Formal Descriptions of Programming Concepts, E.J. Neuhold (ed.)
North-Holland Publishing Company, (1978)

STATIC DETERMINATION OF DYNAMIC PROPERTIES OF RECURSIVE PROCEDURES

Patrick Cousot* and Radhia Cousot**

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

1. INTRODUCTION

We present a general technique for determining properties of recursive procedures. For example, a mechanized analysis of the procedure reverse can show that whenever L is a non-empty linked linear list then reverse(L) is a non-empty linked linear list which shares no elements with L. This information about reverse approximates the fact that reverse(L) is a reversed copy of L.

In section 2, we introduce $\sqcup$-topological lattices that is complete lattices endowed with a $\sqcup$-topology. The continuity of functions is characterized in this topology and fixed point theorems are recalled in this context.

The semantics of recursive procedures is defined by a predicate transformer associated with the procedure. This predicate transformer is the least fixed point of a system of functional equations (§3.2) associated with the procedure by a syntactic algorithm (§3.1).

In section 4, we study the mechanized discovery of approximate properties of recursive procedures. The notion of approximation of a semantic property is introduced by means of a closure operator on the $\sqcup$-topological lattice of predicates. Several characterizations of closure operations are given which can be used in practice to define the approximate properties of interest (§4.1.1). The lattice of closure operators induces a hierarchy of program analyses according to their fineness. Combinations of different analyses of programs are studied (§4.1.2). A closure operator defined on the semantic $\sqcup$-topological space induces a relative

---

* Attaché de Recherche au C.N.R.S., Laboratoire Associé n° 7.

** This work was supported by I.R.I.A.- S.E.S.O.R.I. under grant 76-160.

topology on the complete lattice of approximate properties, so that the space of approximate properties is a $\sqcup$-topological lattice (§4.1.3). Then functions and functionals on the space of semantic properties can be expressed in the space of approximate properties (§4.1.4, §4.1.5). In order to represent the space of approximate properties in a machine we use an homeomorphic space the elements of which can be represented inside a computer (§4.1.6). The systematic correspondence between semantic and approximate properties of programs, allows the association of a system of approximate functional equations with a recursive procedure (§4.1.7). Its mechanical resolution by successive approximations determines an approximate predicate transformer which is a partial representation of the meaning of the procedure (§4.2). In practice chaotic iterations are used to construct this solution when the space of approximate properties is finite (§4.2.1) but when infinite strengthened chaotic iterations must be used to accelerate the convergence (§4.2.2).

Throughout the paper various practical examples are given.

2. $\sqcup$ - TOPOLOGICAL LATTICES, CONTINUOUS FUNCTIONS AND FIXED POINT THEOREM

The mathematical preliminaries which are developed in this section will be used throughout the paper. Topologization of lattices is well-known, but our hypothesis beeing weaker than classical ones (Birkhoff[1967], Frink[1942], Scott[1972] etc.) we carefully specify which topology we want to be associated with a complete lattice.

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

An *up-directed set* is a poset (partially ordered set) in which any two elements, and hence any finite subset, has an upper bound in the set.

We define a *net* $[\![x_\delta:\delta\in\Delta]\!]$ of points of a space S as a function on the directed set Δ of indices into S.

Let $[\![x_\delta:\delta\in\Delta]\!]$ be a net on the up-directed set $(\Delta,\leq)$ with values in the poset $(L,\sqsubseteq)$. We say that $[\![x_\delta:\delta\in\Delta]\!]$ is a *monotone increasing net* if and only if whenever $\gamma,\delta\in\Delta$ and $\gamma\leq\delta$ then $x_\gamma\sqsubseteq x_\delta$. Hereafter we will only consider monotone increasing nets.

A *dual ideal* of L is a non-void subset J of the lattice L with the properties : (i) $\{\forall a\in J,\ \forall x\in L,\ \{a\sqsubseteq x\}\Rightarrow\{x\in J\}\}$ and (ii) $\{\forall a\in J,\ \{b\in J\}\Rightarrow\{a\sqcap b\}\in J\}$.

Let $\mathbb{I}d(L)$ denote the set of all dual ideals of L, the *augmented set of dual ideals* of L is $\mathbb{I}d_0(L)=\mathbb{I}d(L)\cup\{\emptyset\}$.

LEMMA 2.1 If $I,J\in\mathbb{I}d_0(L)$ then $I\cap J=\{a\sqcup b : (a\in I)\ \underline{\text{and}}\ (b\in J)\}$ and $(I\cap J)\in\mathbb{I}d_0(L)$.

Proof : Since $a\sqsubseteq a\sqcup b$ and $b\sqsubseteq a\sqcup b$ any such $a\sqcup b$ is in I and in J, hence it is in their intersection $I\cap J$. Conversely, $\forall c\in(I\cap J)$ then $c=c\sqcup c$ where $c\in I$ and $c\in J$. Let us prove now that $(I\cap J)\in\mathbb{I}d_0(L)$. $\forall c\in(I\cap J)$, $\exists a\in I$, $\exists b\in J$, such that $c=a\sqcup b$, hence $\forall x\in L$ such that $c\sqsubseteq x$ we have $a\sqsubseteq x$ and $b\sqsubseteq x$ so that $x\in I$ and $x\in J$ proving that $x\in(I\cap J)$. Finally, $\forall c_1,c_2\in(I\cap J)$, $\exists a_1,a_2\in I$, $\exists b_1,b_2\in J$ such that $c_1=a_1\sqcup b_1$ and $c_2=a_2\sqcup b_2$. But $(a_1\sqcap a_2)\in I,(b_1\sqcap b_2)\in J$ so that $(a_1\sqcap a_2)\sqcup(b_1\sqcap b_2)\in I\cap J$. Moreover $(a_1\sqcap a_2)\sqcup(b_1\sqcap b_2)\sqsubseteq(a_1\sqcup b_1)\sqcap(a_2\sqcup b_2)=(c_1\sqcap c_2)$ proving that $(c_1\sqcap c_2)\in(I\cap J)$. *End of Proof.*

A net $[\![x_\delta:\delta\in\Delta]\!]$ is *in* a set S if and only if $\forall\delta\in\Delta$, $x_\delta\in S$; it is *eventually in* S if and only if there is an element $\delta_0\in\Delta$ such that $[\![x_\delta:(\delta\in\Delta)\underline{\text{and}}(\delta\geq\delta_0)]\!]$ is *in* S.

DEFINITION 2.2 Let $\mathbb{B}$ be the family of sets B such that
$\{\forall B\in\mathbb{B},$ (i) $B\in\mathbb{I}d_0(L)$
(ii) $\{\bigsqcup_\Delta x_\delta\in B\}\Rightarrow\{[\![x_\delta:\delta\in\Delta]\!]$ is *eventually in* B$\}\}$.

THEOREM 2.3 $\mathbb{B}$ is a *basis* for *open sets.*

Proof : According to (Kelley[1961], Th.11, p.47) it is sufficient to show that for every two members B_1 and B_2 of $\mathbb{B}$ and each point x in $B_1\cap B_2$ there is B_0 in $\mathbb{B}$ such that $x\in B_0$ and $B_0\subseteq B_1\cap B_2$. Hence $B_1\cap B_2$ is a convenient choice for such a B_0 provided that $B_1\cap B_2$ satifies conditions 2.2.(i) and 2.2.(ii). (i), According to lemma 2.1, the set $B_1\cap B_2$ belongs to $\mathbb{I}d_0(L)$. (ii), if $\bigsqcup_\Delta x_\delta\in B_0$ then $[\![x_\delta:\delta\in\Delta]\!]$ is *eventually in* B_1 and *eventually in* B_2 which implies that $\exists\delta_1,\delta_2\in\Delta$ such that $[\![x_\delta:(\delta\in\Delta)\underline{\text{and}}(\delta\geq\delta_1)]\!]$ is *in* B_1 and $[\![x_\delta:(\delta\in\Delta)\underline{\text{and}}(\delta\geq\delta_2)]\!]$ is *in* B_2. Since Δ is up-directed, $\exists\delta_0\in\Delta$ such that $\delta_0\geq\delta_1$ and $\delta_0\geq\delta_2$ so that $[\![x_\delta:(\delta\in\Delta)\ \underline{\text{and}}\ (\delta\geq\delta_0)]\!]$ is *in* $B_1\cap B_2$. *End of Proof.*

DEFINITION 2.4 A $\sqcup$-*topological lattice* L is a complete lattice L with $\mathbb{B}$ as basis for open sets.

THEOREM 2.5 A $\sqcup$-topological lattice is a T_0-*space* (but not a T_1-space).

Proof : Let $x,y\in L$ and $\bar{x},\bar{y}$ be their respective *closures.* We have to prove $\{\{\bar{x}=\bar{y}\}\Rightarrow\{x=y\}\}$. $\bar{x}$ and $\bar{y}$ are the *principal ideals* generated by the points x and y respectively, hence if these ideals are equal so are x and y (Grätzer[1971], p.22) proving that L is a T_0-space. Obviously $\forall x\in L$, $\perp\in\bar{x}$ proving that in general $x\neq\bar{x}$ so that L is not a T_1-space (and consequently not a Hausdorff space). *End of Proof.*

As usual (Kelley[1961], p.66), a net $[\![x_\delta:\delta\in\Delta]\!]$ in a $\sqcup$-topological lattice L *converges* to x (in symbols $x_\delta \to x$) if and only if it is eventually in each open set containing x.

