi \) is a one to one correspondence of \( \alpha \) into \( \omega \). Therefore \( \alpha \) is of cardinality less or equal to that of \( \omega \) which is the desired contradiction. End of Proof.
Program entry point $j$:

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

The respective initial values of the variables $X=(X_1, ..., X_m)$ are the symbols $\overline{X}=(\overline{X}_1^1, ..., \overline{X}_m^1)$ (We may eventually have $(X_i = \mathbb{N})$ when the variable $X_i$ is not initialized).

3.4. OPTIMAL INVARIANTS

According to Tarski's theorem the above system of equations of the form $P=F(P)$ has a least solution $P_\text{opt}$ (least for ordering $\leq$ that is $\Rightarrow$). We call $P_\text{opt}$ the set of optimal invariants since they imply the existence of the system of equations (for...
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 $P^{opt} = \lim_{i \to \infty} P^i$.

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

Most program verification methods use inequalities of the form $P \Rightarrow F(P)$ 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
satisfying \( P \Rightarrow F(P) \). 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 \( P_i = \text{OR} \left( p_j \right) \) where each \( p_j \) is of the form \( (Q_j \land (X_1 = E_{1j}) \land \ldots \land (X_m = E_{mj})) \). 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 \( X \) 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 \( p_j = (Q_j \land (X_1 = E_{1j}) \land \ldots \land (X_m = E_{mj})) \).

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

\[
\{ (Q_j \land (X_1 = E_{1j}) \land \ldots \land (X_m = E_{mj})) \}_{j \in \Delta}
\]

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

\[
\{ \text{OR} (Q_j \land (X_1 = E_{1j}) \land \ldots \land (X_m = E_{mj})) \}_{j \in \Delta}
\]

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

\[
\{ \text{OR} (Q_j \land (X_1 = E_{1j}) \land \ldots \land (X_m = E_{mj})) \}_{j \in \Delta}
\]

We will denote this rule by the shorthand notation:

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

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

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

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

\[
\{ 0 \}
\]

\[
\text{loop:} \{ 1 \} \text{ if } x \leq y \text{ then } \{ 2 \} x := x - y \{ 3 \} \text{ go to loop; } \{ 4 \} \text{ fi;}
\]

we have the system of equations:

\[
\begin{align*}
\{ P_0 \} & = \{ \text{true,} R, Y \} \\
\{ P_1 \} & = P_0 \cup P_3 \\
\{ P_2 \} & = P_1 \land (x \geq y) \\
\{ P_3 \} & = P_1 \land (x \leq y)
\end{align*}
\]

6.4. SYMBOLIC EXECUTION TREE

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

Initialization:

\[
\begin{align*}
p_0^{(i)} & = \varnothing & (i = 0..4) \\
p_1^{(0)} & = \{ \text{true,} R, P \} \\
p_1^{(1)} & = p_1^{(0)} \cup p_3^{(0)} = \{ \text{true,} R, Y \} \\
p_1^{(2)} & = p_1^{(1)} \land (x \geq y) = \{ R \geq y, R, Y \} \\
p_1^{(3)} & = p_1^{(2)} \land (x \leq y) = \{ R \leq y, R, Y \}
\end{align*}
\]
Step 2:

\[ p^2_0 = \{ \langle \text{true}, x, y \rangle \} \]
\[ p^2_1 = \{ \langle \text{true}, x, y \rangle, \langle (x \geq 2y), x = y, y \rangle \} \]
\[ p^2_2 = \{ \langle (x \geq 2y), x \geq y, y \rangle, \langle (x \leq 2y), x = y \rangle \} \]
\[ p^2_3 = \{ \langle (x \leq 2y), x = y \rangle, \langle (x < 2y), x = y \rangle \} \]

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

We have represented the symbolic context \( p^1_1 \) 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 \( \cup \) 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 preview,

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

\[ (P(\lim x^m)) \]

(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 \( \{ \text{Vas}[1, (\text{R} \cdot \text{x})/y_1], \text{R} \cdot \text{y}_2] \) at point 3 of program 6.3.

Base: For the single path of \( p^1_1 \) we have \( \{ \text{Vas}[1, 1], \text{R} \cdot \text{y}_2] \).

Induction step: We assume that at step \( k \) the symbolic context \( p^k_1 \) is equal to \( \{ p_{1, x_1}^{i_k} \cdot y_1^{i_k} : i \in D \} \) with the induction hypothesis \( \{ \text{Vas}[1, (\text{R} \cdot x_i)/y_1], \text{R} \cdot y_2] \} \). The equations 6.3 allow the computation of \( p^{k+1}_1 \):

\[ p^{k+1}_1 = \{ \langle \text{true}, x_1^{i_k}, y_1^{i_k} : i \in D \} \]
\[ p^{k+1}_2 = \{ \langle p_{1}^{i_k}, p_{1}^{i_k} \cdot y_1^{i_k} : i \in D \} \]
\[ p^{k+1}_3 = \{ \langle (x \geq 2y), x \geq y, y \rangle, \langle (x \leq 2y), x = y \rangle \} \]

We must show that the hypothesis holds for all paths of \( p^{k+1}_1 \). This is trivial for the path \( \langle (x \geq 2y), x \geq y, y \rangle \). Otherwise we must show that \( \{ \text{Vas}[1, (\text{R} \cdot x_i)/y_1], \text{R} \cdot y_2] \} \). According to the induction hypothesis, this condition is true \( \{ \text{Vas}[1, (\text{R} \cdot x_i)/y_1] \). Finally for \( i = (\text{R} \cdot x_i)/y_1^{i+1} \), the path condition \( x_1^{i+1} \cdot y_1^{i+1} \) implies \( \text{R} \cdot y_2] \).