LEMMA 2.5 $\{(x_\delta \to x)$ and $(\forall\delta\in\Delta, x_\delta \sqsubseteq x)\} \Rightarrow \{x = \bigsqcup_\Delta x_\delta\}$

Proof : Suppose that x is an upper bound of $\{x_\delta:\delta\in\Delta\}$ with $x \not\sqsubseteq \bigsqcup_\Delta x_\delta$. If $C=\{y\in L: y \sqsubseteq \bigsqcup_\Delta x_\delta\}$ then L-C is open and contains x, therefore $[\![x_\delta:\delta\in\Delta]\!]$ is eventually in L-C then $x_{\delta_0}\in L-C$ for some $\delta_0\in\Delta$ contradicting our assumption that $x_{\delta_0}\in C$. *End of Proof.*

THEOREM 2.6 $\{x_\delta \to x\} \iff \{x \sqsubseteq \bigsqcup_\Delta x_\delta\}$

Proof : By 2.2.(i) every open set U containing x contains its upper bound $\bigsqcup_\Delta x_\delta$ and therefore by 2.2.(ii), $[\![x_\delta:\delta\in\Delta]\!]$ is eventually in U proving that $\{x \sqsubseteq \bigsqcup_\Delta x_\delta\} \Rightarrow \{x_\delta \to x\}$. Reciprocally, assume that $\{x_\delta \to x\}$, then $[\![x_\delta \sqcap x:\delta\in\Delta]\!]$ is a monotone increasing net since meet is order-preserving, moreover every open set of $\mathbb{B}$ containing x contains the x_δ such that $\delta \geq \delta_0$ hence it contains all $x_\delta \sqcap x$ for $\delta\geq\delta_0$ so that $\{(x_\delta \sqcap x) \to x\}$. But $\forall\delta\in\Delta$, $(x_\delta \sqcap x) \sqsubseteq x$ and lemma 2.5 implies $x = \bigsqcup_\Delta (x_\delta \sqcap x) \sqsubseteq (\bigsqcup_\Delta x_\delta) \sqcap x$ proving that $x \sqsubseteq \bigsqcup_\Delta x_\delta$. *End of Proof.*

Since a $\sqcup$-topological lattice is not a Hausdorff space each net in the space does not converge to at most one point. Therefore we must define a notion of limit different from convergence so that the usual rule concerning substitution of equals for equals may remain valid : if $\lim [\![x_\delta:\delta\in\Delta]\!] = s$ and $\lim [\![x_\delta:\delta\in\Delta]\!] = t$ then s=t since we always use equality in the sense of identity.

DEFINITION 2.7 For every net $[\![x_\delta:\delta\in\Delta]\!]$ in L, we define its *limit*, $\lim [\![x_\delta:\delta\in\Delta]\!] = \bigsqcup_\Delta x_\delta$.

DEFINITION 2.8 A function $f\in L\to L$ is *continuous* if and only if for each net $[\![x_\delta:\delta\in\Delta]\!]$ in L which converges to a point s, the composition $[\![f(x_\delta):\delta\in\Delta]\!]$ converges to f(s).

It is easy to prove that a function is continuous in the sense of definition 3.8 if and only if it is continuous in the usual topological sense : the inverse image of each open set is open, (Kelley[1961], Th1., p.66).

THEOREM 2.9 Let $f\in D\to D$ be a function and $[\![x_\delta:\delta\in\Delta]\!]$ a net in D. Then f is continuous if and only if

(i) f is monotone

(ii) $\forall [\![x_\delta:\delta\in\Delta]\!]$, $f(\lim [\![x_\delta:\delta\in\Delta]\!]) \sqsubseteq \lim [\![f(x_\delta):\delta\in\Delta]\!]$.

Proof : Let $x,y\in L$, then $\{\{x \sqsubseteq y\} \iff \{x\in\bar{y}\}\}$.
Also f continuous implies $f(\bar{y}) \subseteq \overline{f(y)}$ (Kelley[1961], Th.1.(g), p.86). Therefore $\{x\sqsubseteq y\} \Rightarrow \{x\in\bar{y}\} \Rightarrow \{f(x)\in f(\bar{y})\} \Rightarrow \{f(x)\in\overline{f(y)}\} \Rightarrow \{f(x)\sqsubseteq f(y)\}$ proving that a continuous function is monotone. All nets such as $[\![x_\delta:\delta\in\Delta]\!]$ converges to $\bigsqcup_\Delta x_\delta$ hence if f is continuous then $[\![f(x_\delta):\delta\in\Delta]\!]$ converges to $f(\bigsqcup_\Delta x_\delta)$ by definition 2.8, and theorem 2.6 implies that $f(\bigsqcup_\Delta x_\delta) \sqsubseteq \bigsqcup_\Delta f(x_\delta)$ proving that $f(\lim [\![x_\delta:\delta\in\Delta]\!]) \sqsubseteq \lim [\![f(x_\delta):\delta\in\Delta]\!]$.

Reciprocally, if f is monotone and such that $f(\bigsqcup_\Delta x_\delta) \sqsubseteq \bigsqcup_\Delta f(x_\delta)$ then for each net $[\![x_\delta:\delta\in\Delta]\!]$ in L which converges to a point s we have (theorem 2.6) $s \sqsubseteq \bigsqcup_\Delta x_\delta$ and by monotony $f(s) \sqsubseteq f(\bigsqcup_\Delta x_\delta) \sqsubseteq \bigsqcup_\Delta f(x_\delta)$ proving that $[\![f(x_\delta):\delta\in\Delta]\!]$ converges to f(s) and f is continuous. *End of Proof.*

Hereafter we will denote by $L \overset{c}{\to} M$ the set of continuous functions on the topological space L into the topological space M.

THEOREM 2.10 Let L be a complete lattice and f a monotone function on L into L. The set of fixed points of f is a complete lattice for the ordering of L with infimum $lfp(f) = \sqcap\{x\in L: f(x) \sqsubseteq x\}$.

(this theorem is proved in Tarski[1955]).

THEOREM 2.11 The least fixed point of any continuous function $f\in L \overset{c}{\to} L$ is $lfp(f) = \lim_{k\to\infty} f^k(p)$ where $p \sqsubseteq lfp(f)$, p is a pre-fixed point of f and f^k is the k-fold composition of f with itself.

Proof : Since f is continuous it is monotone and Tarski's theorem 2.10 proves the existence of $lfp(F) = \sqcap\{x\in L: f(x)\sqsubseteq x\}$. By recurrence on k using $p \sqsubseteq lfp(f)$ and f monotone we have $f^k(p) \sqsubseteq lfp(F)$ and therefore $\bigsqcup_{k=0}^{\infty} f^k(p) \sqsubseteq lfp(F)$. Theorem 2.9 implies $f(\bigsqcup_{k=0}^{\infty} f^k(p)) \sqsubseteq \bigsqcup_{k=0}^{\infty} f^{k+1}(p) = \bigsqcup_{k=0}^{\infty} f^k(p)$ since p is a pre-fixed point of f that is $p \sqsubseteq f(p)$. Since $\bigsqcup_{k=0}^{\infty} f^k(p) \in \{x\in L: f(x) \sqsubseteq x\}$ theorem 2.10 implies $lfp(F) \sqsubseteq \bigsqcup_{k=0}^{\infty} f^k(p)$, and by antisymmetry we have $lfp(F) = \bigsqcup_{k=0}^{\infty} f^k(p)$. *End of Proof.*

Remark : Notice that $\{x_\delta \to x\}$ and $\{y_\delta \to y\}$ if and only if $x \sqsubseteq \bigsqcup_\Delta x_\delta$ and $y \sqsubseteq \bigsqcup_\Delta y_\delta$ therefore $(x \sqcup y) \sqsubseteq (\bigsqcup_\Delta x_\delta \sqcup \bigsqcup_\Delta y_\delta) = \bigsqcup_\Delta (x_\delta \sqcup y_\delta)$ proving that $\{(x_\delta \sqcup y_\delta) \to (x \sqcup y)\}$, L is a $\sqcup$-topological lattice (Birkhoff[1967], p.248). However in general the net $[\![x_\delta \sqcap y_\delta:\delta\in\Delta]\!]$ does not converge to $x \sqcap y$ since $\bigsqcup_\Delta (x_\delta \sqcap y_\delta) \sqsubseteq (\bigsqcup_\Delta x_\delta \sqcap \bigsqcup_\Delta y_\delta) = (x \sqcap y)$, hence L is not in general a topological lattice. Another consequence is that L is not in general a continuous lattice (Scott[1972]). *End of Remark.*

Let X and Y be topological T_0-spaces and $X \circ\!\!\to Y$ the space of all continuous functions f on X into Y provided with the *product topology* (or equivalently the *coordinatewise convergence topology*). Then, as usual (Kelley[1961], p.90), a subbase for the neighborhoods consists in all sets of the form $\{f : f(x) \in U\}$ where $x \in X$ and U is an open set of Y.

The induced partial ordering $\leqslant$ on $X \circ\!\!\to Y$ is such that $\{\forall f, g \in X \to Y, \{f \leqslant g\} \iff \{\forall x \in X, f(x) \sqsubseteq g(x)\}\}$.

THEOREM 2.12 Let $(L, \sqsubseteq, \bot, \top, \sqcup, \sqcap, \bigsqcup, \bigsqcap)$ and $(L', \sqsubseteq', \bot', \top', \sqcup', \sqcap', \bigsqcup', \bigsqcap')$ be complete lattices then $(L \circ\!\!\to L', \leqslant, \downarrow, \uparrow, \dot\sqcup, \dot\sqcap, \dot\bigsqcup)$ is a complete lattice, and the product topology is *coarser* (or less *finer*) than the one of the $\dot\sqcup$-topological lattice.

Proof : We define $\downarrow = \lambda x.\bot'$, $\uparrow = \lambda x.\top'$, $(\forall x \in L, (f \dot\sqcup g)(x) = f(x) \sqcup' g(x))$, $(\forall x \in L, (f \dot\sqcap g)(x) = f(x) \sqcap' g(x))$. $\downarrow, \uparrow, f \dot\sqcap g$ and $f \dot\sqcup g$ are continuous functions and pointwise arguments then show that $L \circ\!\!\to L'$ is a lattice. The infinite union $\dot\bigsqcup$ is defined by $\{\forall F \subseteq L \circ\!\!\to L', \forall x \in L, (\dot\bigsqcup F)(x) = \bigsqcup'\{f(x) : f \in F\}\}$. $\dot\bigsqcup F$ is monotone since $\{x \sqsubseteq y\} \Rightarrow \{\forall f \in F, f(x) \sqsubseteq' f(y)\} \Rightarrow \{\bigsqcup'\{f(x) : f \in F\} \sqsubseteq' \bigsqcup'\{f(y) : f \in F\}\} \Rightarrow \{(\dot\bigsqcup F)(x) \sqsubseteq' (\dot\bigsqcup F)(y)\}$ by monotony of the f and $\bigsqcup'$. Let $[\![x_\delta : \delta \in \Delta]\!]$ be a monotone increasing net on L. $(\dot\bigsqcup F)(\lim [\![x_\delta : \delta \in \Delta]\!]) = (\dot\bigsqcup F)(\bigsqcup_\Delta x_\delta) = \bigsqcup_F'(f(\bigsqcup_\Delta x_\delta)) \sqsubseteq \bigsqcup_F' \bigsqcup_\Delta' f(x_\delta)$ since the f are continuous and $\bigsqcup'$ is monotone. Now $\bigsqcup_F \bigsqcup_\Delta' f(x_\delta) = \bigsqcup_\Delta' \bigsqcup_F f(x_\delta) = \bigsqcup_\Delta'(\dot\bigsqcup F)(x_\delta) = \lim [\![(\dot\bigsqcup F)(x_\delta) : \delta \in \Delta]\!]$. Therefore theorem 2.9 (adapted to deal with different domain L and range L') proves the continuity of $(\dot\bigsqcup F)$. A pointwise argument shows that $(\dot\bigsqcup F)$ is the least upper bound of F proving that $L \circ\!\!\to L'$ is a complete join semi-lattice. It has an infimum, therefore it is complete (Birkhoff[1967], p.114-115).

Let us prove that every open set of the product topology is open in the $\dot\sqcup$-topological lattice. Let $B = (f : \forall x \in L, f(x) \in U')$ where U' is an open set of the base of L'. (i) $\forall f \in B$, $\forall g \in L \circ\!\!\to L'$, $\{f \leqslant g\} \Rightarrow \{\forall x \in L, f(x) \sqsubseteq' g(x)\} \Rightarrow \{\forall x \in L, g(x) \in U'\}$ since $f(x) \in U'$, $f(x) \sqsubseteq g(x)$ and U' is open. Hence $\{f \leqslant g\} \Rightarrow \{g \in B\}$. Now (ii) $\forall f, g \in B$, $\forall x \in L$, $(f(x) \sqcap g(x)) \in U'$ since $f(x) \in U'$, $g(x) \in U'$ and U' is a dual ideal. Therefore $(f \dot\sqcap g) \in B$ proving that B is a dual ideal of $L \circ\!\!\to L'$.(iii) Let $[\![f_\delta : \delta \in \Delta]\!]$ be a net in $L \circ\!\!\to L'$ such that $\dot\bigsqcup_\Delta f_\delta \in B$. This implies $\forall x \in L$, $(\dot\bigsqcup_\Delta f_\delta)(x) = \bigsqcup'(f_\delta(x)) \in U'$ and $[\![f_\delta(x) : \delta \in \Delta]\!]$ is eventually in U'. Hence for all $\delta \geq \delta_0$, $f_\delta(x) \in U'$ proving that $f_\delta \in B$ whenever $\delta \geq \delta_0$ so that $[\![f_\delta : \delta \in \Delta]\!]$ is eventually in B. According to definition 2.2, B is in the open base of the $\dot\sqcup$-topological lattice $L \circ\!\!\to L'$. *End of Proof.*

LEMMA 2.13 $\forall f, g \in L \circ\!\!\to L$, $\{f \leqslant g\} \Rightarrow \{lfp(f) \sqsubseteq lfp(g)\}$

Proof : Since $f \leqslant g$ we have $\forall x \in L$, $\{g(x) \sqsubseteq x\} \Rightarrow \{f(x) \sqsubseteq x\}$, therefore $\{x \in L : g(x) \sqsubseteq x\} \subseteq \{x \in L : f(x) \sqsubseteq x\}$ and consequently $lfp(f) = \bigsqcap\{x \in L : f(x) \sqsubseteq x\} \sqsubseteq \bigsqcap\{x \in L : g(x) \sqsubseteq x\} = lfp(g)$
End of Proof.

3. SYNTAX AND DEDUCTIVE SEMANTICS OF THE LANGUAGE

We introduce a simple programming language which contains the main programming concepts relevant for our purpose. The language contains skip, assignment, conditional, compound and loop statements, blocks with declarations and procedures. Admittedly we can do without labels and unconditional branch statements. We will assume that a variable of type t is initialized when declared, eventually by the value Ω_t denoting the uninitialized value of type t. We use blocks as in ALGOL 60 with the syntactic restriction that at each program point only one declaration may be applicable to a visible identifier. In other words, an identifier declared local to a block A cannot be redeclared in a block B inner to A, with the effect of masking the outer declaration. The mechanism used in procedures for passing parameters is the one of value parameters and result parameters of ALGOL W. Value-result parameter passing has been eliminated because it can be easily simulated by a value parameter passing followed by a result parameter passing. The same way, we will assume that actual parameters are variables, since an expression can be passed as an actual value parameter using a local intermediate variable holding the value of this expression. We will assume that no global variable is visible in a procedure because when a procedure needs to access to or makes a side-effect on a global variable, the parameter passing mechanism can be used. A program is a procedure the value parameters of which are the inputs of the program whereas the result parameters are the outputs. The above restrictions have been accepted for the sake of simplicity and could be eliminated by a simple syntactic program transformer. However the restriction that no functions or procedures can be passed to or returned from a procedure as well as the exclusion of arbitrary jumps out of blocks and procedures clearly simplify the problem of defining the semantics of this language.

Example 3.0.1 :

```
proc main( value input∈𝕀  ; result output∈𝕀  ) =
  ( proc f-91( value x∈𝕀  ; result y∈𝕀  ) =
       if x > 100 then
           y := x-10;
       else
           begin new z∈𝕀  := x+11 ;
              begin new t∈𝕀  := Ω𝕀  ;
                 f-91(z;t);
                 f-91(t;y);
              end;
           end;
       fi;
     f-91(input;output) ).
```

End of Example.

Knowing the set of possible states of variables before a procedure call, we are interested in discovering the set of corresponding states after the procedure has been called. Representing state sets by predicates it seems appropriate to describe the effect of a procedure call by a predicate transformer ϕ associated with the procedure f (Dijkstra [1976]). Whenever $P(v_1,\dots,v_n)$ is true of the actual value parameters before the procedure call $f(v_1,\dots,v_n,r_1,\dots,r_q)$ then $\phi(P)(v_1,\dots,v_n,r_1,\dots,r_q)$ is true after the call. Since procedures cannot have non-explicit side effects no variable of the environment other than $(r_1,\dots,r_q)$ has been affected. Notice that $\phi(P)$ is the conjunction of a predicate on $(v_1,\dots,v_n)$ stating the necessary and sufficient conditions for the procedure execution to terminate and a predicate defining the resulting value of $(r_1,\dots,r_q)$. For example the predicate transformer ϕ associated with the procedure f defined by :

$\underline{\text{proc}}\ f(\ \underline{\text{value}}\ x\in\mathbb{I}\ ;\ \underline{\text{result}}\ y\in\mathbb{I}\ ) = \underline{\text{if}}\ x=0\ \underline{\text{then}}\ y:=1\ ;\ \underline{\text{else}}\ x:=x-1;\ f(x,y);$

$y:=(x+1)*y;\ \underline{\text{fi}};$

would be :

$\phi=\lambda P.\{\lambda(x,y).[P(x)\ \underline{\text{and}}\ (0\le x)\ \underline{\text{and}}\ (y=x!)]\}$

since for a negative input parameter x the procedure f never terminates. ϕ is obtained as the least fixed point of a system of functional equations associated with the body of the procedure f.

3.1. SYSTEM OF SEMANTIC EQUATIONS ASSOCIATED WITH A PROCEDURE

An environment $(v_1\in t_1,\dots,v_n\in t_n)$ is associated with any program point. It defines the set of variables $v_1,\dots,v_n$ which are visible at this point as well as the type t_i of each variable v_i. To each statement I of the procedure body which is in environment $(v_1\in t_1,\dots,v_n\in t_n)$ we associate a predicate transformer $\underline{\text{pt}}(I,(v_1\in t_1,\dots,v_n\in t_n))$ of type $(t_1\times\dots\times t_n\to\mathbb{B})\xrightarrow{\circ}(t_1\times\dots\times t_n\to\mathbb{B})$ where $\mathbb{B}=\{\underline{\text{true}},\underline{\text{false}}\}$ which defines the effect of this statement.

3.1.1. SKIP STATEMENT

$\underline{\text{pt}}(\underline{\text{skip}};,(v_1\in t_1,\dots,v_n\in t_n)) = \lambda P.\{P\}$

3.1.2. ASSIGNMENT STATEMENT

$\underline{\text{pt}}(v_i:=E(v_1,\dots,v_n);\ ,\ (v_1\in t_1,\dots,v_n\in t_n)) = \underline{\text{assign}}(i,E,(t_1,\dots,t_n))$ where $(1\le i\le n)$ and $E\in(t_1\times\dots\times t_n\to t_i)$ and $\underline{\text{assign}}(i,E,(t_1,\dots,t_n))=$

$\lambda P.\{\lambda(v_1,\dots,v_n).[\exists a\in t_i:P(v_1,\dots,v_{i-1},a,v_{i+1},\dots,v_n)\ \underline{\text{and}}$

$v_i=E(v_1,\dots,v_{i-1},a,v_{i+1},\dots,v_n)]\}.$

3.1.3. CONDITIONAL STATEMENT

$\underline{\text{pt}}(\underline{\text{if}}\ Q(v_1,\dots,v_n)\ \underline{\text{then}}\ It\ \underline{\text{else}}\ If\ \underline{\text{fi}};\ ,\ (v_1\in t_1,\dots,v_n\in t_n)) =$

$\lambda P.\{\phi_{It}(P\ \underline{\text{and}}\ Q)\ \underline{\text{or}}\ \phi_{If}(P\ \underline{\text{and}}\ \underline{\text{not}}\ Q)\}$

where $Q\in((t_1\times\dots\times t_n)\to\mathbb{B})$ and the auxiliary predicate transformers ϕ_{It} and ϕ_{If} are defined by $\phi_{It}=\underline{\text{pt}}(It,(v_1\in t_1,\dots,v_n\in t_n))$ and $\phi_{If}=\underline{\text{pt}}(If,(v_1\in t_1,\dots,v_n\in t_n))$.

Moreover if $P,Q\in(t_1\times\dots\times t_n\to\mathbb{B})$ then $P\ \underline{\text{and}}\ Q$ is $\lambda(v_1,\dots,v_n).\ [P(v_1,\dots,v_n)\ \underline{\text{and}}\ Q(v_1,\dots,v_n)]$, same definition for $\underline{\text{or}}$ and $\underline{\text{not}}$.

3.1.4. COMPOUND STATEMENT

$\underline{\text{pt}}(I_1I_2\dots I_p\ ,\ (v_1\in t_1,\dots,v_n\in t_n)) = \lambda P.\{\phi_{I_p}\circ\phi_{I_{p-1}}\circ\dots\circ\phi_{I_2}\circ\phi_{I_1}(P)\}$ where $\circ$ denotes functional composition and $\forall j\in[1,p],\ \phi_{I_j}=\underline{\text{pt}}(I_j,(v_1\in t_1,\dots,v_n\in t_n))$.