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
We can express $P_1^\infty$ by the following equivalent forms:

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

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

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

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

Although these two forms of $P_1^\infty$ 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]): ($\div$ is integer division with truncation)

```
{0}
  z:=1;
{1} loop: {2} if y=0 then {3} if odd(y) then {4} fi; {5} (y,x):=(y\div 2,x^2); {6} go to loop; {7} fi;
```

Following:

$$P_1 = P_0 \cup (P_1 \circ \text{and}(P_1(y) \circ \text{even}(P_1(y)))(y+P_1(y)\div 2,x+P_6(x),z+P_1(z)))$$

$$\cup (P_1 \circ \text{and}(P_1(y) \circ \text{odd}(P_1(y)))(y+P_1(y)\div 2,x+(P_6(x))^2,z+P_1(z)\circ P_1(x)))$$

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

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

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

- $(\text{true} \circ v \cup Q \circ v') = v \cup (Q \circ v')$
- $(\text{false} \circ v \cup Q \circ v') = (Q \circ v')$
- $Q \cup Q' = Q \circ Q'$
- $Q \circ Q' = Q \circ Q'$
- $\text{nor}(Q \circ v) = (Q \circ v')$
- $\text{not}(Q \circ v) = \text{not}(Q \circ v')$
- $Q \circ (Q' \circ v) = (Q \circ v')$
- $f(v_1,\ldots,v_n) = \bigcup f(v_1,\ldots,v_n)$
- $f(v_1,\ldots,(Q \circ w),\ldots,v_n) = (Q \circ f(v_1,\ldots,w,\ldots,v_n))$
- etc.

The general form for the solution for the equation defining $P_1$ is $P_1 = \bigcup f_i(P_0)$. For determining $f_i(P_0)$ we use the difference equations:

$$f_0(P_0) = \{\text{true},a,b,1\}$$

$$f_1(P_0) = \{\text{false},x_0,y_0,z_0\}$$

$$f_{i+1}(P_0) = f_i(\{q_i \circ x_i,y_i,z_i\})$$

$$= \{q_i \circ (y_i=0),x_i \div 2,y_i \div 2, (\text{even}(y_i) \circ 1 \cup \text{odd}(y_i) \circ x_i)\}$$

$$= \{q_i \circ x_{i+1},y_{i+1},z_{i+1}\}$$

The formal resolution proceeds as follows:

$$x_0 = a \Rightarrow x_i = a^i$$

$$x_{i+1} = x_i^2$$
Finally the optimal output invariant of the program is:
\[
P_y = \{ x \mid (y = 0) \}
\]
= \{ x \mid (y = 0) \}
\]
= \{ x \mid (y = 0) \}
\]
Since the path condition at the halting point \( (7) \):
\( b = 0 \) or \( \neg q \) or \( x = b \) or \( b^2 = 0 \)
is true for any input value \( b \) of \( y \), the program always terminates.

It remains to show that \( P_y = \{ x \mid (y = 0) \} \) 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:
\[
b = \sum_{i=0}^{l-1} b_i 2^i \text{ with } b_i \in \{0, 1\}
\]
Evaluating \( a \delta b \) using the property that:
\[
a \delta b = a \oplus b \quad \text{if } a \neq b
\]
we get:
\[
a \delta b = \sum_{i=0}^{l-1} (a_i \oplus b_i) 2^i
\]
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 \( I_P(F) \subseteq P \), that is \( I_P(F) \subseteq \mathbb{P} \) 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 \( F : \mathbb{R}^n \rightarrow \mathbb{R}^n \)

and starting with a given vector \( X_0 \) such that \( X^k \in F(X^k) \) is a sequence \( X^k, k = 0, 1, \ldots \) of vectors of \( \mathbb{R}^n \) defined recursively by \( X^{k+1} = F(X^k) \) where \( \{ X_k \} \subseteq \mathbb{R}^n \).

**Theorem 8.3** The limit \( X^\infty \) of a chaotic iteration sequence \( X^k, k = 0, 1, \ldots \) which stabilizes after \( s \) steps is such that \( I_P(F) \subseteq X^\infty \).

Proof: Let us first prove that a chaotic iteration sequence is an increasing chain.

**Definition 8.2** A strengthened chaotic iteration is said to stabilize after \( s \) steps if and only if \( \{ s \geq s_0 : (X^{s+1} = X^s) \} \) and \( \{ \forall x : S \subseteq X^s \} \)

Notice that stabilization can always be enforced.

Proof: Let us first prove that a chaotic iteration sequence is an increasing chain.