3.1.5. BLOCKS AND DECLARATIONS

$\underline{\text{pt}}(\underline{\text{begin}}\ \underline{\text{new}}\ v\in t := E(v_1,\dots,v_n);I\ \underline{\text{end}};\ ,\ (v_1\in t_1,\dots,v_n\in t_n)) =$

$\lambda P.\{\bar{\sigma}^{n+1}_{n+1}(\phi_I(\underline{\text{block}}(E,(t_1,\dots,t_n),t)(P)))\}$

where $\forall j\in[1,n]$ v is syntactically different from v_j and $E\in(t_1\times\dots\times t_n\to t)$ and $\phi_I=\underline{\text{pt}}(I,(v_1\in t_1,\dots,v_n\in t_n,v\in t))$ and $\underline{\text{block}}(E,(t_1,\dots,t_n),t) =$

$\lambda P.\{\lambda(v_1,\dots,v_n,v).[P(v_1,\dots,v_n)\ \underline{\text{and}}\ v=E(v_1,\dots,v_n)]\}$

Let P, i, n, $i_1,\dots,i_q$ be such that $P\in(t_1\times\dots\times t_n\to\mathbb{B})$, $(1\le i\le n)$, $(\forall k\in[1,q],\ (1\le i_k\le n))$ and $(\forall k,l\in[1,q],(k\ne l)\Rightarrow(i_k\ne i_l))$.

We use the following notations :

$\bar{\sigma}^n_i(P) = \lambda(v_1,\dots,v_{i-1},v_{i+1},\dots,v_n).[\exists a\in t_i\ :\ P(v_1,\dots,v_{i-1},a,v_{i+1},\dots,v_n)]$

$\bar{\sigma}^n_{i_1,\dots,i_q}(P) = \bar{\sigma}^n_{i_1}\circ\bar{\sigma}^n_{i_2}\circ\dots\circ\bar{\sigma}^n_{i_q}(P)$

$\sigma^n_i(P) = \lambda(v_i).[(\forall k\in[1,n]-\{i\},\exists a_k\in t_k)\ :\ P(a_1,\dots,a_{i-1},v_i,a_{i+1},\dots,a_n)]$

$\sigma^n_{i_1,\dots,i_q}(P) = \lambda(v_{i_1},\dots,v_{i_q}).[(\forall k\in[1,n]-\{i_1,\dots,i_q\},\exists a_k\in t_k)\ :\ P(w_1,\dots,w_n)]$

where $\forall l\in[1,n],\ w_l=\underline{\text{if}}\ l\in\{i_1,\dots,i_q\}\ \underline{\text{then}}\ v_l\ \underline{\text{else}}\ a_l\ \underline{\text{fi}}.$

A block with multiple declarations such as :

$\underline{\text{begin}}\ \underline{\text{new}}\,v_1\in t_1:=E_1;\ \underline{\text{new}}\,v_2\in t_2:=E_2;\ \dots\ ;\underline{\text{new}}\,v_s\in t_s:=E_s;\ I\ \underline{\text{end}};$

is equivalent to :

$\underline{\text{begin}}\ \underline{\text{new}}\,v_1\in t_1:=E_1;\ \underline{\text{begin}}\ \underline{\text{new}}\,v_2\in t_2:=E_2;\ \dots\ ;\underline{\text{begin}}\ \underline{\text{new}}\,v_s\in t_s:=E_s;\ I\ \underline{\text{end}};$
$\dots\underline{\text{end}};\underline{\text{end}};$

3.1.6. PROCEDURE DECLARATION

$\underline{\text{pt}}(\underline{\text{proc}}\ f(\underline{\text{value}}\ x_1\in t_1;\dots;x_q\in t_q;\ \underline{\text{result}}\ y_{q+1}\in t_{q+1},\dots,y_r\in t_r)=I;\ ,\ (z_1\in t'_1,\dots,z_n\in t'_n))$
$=\phi_f$

where :

$\phi_f \in (t_1 \times \ldots \times t_q \to \mathbb{B}) \circ\!\!\to (t_1 \times \ldots \times t_q \times t_{q+1} \times \ldots \times t_r \to \mathbb{B})$

$\phi_f = \lambda P.\{\sigma^{r+q}_{1,\ldots,r}(\phi_I(\underline{\text{entry}}[(t_1,\ldots,t_q),(t_{q+1},\ldots,t_r)](P)))\}$

$\phi_I = \underline{\text{pt}}(I,(x'_1\epsilon t_1,\ldots,x'_q\epsilon t_q, x_1\epsilon t_1,\ldots,x_q\epsilon t_q, y_{q+1}\epsilon t_{q+1},\ldots,y_r\epsilon t_r))$

where the x'_i are new identifiers syntactically different from the x_i and y_j.

$\underline{\text{entry}}[(t_1,\ldots,t_q),(t_{q+1},\ldots,t_r)](P)$ where $P\epsilon(t_1 \times \ldots \times t_q \to \mathbb{B})$ is equal to

$\lambda(x'_1,\ldots x'_q,y_{q+1},\ldots,y_r,x_1,\ldots,x_r).[P(x'_1,\ldots,x'_q)\ \underline{\text{and}}\ \underline{\text{AND}}_{j=1}^{q}(x_j=x'_j)\ \underline{\text{and}}\ \underline{\text{AND}}_{j=q+1}^{r}(y_j=\Omega_{t_j})]$.

Since the effect of a procedure is independent of the environment at the point of declaration we will consider that the declaration of a procedure is visible everywhere in the program.

Intuitively the value parameters x_j are copied in the new variables x'_j on procedure entry. The predicate which holds before execution of the body I states that P is true of the values of the input parameters, $x_j=x'_j$ and that the result parameters y_j are not initialized. ϕ_I represents the effect of a terminating execution of the procedure body. It returns a predicate defining the final value of the value and result parameters and stating the termination condition. The final values of the value parameters $x_1,\ldots,x_q$ are of no interest to the calling environment so that they are eliminated. The result is therefore a predicate which depends on the input values of the value parameters and on the output values of the result parameters. It specifies the outputs as a function of the inputs as well as the condition which must be true of the input parameters for the procedure to terminate.

3.1.7. PROCEDURE CALL STATEMENT

$\underline{\text{pt}}(f(v_{i_1},\ldots,v_{i_q};v_{i_{q+1}},\ldots,v_{i_r});\ ,\ (v_1\epsilon t_1,\ldots,v_n\epsilon t_n)) =$

$$\lambda P.\{\Sigma^n_{i_1,\ldots,i_r}(\phi_f(\sigma^n_{i_1,\ldots,i_q}(P)))\ \underline{\underline{\text{and}}}\ \Sigma^n_{i_{q+1},\ldots,i_r}(\bar\sigma^n_{i_{q+1},\ldots,i_r}(P))\}$$

where $\forall k,l\epsilon[1,r]$, $(k\neq l)$ implies that v_{i_k} and v_{i_l} are syntactically different variables, $\forall k\epsilon[1,r]$ the type of the actual parameter v_{i_k} is the same as the type of the corresponding formal parameter. Moreover if $Q\epsilon(t_1 \times \ldots \times t_r \to \mathbb{B})$, $r\leq n$, $(\forall k\epsilon[1,r], 1\leq i\leq n)$ and $(\forall k,l\epsilon[1,r],\ (k\neq l) \Rightarrow (i_k\neq i_l))$ we define the notation :

$$\Sigma^n_{i_1,\ldots,i_r}(Q) = \lambda(v_1,\ldots,v_n).[Q(v_{i_1},\ldots,v_{i_r})]$$

if $Q\epsilon(t_1 \times \ldots \times t_{n-r} \to \mathbb{B})$ we can use the notation :

$$\bar\Sigma^n_{i_1,\ldots,i_r}(Q) = \lambda(v_1,\ldots,v_n).[Q(v_{j_1},\ldots,v_{j_{n-r}})] \quad \text{where } \forall k\epsilon[1,n-r],$$

$j_k = \underline{\min}\{i:(i>j_{k-1})\ \underline{\text{and}}\ (\forall l\epsilon[1,r], i\neq i_l)\}$ with $j_0=0$.

Intuitively if P holds for the variables $v_1,\ldots,v_n$ which are in the environment of the calling point, $\sigma^n_{i_1,\ldots,i_q}(P)$ defines the possible states of the actual value parameters which are passed to the procedure f. The effect of the procedure call is described by $\phi_f(\sigma^n_{i_1,\ldots,i_q}(P))(v_{i_1},\ldots,v_{i_r})$ whereas $\Sigma^n_{i_{q+1},\ldots,i_r}(\bar\sigma^n_{i_{q+1},\ldots,i_r}(P))$ specifies the value of those variables which are not used as actual result parameters and therefore have not been modified by the procedure.

3.1.8. LOOP STATEMENT

$\underline{\text{pt}}(\underline{\text{while}}\ Q(v_1,\ldots,v_n)\ \underline{\text{do}}\ I\ \underline{\text{od}};\ ,\ (v_1\epsilon t_1,\ldots,v_n\epsilon t_n)) = \phi$

where $Q\ \epsilon\ (t_1 \times \ldots \times t_n \to \mathbb{B})$

$\phi\ = \sigma^{2n}_{n+1,\ldots,2n}(\psi(\underline{\text{entry}}[(t_1,\ldots,t_n),()](P)))$

$\psi\ \epsilon\ (t_1 \times \ldots \times t_n \times t_1 \times \ldots \times t_n \to \mathbb{B})$

$\psi\ = \lambda P.\{(P\ \underline{\underline{\text{and not}}}\ \Sigma^{2n}_{n+1,\ldots,2n}(Q))\ \underline{\underline{\text{or}}}\ \psi(\phi_I(P\ \underline{\underline{\text{and}}}\ \Sigma^{2n}_{n+1,\ldots,2n}(Q)))\}$

$\phi_I = \underline{\text{pt}}(I,(v'_1\epsilon t_1,\ldots,v'_n\epsilon t_n,v_1\epsilon t_1,\ldots,v_n\epsilon t_n))$

3.1.9. EXAMPLE

The following program is hoped to compute x! for any integer x :

```
φ1|         proc f( value x∈I ; result y∈I ) =
  |φ2|           if x=0 then
  |  |φ3|            y:=1;
  |  |           else
  |  |φ4|φ5|         x:=x-1;
  |  |  |  φ6|       f(x;y);
  |  |  |     φ7|    y:=(x+1)*y;
  |  |           fi;
```

The system of semantic equations associated with this procedure is :

$\phi_1\ \epsilon\ (\mathbb{I}\to\mathbb{B})\circ\!\!\to(\mathbb{I}^2\to\mathbb{B})$