**Definition 8.1** A strengthened chaotic iteration corresponding to the continuous operator \( F : \mathbb{R}^n \rightarrow \mathbb{R}^n \)

and starting with a given vector \( X_0 \) such that \( X^k \in F(X^k) \) is a sequence \( X^k, k = 0, 1, \ldots \) of vectors of \( \mathbb{R}^n \) defined recursively by \( X^{k+1} = F(X^k) \) where \( \{ X_k \} \subseteq \mathbb{R}^n \).

**Theorem 8.3** The limit \( X^\infty \) of a chaotic iteration sequence \( X^k, k = 0, 1, \ldots \) which stabilizes after \( s \) steps is such that \( I_P(F) \subseteq X^\infty \).

Proof: Let us first prove that a chaotic iteration sequence is an increasing chain.

**Definition 8.2** A strengthened chaotic iteration is said to stabilize after \( s \) steps if and only if \( \{ s \geq s_0 : (X^{s+1} = X^s) \} \) and \( \{ \forall x : S \subseteq X^s \} \)

Notice that stabilization can always be enforced.

Proof: Let us first prove that a chaotic iteration sequence is an increasing chain.

**Definition 8.1** A strengthened chaotic iteration corresponding to the continuous operator \( F : \mathbb{R}^n \rightarrow \mathbb{R}^n \)

and starting with a given vector \( X_0 \) such that \( X^k \in F(X^k) \) is a sequence \( X^k, k = 0, 1, \ldots \) of vectors of \( \mathbb{R}^n \) defined recursively by \( X^{k+1} = F(X^k) \) where \( \{ X_k \} \subseteq \mathbb{R}^n \).

**Theorem 8.3** The limit \( X^\infty \) of a chaotic iteration sequence \( X^k, k = 0, 1, \ldots \) which stabilizes after \( s \) steps is such that \( I_P(F) \subseteq X^\infty \).

Proof: Let us first prove that a chaotic iteration sequence is an increasing chain.

**Definition 8.2** A strengthened chaotic iteration is said to stabilize after \( s \) steps if and only if \( \{ s \geq s_0 : (X^{s+1} = X^s) \} \) and \( \{ \forall x : S \subseteq X^s \} \)

Notice that stabilization can always be enforced.

Proof: Let us first prove that a chaotic iteration sequence is an increasing chain.

**Definition 8.1** A strengthened chaotic iteration corresponding to the continuous operator \( F : \mathbb{R}^n \rightarrow \mathbb{R}^n \)

and starting with a given vector \( X_0 \) such that \( X^k \in F(X^k) \) is a sequence \( X^k, k = 0, 1, \ldots \) of vectors of \( \mathbb{R}^n \) defined recursively by \( X^{k+1} = F(X^k) \) where \( \{ X_k \} \subseteq \mathbb{R}^n \).

**Theorem 8.3** The limit \( X^\infty \) of a chaotic iteration sequence \( X^k, k = 0, 1, \ldots \) which stabilizes after \( s \) steps is such that \( I_P(F) \subseteq X^\infty \).

Proof: Let us first prove that a chaotic iteration sequence is an increasing chain.

**Definition 8.2** A strengthened chaotic iteration is said to stabilize after \( s \) steps if and only if \( \{ s \geq s_0 : (X^{s+1} = X^s) \} \) and \( \{ \forall x : S \subseteq X^s \} \)

Notice that stabilization can always be enforced.
approximation sequence at §3.4:

\[ p_2^0 = \text{false} \]

\[ p_2^1 = (x=y) \land (x=x) \land (y=y) \]

\[ p_2^2 = [(x=y) \land (x=x) \land (y=y)] \]

\[ \lor \]

\[ [(\forall k \in [1, 2]: x=k \land y) \land (x=x-y) \land (y=y)] \]

\[ p_2^3 = [(x=y) \land (x=x) \land (y=y)] \]

\[ \lor \]

\[ [(\forall k \in [1, 3]: x=k \land y) \land (x=x-y) \land (y=y)] \]

An easy guess is that \( p_2^3 \Rightarrow [\exists k \in [0, 2]: (x=x-k \land y) \land (y=y)] \). Therefore taking this assertion as a strengthened version of \( p_2^3 \) we compute \( p_2^4 \):

\[ p_2^4 = [(x=y) \land (x=x) \land (y=y)] \]

\[ \lor \]

\[ [(\forall k \in [1, 3]: x=k \land y) \land (x=x-y) \land (y=y)] \]

Since \( p_2^4 \) does not imply \( p_2^3 \) we strengthen \( p_2^4 \) to get:

\[ p_2^5 = [\exists k \geq 0: (x=x-k \land y)] \]

Iterating again we obtain:

\[ p_2^6 = [(x=y) \land (x=x) \land (y=y)] \]

\[ \lor \]

\[ [(\forall k \in [1, 3]: (x=(k+1) \land y) \land (x=x-k \land y) \land (y=y)] \]

which implies \( p_2^6 \) so that \( p_2^6 \) can be chosen to be an approximate invariant at program point \( 2 \).

End of Example.

Acknowledgments: We were very lucky to have...


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.