$\quad = \lambda P.\{\sigma^3_{1,2}(\phi_2(\underline{\text{entry}}[(\mathbb{I}),(\mathbb{I})](P)))\}$

$\quad = \lambda P.\{\sigma^3_{1,2}(\phi_2(\lambda(a,y,x).[P(a)\ \underline{\text{and}}\ (x=a)\ \underline{\text{and}}\ (y=\Omega)]))\}$

$\phi_2\ \epsilon\ (\mathbb{I}^3\to\mathbb{B})\circ\!\!\to(\mathbb{I}^3\to\mathbb{B})$

$\quad = \lambda P.\{\phi_3(P\ \underline{\underline{\text{and}}}\ \lambda(a,y,x).[x=0])\ \underline{\underline{\text{or}}}\ \phi_4(P\ \underline{\underline{\text{and not}}}\ \lambda(a,y,x).[x=0])\}$

$\quad = \lambda P.\{\phi_3(\lambda(a,y,x).[P(a,y,x)\ \underline{\text{and}}\ x=0])\ \underline{\underline{\text{or}}}\ \phi_4(\lambda(a,y,x).[P(a,y,x)\ \underline{\text{and}}\ (x\neq 0)])\}$

$\phi_3\ \epsilon\ (\mathbb{I}^3\to\mathbb{B})\circ\!\!\to(\mathbb{I}^3\to\mathbb{B})$

$\quad = \underline{\text{assign}}(2,\lambda(a,y,x).[1],\mathbb{I}^3)$

$\quad = \lambda P.\{\lambda(a,y,x).[\exists m\epsilon\mathbb{I}:P(a,m,x)\ \underline{\text{and}}\ (y=1)]\}$

$\phi_4\ \epsilon\ (\mathbb{I}^3\to\mathbb{B})\circ\!\!\to(\mathbb{I}^3\to\mathbb{B})$

$\quad = \lambda P.\{\phi_7\circ\phi_6\circ\phi_5(P)\}$

$\phi_5\ = \underline{\text{assign}}(3,\lambda(a,y,x).[x-1],\mathbb{I}^3)$

$\quad = \lambda P.\{\lambda(a,y,x).[\exists m\epsilon\mathbb{I}:P(a,y,m)\ \underline{\text{and}}\ (x=m-1)]\}$

$\phi_6 \in (\mathbb{I}^3 \to \mathbb{B}) \circ\!\!\to (\mathbb{I}^3 \to \mathbb{B})$

$= \lambda P.\{\Sigma^3_{3,2}(\phi_1(\sigma^3_3(P)))\ \underline{\text{and}}\ \overline{\Sigma}^3_2(\overline{\sigma}^3_2(P))\}$

$= \lambda P.\{\lambda(a,y,x).[\phi_1(\sigma^3_3(P))(x,y)]\ \underline{\text{and}}\ \lambda(a,y,x).[\overline{\sigma}^3_2(P)(a,x)]\}$

$= \lambda P.\{\lambda(a,y,x).[\phi_1(\sigma^3_3(P))(x,y)\ \underline{\text{and}}\ \sigma^3_{1,3}(P)(a,x)]\}$

$\phi_7 \in (\mathbb{I}^3 \to \mathbb{B}) \circ\!\!\to (\mathbb{I}^3 \to \mathbb{B})$

$= \underline{\text{assign}}(2, \lambda(a,y,x).[(x+1)*y], \mathbb{I}^3)$

$= \lambda P.\{\lambda(a,y,x).[\exists m \in \mathbb{I}: P(a,m,x)\ \underline{\text{and}}\ (y=(x+1)*m)]\}$

This system of equations can be simplified by elimination of $\phi_2,\ldots,\phi_7$ we get :

$$\left\{\begin{array}{l}\phi_1 = \lambda\psi.[\![\lambda P.\{\lambda(a,y).[P(a)\ \underline{\text{and}}\ (a=0)\ \underline{\text{and}}\ (y=1)]\ \underline{\text{or}}\ \overline{\sigma}^3_3(\lambda(a,y,x).[\exists n:\psi(\lambda x.[P(x+1) \\ \qquad \underline{\text{and}}\ (x+1\neq 0)])(x,n)\ \ \underline{\text{and}}\ P(a)\ \underline{\text{and}}\ (a\neq 0)\ \underline{\text{and}}\ (x=a-1)\ \underline{\text{and}}\ (y=a*n)])\}]\!][\![\phi_1]\!]\end{array}\right.$$

As will be shown in the next paragraph, the solution ϕ_1^∞ of this fixed point functional equation is :

$$\left[\phi_1^\infty = \lambda P.\{\lambda(a,y).[P(a)\ \underline{\text{and}}\ (0\leq a)\ \underline{\text{and}}\ (y=a!)]\}\right.$$

(f does not terminate for negative value parameters).

3.2. RESOLUTION OF THE SEMANTIC EQUATIONS

The system of semantic equations associated with a procedure is of the form $\phi = F[\![\phi]\!]$ where ϕ is a vector of predicate transformers $(\phi_1,\ldots,\phi_n)$. The system $\phi = F[\![\phi]\!]$ can be detailed as :

$$\phi_i \in ((t_{i_1} \times\ldots\times t_{i_p} \to \mathbb{B}) \circ\!\!\to (t_{i_1} \times\ldots\times t_{i_p} \times\ldots\times t_{i_q} \to \mathbb{B})) = (D_i \circ\!\!\to D'_i)$$

$$\left\{\begin{array}{l}\phi_i = F_i[\![\phi_1,\ldots,\phi_n]\!] = \lambda(\psi_1,\ldots,\psi_n).[\![\lambda P.\{f_i(P,\psi_1,\ldots,\psi_n)\}]\!][\![\phi_1,\ldots,\phi_n]\!] \\ i = 1,\ldots,n\end{array}\right.$$

The D_i and D'_i are complete lattices since $(t_1\times\ldots\times t_r \to \mathbb{B})$ is a complete lattice $(\Rightarrow, \lambda(v_1,\ldots,v_r).[\underline{\text{false}}], \lambda(v_1,\ldots,v_r).[\underline{\text{true}}], \underline{\text{or}}, \underline{\text{and}}, \underline{\text{OR}}, \underline{\text{AND}})$. Hence the space of continuous functions $(D_i \circ\!\!\to D'_i)$ is a complete lattice (theorem 2.12) therefore F is a map on the complete lattice $((D_1 \circ\!\!\to D'_1)\times\ldots\times(D_n \circ\!\!\to D'_n))$ into itself. F is continuous because each of the F_i, $i\in[1,n]$ are continuous (Kelley[1961], Th.3, p.91) so that $lfp(F)$ exists and can be constructed by successive approximations as defined by the fixed point theorem 2.11. To prove the continuity of the F_i, it suffices to prove that they are obtained by composition of the continuous unknown $(\psi_1,\ldots,\psi_n)$ and continuous basic functions, (Kelley[1961], p.85). The identity is continuous (3.1.1). For 3.1.2 we obviously have :

$$\lambda(x,y).[\exists a:(\underset{i\in\Delta}{\underline{\text{OR}}}\ P_i(a,y))\ \underline{\text{and}}\ (x=f(a,y))] \Rightarrow \lambda(x,y).(\underset{i\in\Delta}{\underline{\text{OR}}}\ [\exists a:P_i(a,y)\ \underline{\text{and}}\ (x=f(a,y))])$$

which can be extended to the case of more than two variables.

For 3.1.3, $\underline{\text{or}}$ and $\underline{\text{and}}$ are infinitely distributive in a complete boolean lattice, the operator $\underline{\text{not}}$ is not continuous but $(\underline{\text{not}}\ Q)$ is a constant therefore $h_Q(P) = P\ \underline{\text{and}}\ (\underline{\text{not}}\ Q)$ is continuous. For 3.1.4 we used composition of continuous functions. In 3.1.5, σ is continuous and so are $\overline{\sigma}$, Σ and $\overline{\Sigma}$. 3.1.6, 3.1.7 and 3.1.8 use basic functions previously examinated.

Before giving examples of resolution of the semantic equations let us point out that the functionals must be evaluated by computing the innermost terms first which corresponds to call by value (De Bakker[1976]).

Example : The following equation has been shown to correspond to the factorial procedure of paragraph 3.1.9 :

$$\left\{\begin{array}{l}\phi_1 = \lambda\psi.[\![\lambda P.\{\lambda(a,y).[P(a)\ \underline{\text{and}}\ (a=0)\ \underline{\text{and}}\ (y=1)]\ \ \underline{\text{or}}\ \ \overline{\sigma}^3_3(\lambda(a,y,x).[\exists m:\psi(\lambda x.[P(x+1) \\ \qquad \underline{\text{and}}\ (x+1\neq 0)])(x,m)\ \underline{\text{and}}\ P(a)\ \underline{\text{and}}\ (a\neq 0)\ \underline{\text{and}}\ (x=a-1)\ \underline{\text{and}}\ (y=a*m)]\}]\!][\![\phi_1]\!]\end{array}\right.$$

Solving by successive approximations starting from the initial approximation

$[\phi_1^0 = \lambda P.\{\lambda(a,y).[\underline{\text{false}}]\}$ we get :

$[\phi_1^1 = \lambda P.\{\lambda(a,y).[P(a)\ \underline{\text{and}}\ (a=0)\ \underline{\text{and}}\ (y=a!)]\}$

$[\phi_1^2 = \lambda P.\{\lambda(a,y).[P(a)\ \underline{\text{and}}\ (0\leq a\leq 1)\ \underline{\text{and}}\ (y=a!)]\}$

The computation of these first few iterates leads us to conjecture the following induction hypothesis :

$[\phi_1^k = \lambda P.\{\lambda(a,y).[P(a)\ \underline{\text{and}}\ (0\leq a\leq k-1)\ \underline{\text{and}}\ (y=a!)]\}$

The induction step proceeds as follows :

$\phi_1^k\ (\lambda x.[P(x+1)\ \underline{\text{and}}\ (x+1\neq 0)]) = \lambda(a,y).[P(a+1)\ \underline{\text{and}}\ (a+1\neq 0)\ \underline{\text{and}}\ (0\leq a\leq k-1)\ \underline{\text{and}}\ (y=a!)]$

$\phi_1^k\ (\lambda x.[P(x+1)\ \underline{\text{and}}\ (x+1\neq 0)])(x,m) = (P(x+1)\ \underline{\text{and}}\ (0\leq x\leq k-1)\ \underline{\text{and}}\ (m=x!))$

$\overline{\sigma}^3_3\ (\lambda(a,y,x).[\exists m:\phi_1^k(\lambda x.[P(x+1)\ \underline{\text{and}}\ (x+1\neq 0)])(x,m)\ \underline{\text{and}}\ P(a)\ \underline{\text{and}}\ (a\neq 0)\ \underline{\text{and}}\ (x=a-1)$
$\qquad \underline{\text{and}}\ (y=a*m)]) = \lambda(a,y).[P(a)\ \underline{\text{and}}\ (1\leq a\leq k)\ \underline{\text{and}}\ (y=a!)]$

$[\phi_1^{k+1} = \lambda P.\{\lambda(a,y).[P(a)\ \underline{\text{and}}\ (0\leq a\leq k)\ \underline{\text{and}}\ (y=a!)]$

By mathematical induction on k we have shown that ϕ_1^k is the general term of the ascending approximation sequence, so that the least fixed point is obtained by passing to the limit :

$[\lim_{k\to\infty} \phi_1^k = \phi_1^\infty = \lambda P.[P(a)\ \underline{\text{and}}\ (0\leq a)\ \underline{\text{and}}\ (y=a!)]$. *End of Example.*

Example : MacCarthy's 91-function may be defined by the following procedure with integer parameters :

```
φ1 φ2 φ3   proc f( value x∈Π ; result y∈Π ) =
              if x>100 then
                 y:=x-10;
              else
     φ4          begin new z∈Π  := x+11  ;
       φ5 φ6 φ7 φ8  begin new t∈Π  := Ω  ;
                       f(z;t);
                       f(t;y);
                    end;
                 end;
              fi;
```

The system of semantic equations associated with this procedure is :

$$
\begin{cases}
\phi_1 \in (\Pi\to\mathbb{B})\circ\!\!\to(\Pi^2\to\mathbb{B}) = \lambda P.\{\sigma^3_{1,2}(\phi_2(\lambda(a,y,x).[P(a) \text{ and } (a=x) \text{ and } (y=\Omega)]))\} \\
\phi_2 \in (\Pi^3\to\mathbb{B})\circ\!\!\to(\Pi^3\to\mathbb{B}) = \lambda P.\{\phi_3(\lambda(a,y,x).[P(a,y,x) \text{ and } (x>100)]) \text{ or } \\
\qquad\qquad \phi_4(\lambda(a,y,x).[P(a,y,x) \text{ and } (x\le 100)])\} \\
\phi_3 \in (\Pi^3\to\mathbb{B})\circ\!\!\to(\Pi^3\to\mathbb{B}) = \lambda P.\{\lambda(a,y,x).[\exists m\in\Pi : P(a,m,x) \text{ and } (y=x-10)]\} \\
\phi_4 \in (\Pi^3\to\mathbb{B})\circ\!\!\to(\Pi^3\to\mathbb{B}) = \lambda P.\{\bar\sigma^4_4(\phi_5(\lambda(a,y,x,z).[P(a,y,x) \text{ and } (z=x+11)]))\} \\
\phi_5 \in (\Pi^4\to\mathbb{B})\circ\!\!\to(\Pi^4\to\mathbb{B}) = \lambda P.\{\bar\sigma^5_5(\phi_6(\lambda(a,y,x,z,t).[P(a,y,x,z) \text{ and } (t=\Omega)]))\} \\
\phi_6 \in (\Pi^5\to\mathbb{B})\circ\!\!\to(\Pi^5\to\mathbb{B}) = \lambda P.\{\phi_8(\phi_7(P))\} \\
\phi_7 \in (\Pi^5\to\mathbb{B})\circ\!\!\to(\Pi^5\to\mathbb{B}) = \lambda P.\{\lambda(a,y,x,z,t).[\phi_1(\sigma^5_4(P))(z,t) \text{ and } \bar\sigma^5_5(P)(a,y,x,z)]\} \\
\phi_8 \in (\Pi^5\to\mathbb{B})\circ\!\!\to(\Pi^5\to\mathbb{B}) = \lambda P.\{\lambda(a,y,x,z,t).[\phi_1(\sigma^5_5(P))(t,y) \text{ and } \bar\sigma^5_2(P)(a,x,z,t)]\}
\end{cases}
$$

This system can be substantially simplified to get :

$$
\begin{cases}
\phi_1 = \lambda P.\{\lambda(a,y).[P(a) \text{ and } a>100 \text{ and } y=a-10] \text{ or } \\
\qquad \sigma^5_{1,2}(\phi_8(\lambda(a,y,x,z,t).[\phi_1(\lambda z.[P(z-11) \text{ and } (z\le 111)])(z,t) \text{ and } P(a) \text{ and } (a\le 100) \\
\qquad \text{ and } (x=a) \text{ and } (y=\Omega) \text{ and } (z=x+11)]))\} \\
\phi_8 = \lambda P.\{\lambda(a,y,x,z,t).[\phi_1(\sigma^5_5(P))(t,y) \text{ and } \bar\sigma^5_2(P)(a,x,z,t)]\}
\end{cases}
$$

We now solve by successive approximations starting from the initial approximation :

$$
\begin{bmatrix}
\phi^0_1 = \lambda P.\{\lambda(a,y).[\text{false}]\} \\
\phi^0_8 = \lambda P.\{\lambda(a,y,x,z,t).[\text{false}]\}
\end{bmatrix}
$$

$$
\left[\phi^1_1 = \lambda P.\{\lambda(a,y).[P(a) \text{ and } (a>100) \text{ and } (y=a-10)]\}\right.
$$

$$
\left[\phi^2_1 = \lambda P.\{\lambda(a,y).[P(a) \text{ and } (((a>100) \text{ and } (y=a-10)) \text{ or } ((a=100) \text{ and } (y=91)))]\}\right.
$$

After computing these first few iterates and an unsuccessfull but enlightening trial we conjectured the following induction hypothesis :

$$
\begin{bmatrix}
\phi^k_1 = \lambda P.\{\lambda(a,y).[P(a) \text{ and } \{((a>100) \text{ and } (y=a-10)) \text{ or } ((k\le 11) \text{ and } (102-k\le a\le 100) \text{ and } (y=91)) \\
\qquad \text{ or } ((k\ge 11) \text{ and } (91-11*(k-11)\le a\le 100) \text{ and } (y=91)\}]\} \\
\phi^k_8 = \lambda P.\{\lambda(a,y,x,z,t).[\phi^k_1(\sigma^5_5(P))(t,y) \text{ and } \bar\sigma^5_2(P)(a,x,z,t)]\}
\end{bmatrix}
$$

The induction step (k≥2) proceeds as follows :

$$
\phi^k_1(\lambda z.[P(z-11) \text{ and } (z\le 111)])(z,t) = [P(z-11) \text{ and } (z\le 111) \text{ and } \{((z>100) \text{ and } (t=z-10)) \\
\text{ or } ((k\le 11) \text{ and } (102-k\le z\le 100) \text{ and } (t=91)) \text{ or } ((k\ge 11) \text{ and } (91-11*(k-11)\le z\le 100) \\
\text{ and } (t=91))\}]
$$

Let Q be $\lambda(a,y,x,z,t).[\phi^k_1(\lambda z.[P(z-11) \text{ and } (z\le 111)])(z,t) \text{ and } P(a) \text{ and } (a\le 100) \text{ and } (x=a) \text{ and } (y=\Omega) \text{ and } (z=x+11)]$

By substitutions and simplifications we get :

$$
Q = \lambda(a,y,x,z,t).[P(a) \text{ and } (x=a) \text{ and } (y=\Omega) \text{ and } (z=a+11) \text{ and } \{((90\le a\le 100) \text{ and } (t=a+1)) \\
\text{ or } ((k\le 11) \text{ and } (91-k\le a\le 89) \text{ and } (t=91)) \text{ or } ((k\ge 11) \text{ and } (91-11*((k+1)-11)\le a\le 89) \\
\text{ and } (t=91))\}
$$

therefore $\sigma^5_5(Q) = \lambda t.[((91\le t\le 100) \text{ and } P(t-1)) \text{ or } (t=91)]$ so that we can evaluate the recursive call on ϕ^k_1 :

$$
\phi^k_1(\sigma^5_5(P))(t,y) = ((y=91) \text{ and } (\{P(t-1) \text{ and } \{(t=101) \text{ or } ((k\le 11) \text{ and } (102-k\le t\le 100)) \text{ or } \\
((k\ge 11) \text{ and } (91\le t\le 100))\}\} \text{ or } \{(k\ge 11) \text{ and } (t=91)\}))
$$

Now,

$$
\sigma^5_{1,2}(\phi^k_8(Q)) = \lambda(a,y).[P(a) \text{ and } (y=91) \text{ and } \{(a=100) \text{ or } ((k\le 11) \text{ and } (102-(k+1)\le a\le 99)) \\
\text{ or } ((k\ge 11) \text{ and } (90\le a\le 99)) \text{ or } ((k\ge 11) \text{ and } (a=90)) \text{ or } ((k=11) \\
\text{ and } (91-k\le a\le 89)) \text{ or } ((k\ge 11) \text{ and } (91-11*((k+1)-11)\le a\le 89))\}]
$$

therefore,

$$
\left[\phi^{k+1}_1 = \lambda P.\{\lambda(a,y).[P(a) \text{ and } \{((a>100) \text{ and } (y=a-10)) \text{ or } ((k+1\le 11) \text{ and } (102-(k+1)\le a\le 100) \\
\text{ and } (y=91)) \text{ or } ((k+1\ge 11) \text{ and } (91-11*((k+1)-11)\le a\le 100) \text{ and } (y=91))\}]\}\right.
$$

Now the solution ϕ^∞_1 is obtained by passing to the limit*:

$$
\left[\phi^\infty_1 = \lambda P.\{\lambda(a,y).[P(a) \text{ and } \{((a>100) \text{ and } (y=a-10)) \text{ or } ((a\le 100) \text{ and } (y=91))\}]\}\right.
$$

End of Example.

The deductive semantics of procedures which we have briefly sketched can be used to analyze programs by hand. Yet our interest is in *automatic* analysis of programs. Obviously a mechanized analysis cannot be as deep as the ones which can be done by hand. The essential idea is therefore to be satisfied by a mechanical discovery of *approximate* properties of programs. For example MacCarthy's 91-function might be shown to deliver a result greater or equal to 91 whenever it terminates. We now examine what is meant by approximate properties of programs and how these properties can be discovered by a mechanical resolution or approximation of the solution of a system of equations which is infered from the semantic equations.

*A justification of the chaotic iterations and of the passage to the limit may be found in Cousot[1977c].

4. MECHANIZED DISCOVERY OF APPROXIMATE PROPERTIES OF PROCEDURES

4.1. CLOSURE OPERATORS, RELATIVE TOPOLOGY, INDUCED FUNCTION SPACE AND HOMEOMORPH SPACES

4.1.1. CLOSURE OPERATORS

DEFINITION 4.1.1.1 A mapping $\rho \in L \to L$ is said to be a *closure operator* if and only if it is (i) monotone, (ii) extensive $(\forall x \in L, x \sqsubseteq \rho(x))$ and (iii) idempotent $(\rho \circ \rho = \rho)$.

THEOREM 4.1.1.2 Let $\rho \in L \to L$ be a closure operator on the complete lattice $(L, \sqsubseteq, \bot, \top, \sqcup, \sqcap, \bigsqcup, \bigsqcap)$ and let $L' = \rho(L)$ be the range of ρ. L' is the set of fixed points of ρ, it is a complete lattice $(L', \sqsubseteq', \bot', \top', \sqcup', \sqcap', \bigsqcup', \bigsqcap')$ where $\sqsubseteq = \sqsubseteq'$, $\bot' = \rho(\bot)$, $\top' = \top$, $x \sqcup' y = \rho(x \sqcup y)$, $\sqcap' = \sqcap$, $\bigsqcup' X = \rho(\bigsqcup X)$, $\bigsqcap' = \bigsqcap$.

Proof : The set $fp(\rho)$ of fixed points of ρ is a non-empty complete lattice for the ordering of L (theorem 2.10), hence $fp(\rho) \subseteq L'$. $\forall y' \in L'$, $\exists y \in L$ such that $y' = \rho(y)$ so that $\rho(y') = \rho(\rho(y)) = \rho(y) = y' \in fp(\rho)$ proving that $fp(\rho) = L'$.

Let $X \subseteq L'$, $\bigsqcap X$ exists since L is complete. $\forall x \in X$, $\bigsqcap X \sqsubseteq x$ and $\rho(\bigsqcap X) \sqsubseteq \rho(x) = x$ since ρ is monotone and $fp(\rho) = L'$. Then $\rho(\bigsqcap X)$ is a lower bound of X so that $\rho(\bigsqcap X) \sqsubseteq \bigsqcap X$. Also $\bigsqcap X \sqsubseteq \rho(\bigsqcap X)$ since ρ is extensive therefore $\rho(\bigsqcap X) = \bigsqcap X$ by antisymmetry. Hence $\bigsqcap X \in L'$ proving that $\bigsqcap' X = \bigsqcap X$.

Let $u = \rho(\bigsqcup X)$, $\bigsqcup X$ exists since L is complete. ρ is extensive, then $\bigsqcup X \sqsubseteq u$ and u is an upper bound of X in L. But $u \in L'$ so that u is also an upper bound of X in L' since $\sqsubseteq' = \sqsubseteq$. Now, let y be an upper bound of X in L, then $\bigsqcup X \sqsubseteq y$ and $u = \rho(\bigsqcup X) \sqsubseteq \rho(y)$ by monotony. If in particular $y \in L'$ then $\rho(y) = y$ and $u \sqsubseteq y$ proving that u is the least upper bound $\bigsqcup' X$ of X in L'.

Set $\bot' = \rho(\bot)$, $\forall y' \in L'$, $\exists y \in L$ such that $y' = \rho(y)$. Since ρ is monotone $\bot' = \rho(\bot) \sqsubseteq \rho(y) = y'$ proving that $\bot'$ is the infimum of L'. Since $\rho(\top) \sqsubseteq \top$ in L and ρ is extensive then by antisymmetry $\rho(\top) = \top$ proving that $\top' = \top$. *End of Proof.*

COROLLARY 4.1.1.3 A closure operator $\rho \in L \to L$ is a *quasi-complete-morphism* : $\forall S \subseteq L$, $\bigsqcap\{\rho(x) : x \in S\} = \rho(\bigsqcap\{\rho(x) : x \in S\})$ and $\rho(\bigsqcup\{x : x \in S\}) = \rho(\bigsqcup\{\rho(x) : x \in S\})$.

Proof : $\bigsqcap\{\rho(x) : x \in S\}$ is a fixed point of ρ since it belongs to $\rho(L)$. Since $\forall x \in S$, $x \sqsubseteq \rho(x)$ we have $\bigsqcup S \sqsubseteq \bigsqcup\{\rho(x) : x \in S\}$ then $\rho(\bigsqcup S) \sqsubseteq \rho(\bigsqcup\{\rho(x) : x \in S\})$. Conversely, $\forall x \in S$, $x \sqsubseteq \bigsqcup S$ hence $\rho(x) \sqsubseteq \rho(\bigsqcup S)$, $\rho(\bigsqcup S)$ is an upper bound of S so that $\bigsqcup\{\rho(x) : x \in S\} \sqsubseteq \rho(\bigsqcup S)$ and $\rho(\bigsqcup\{\rho(x) : x \in S\}) \sqsubseteq \rho(\rho(\bigsqcup S)) = \rho(\bigsqcup S)$. Antisymmetry proves the equality. *End of Proof.*

To each closure operation ρ on a complete lattice L we may associate a lattice $L' = \rho(L)$. Conversely if S is any set of elements x of L we want to determine a set L' and a closure operator ρ such that L' is the smallest subset satisfying $S \subseteq L'$ and $\rho(L) = L'$.

$\forall x \in L$, let us define the *ideal operator* ρ_x associated with x by $\rho_x = \lambda y.$ if $y \sqsubseteq x$ then x else $\top$ fi. Note that it is closure operator.

$\forall S \subseteq L$: $S \neq \emptyset$ let us define the ideal operator ρ_S associated with S by $\rho_S = \lambda y.(\bigsqcap\{\rho_x(y) : x \in S\})$.

LEMMA 4.1.1.4 $\forall S \subseteq L$, the ideal operator ρ_S is a closure operator.

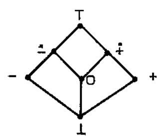
Proof : Whenever $y \sqsubseteq z$ we have $\rho_S(y) = \bigsqcap\{\rho_x(y) : x \in S\}$ and $\rho_S(z) = \bigsqcap\{\rho_x(z) : x \in S\}$. Since the ρ_x are monotone $\rho_x(y) \sqsubseteq \rho_x(z)$, $\forall x \in S$ proving that $\bigsqcap\{\rho_x(y) : x \in S\} \sqsubseteq \bigsqcap\{\rho_x(z) : x \in S\}$ and that ρ_S is monotone. $\forall x \in S$, $y \sqsubseteq \rho_x(y)$ therefore $y = \bigsqcap\{y : x \in S\} \sqsubseteq \bigsqcap\{\rho_x(y) : x \in S\} = \rho_S(y)$ proving that ρ_S is extensive. Moreover $\forall y \in S$, $\rho_S(\rho_S(y)) = \rho_S(\bigsqcap\{\rho_x(y) : x \in S\}) = \bigsqcap\{\rho_x(\bigsqcap\{\rho_x(y) : x \in S\} : x \in S)\} = \bigsqcap\{\bigsqcap\{\rho_x(y) : x \in S\} : x \in S\}$ by the quasi-linear property. Finally $\rho_S(\rho_S(y)) = \bigsqcap\{\rho_x(y) : x \in S\} = \rho_S(y)$ proving that ρ_S is idempotent. *End of Proof.*

Notice that $\rho_S = \lambda y.[\bigsqcap\{x \in (S \cup \{\top\}) : y \sqsubseteq x\}]$.

THEOREM 4.1.1.5 Let S be a subset of L, ρ_S the associated ideal operator and $L' = \rho_S(L)$. L' is the smallest image of L by a closure operator containing S.

Proof : Let θ be a closure operator such that $S \subseteq \theta(L)$. Let us show that $L' \subseteq \theta(L)$ that is $\{y' \in L'\} \Rightarrow \{y' \in \theta(L)\}$. $\forall y' \in L'$, $\exists y \in L$: $y' = \rho_S(y) = \bigsqcap\{$if $y \sqsubseteq x'$ then x' else $\top$ fi : $x' \in S\}$. Let R be the set $\{x : \theta(x) \in S\}$. $y' = \bigsqcap\{$if $y \sqsubseteq \theta(x)$ then $\theta(x)$ else $\theta(\top)$ fi : $x \in R\}$ so that y' is the meet of elements of $\theta(L)$ which according to theorem 4.1.1.2 proves that $y' \in \theta(L)$. *End of Proof.*

Example 4.1.1.6 : The following lattice can be used to determine the sign of integer variables of a program :

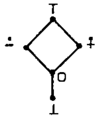
L = 

Assume that we are interested only in knowing whether a variable is uninitialized $(\bot)$, positive or zero $(\underline{+})$ or negative or zero $(\underline{-})$ but not interested in the fact that a variable is strictly negative $(-)$, strictly positive $(+)$ or zero. We choose $S = \{\underline{+}, \underline{-}, \bot\}$ and determine the set L' of approximate properties containing S :

x	$\bot$	$\dot{-}$	$-$	0	$+$	$\dot{+}$	$\top$
$\rho_S(x)$	$\bot$	$\dot{-}$	$\dot{-}$	0	$\dot{+}$	$\dot{+}$	$\top$

hence for $S=\{\dot{+},\dot{-},\bot\}$ we have :

$\rho_S(L)$ =



End of Example.

Remark : Let ρ be a closure operator on L. $L'=\rho(L)$ is a sublattice of L if and only if ρ is a *complete join morphism* : $\rho(\bigsqcup\{x:x\in S\}) = \bigsqcup\{\rho(x):x\in S\}$, (Ward[1942], p.193). *End of Remark.*

L' can also be characterized by a *complete join congruence relation* θ of L (that is an equivalence relation satisfying the *join-substitution property* : $\{\forall R,S\subseteq L: \{\{\forall x\in R,\exists y\in S:x\equiv y(\theta)\}$ and $\{\forall y\in S,\exists x\in R:y\equiv x(\theta)\}\} \Rightarrow \{\bigsqcup R\equiv\bigsqcup S(\theta)\}\}$.

For $x\in L$ we write $[x]\theta$ the congruence class containing x that is $[x]\theta = \{y\in L:x\equiv y(\theta)\}$.

LEMMA 4.1.1.7 $\forall x\in L$, $[x]\theta$ is a complete join-sublattice of L which is *convex* : if $y,z\in[x]\theta$, $t\in L$ and $y\sqsubseteq t\sqsubseteq z$ then $t\in[x]\theta$.

Proof : $\forall X\subseteq[x]\theta$, $\bigsqcup X$ exists in L and belongs to $[x]\theta$ since $\{\forall y\in X, y\equiv x(\theta)\} \Rightarrow \{\bigsqcup X\equiv x(\theta)\}$ proving that $[x]\theta$ is a complete join-sublattice of L. Moreover if $y,z\in[x]\theta$ then $y\equiv x(\theta)$ and $z\equiv x(\theta)$. if $y\sqsubseteq t\sqsubseteq z$ then $t=t\sqcup y \equiv t\sqcup x(\theta)$ and $y=t\sqcup y \equiv t\sqcup x(\theta)$ which imply by transitivity and symmetry that $t\equiv y(\theta)$. Besides $y\equiv x(\theta)$ then by transitivity $t\equiv x(\theta)$ proving that $[x]\theta$ is convex. *End of Proof.*

THEOREM 4.1.1.8 Let $\rho \in L\to L$ be defined by $\rho=\lambda x.\bigsqcup[x]\theta$. ρ is a closure operation.

Proof : Since $x\in[x]\theta$ we have $x\sqsubseteq\bigsqcup[x]\theta=\rho(x)$ proving that ρ is extensive. According to lemma 4.1.1.7 $\bigsqcup[x]\theta\in[x]\theta$ which implies that $\rho(\rho(x)) = \rho(\bigsqcup[x]\theta) = \bigsqcup[\bigsqcup[x]\theta]\theta = \bigsqcup[x]\theta = \rho(x)$ proving that ρ is idempotent. If $x\sqsubseteq y$ then $y=x\sqcup y \equiv (\rho(x)\sqcup\rho(y))(\theta)$ by the join substitution property. By transitivity $\rho(y)\equiv(\rho(x)\sqcup\rho(y))(\theta)$ so that $(\rho(x)\sqcup\rho(y))\in[\rho(y)]\theta$ and therefore $(\rho(x)\sqcup\rho(y)) \sqsubseteq \bigsqcup[\rho(y)]\theta=\rho(y)$. Besides it is obvious that $\rho(y)\sqsubseteq\rho(x)\sqcup\rho(y)$ hence by antisymmetry $\rho(y)=\rho(x)\sqcup\rho(y)$ then $\rho(x)\sqsubseteq\rho(y)$ proving that ρ is monotone. *End of Proof.*

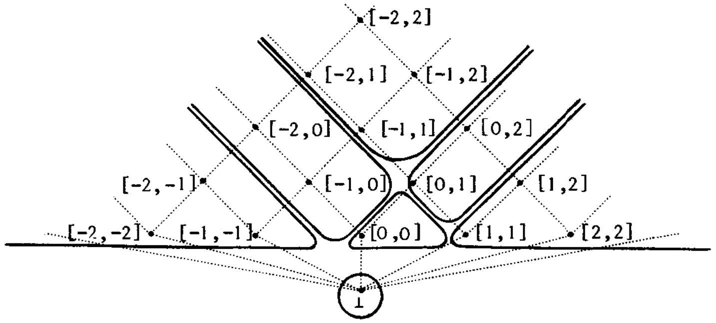
ρ can be characterized by a complete join congruence relation θ of L, but conversely a given ρ defines a complete join congruence relation (ρ) of L :

THEOREM 4.1.1.9 Let ρ be a closure operator on L. Let (ρ) be the relation defined by $\{x\equiv y(\rho)\} \Leftrightarrow \{\rho(x)=\rho(y)\}$, (ρ) is a complete join congruence relation.

Proof : Since equality is reflexive, symmetric and transitive (ρ) is an equivalence relation. $\forall R,S\subseteq L$: $\{\forall x\in R,\exists y\in S:\rho(x)=\rho(y)\}$ and $\{\forall y\in S,\exists x\in R:\rho(y)=\rho(x)\}$, $\rho(\bigsqcup R)=\rho(\bigsqcup\{\rho(x):x\in R\}) = \rho(\bigsqcup\{\rho(y):y\in S\}) = \rho(\bigsqcup S)$ since ρ is a quasi join complete morphism. *End of Proof.*

Example 4.1.1.10 Let L be the complete lattice of intervals $[a,b]$ of $\mathbb{N}\cup\{-\infty,+\infty\}$. The following schema gives the join congruence classes defining ρ :

L =



We have $\rho(\bot)=\bot$, $\rho([0,0])=[0,0]$, whenever $n>0$, we have $\rho([0,n])=[0,+\infty]$, $\rho([-n,0])=[-\infty,0]$ whenever $0<n\leq m$ we have $\rho([n,m])=[1,+\infty]$, $\rho([-m,-n])=[-\infty,1]$ and finally whenever $n>0$ and $m>0$ we have $\rho([-m, n])=[-\infty,+\infty]$ so that $\rho(L)$ is isomorphic with the lattice of example 4.1.1.6. *End of Example.*

4.1.2. THE LATTICE OF CLOSURE OPERATORS

Given a set R of closure operators ρ on L, we have defined the meet $\sqcap R$ by $\{\forall x\in L,(\sqcap R)(x)=\sqcap\{\rho(x):\rho\in R\}\}$. The induced partial ordering is $\{\rho\leq\eta\} \Leftrightarrow \{\forall x\in L,\rho(x)\sqsubseteq\eta(x)\}$.

THEOREM 4.1.2.1 The set $\mathbb{R}$ of closure operators on a complete lattice L form a complete lattice $(\mathbb{R},\leq,\downarrow,\uparrow,\bigsqcup,\sqcap)$.

Proof : The lemma 4.1.1.4 can be easily adapted to prove that the meet of a set of closure operators exists and is a closure operator, therefore $\mathbb{R}$ is a complete meet semilattice. The operator $\uparrow=\lambda x.\top$ is a closure operator supremum of $\mathbb{R}$, so that $\mathbb{R}$ is a complete lattice with $\bigsqcup\{\rho:\rho\in R\} = \sqcap\{\eta\in\mathbb{R}:\rho\leq\eta,\forall\rho\in R\}$ and the infimum $\downarrow$ being the identity function. *End of Proof.*

LEMMA 4.1.2.2 $\forall \rho,\eta \in \mathbb{R}$, $\{\rho \sqsubseteq \eta\} \Leftrightarrow \{\eta(L) \subseteq \rho(L)\}$

Proof : Assume $\rho \sqsubseteq \eta$ and let $x \in \eta(L)$, then $\eta(x)=x$ and $\rho(x) \sqsubseteq \eta(x)$ hence $\rho(x) \sqsubseteq x$. Therefore $x=\rho(x)$ since ρ is extensive proving that $x \in \rho(L)$ and $\eta(L) \subseteq \rho(L)$. Reciprocally, ρ and η are the ideal operators of the respective $\rho(L)$ and $\eta(L)$: $\rho=\lambda y.[\sqcap\{x \in \rho(L):y \sqsubseteq x\}]$ and $\eta=\lambda y.[\sqcap\{x \in \eta(L):y \sqsubseteq x\}]$. Since $\eta(L) \subseteq \rho(L)$ we have $\forall y \in L$, $\{x \in \eta(L):y \sqsubseteq x\} \subseteq \{x \in \rho(L):y \sqsubseteq x\}$ proving that $\sqcap\{x \in \rho(L):y \sqsubseteq x\} \sqsubseteq \sqcap\{x \in \eta(L):y \sqsubseteq x\}$ and $\rho \sqsubseteq \eta$. *End of Proof.*

LEMMA 4.1.2.3 For any closure operator ρ on L : $\forall x,y \in L$, $\{x \sqsubseteq \rho(y)\} \Leftrightarrow \{\rho(x) \sqsubseteq \rho(y)\}$

Proof : This is Morgado[1962]'s characterization of the closure operators by means of one axiom. *End of Proof.*

The characterization of $\dot{\sqcup}$ by theorem 4.1.2.1 is not constructive, therefore in practice we use :

THEOREM 4.1.2.4 Let $R \subseteq \mathbb{R}$, $S=\cup\{\rho(L):\rho \in R\}$ then $\dot{\sqcap}R$ is the ideal operator ρ_S. Let $S'=\cap\{\rho(L):\rho \in R\}$, then $\dot{\sqcup}R$ is the ideal operator $\rho_{S'}$ and $\rho_{S'}(L)=S'$.

Proof : We have $S=\{x' \in \rho(L):\rho \in R\} = \{\rho(x):(x \in L) \text{ and } (\rho \in R)\}$ hence $\rho_S=\lambda y.\sqcap\{x' \in S:y \sqsubseteq x'\}$ $= \lambda y.\sqcap\{\rho(x):(x \in L) \text{ and } (y \sqsubseteq \rho(x)) \text{ and } (\rho \in R)\}$, besides $\dot{\sqcap}R=\lambda y.\sqcap\{\rho(y):\rho \in R\}$. But $\forall y \in L$, $\rho(y) \in \{\rho(x):(x \in L) \text{ and } (y \sqsubseteq \rho(x))\}$ hence $\{\rho(y):\rho \in R\} \subseteq \{\rho(x):(x \in L) \text{ and } (y \sqsubseteq \rho(x)) \text{ and } (\rho \in R)\}$ so that $\sqcap\{x \in S:y \sqsubseteq x\} \sqsubseteq \sqcap\{\rho(y):\rho \in R\}$ proving that $\rho_S \sqsubseteq \dot{\sqcap}R$.

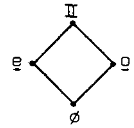
Conversely, $\{x' \in \rho(L):(y \sqsubseteq x') \text{ and } (\rho \in R)\} \subseteq \{x' \in \rho(L):(\rho(y) \sqsubseteq \rho(x')) \text{ and } (\rho \in R)\}$ by monotony, hence $\sqcap\{\rho(y):\rho \in R\} \sqsubseteq \sqcap\{\rho(x):(x \in L) \text{ and } (\rho(y) \sqsubseteq \rho(\rho(x))) \text{ and } (\rho \in R)\}$ $\sqsubseteq \sqcap\{\rho(x):(x \in L) \text{ and } (y \sqsubseteq \rho(x)) \text{ and } (\rho \in R)\}$ proving that $\dot{\sqcap}R \sqsubseteq \rho_S$ and $\rho_S=\dot{\sqcap}R$ by antisymmetry.

Since $\forall X \subseteq S'$, $\{x \in X\} \Rightarrow \{\rho(x)=x\}$, $\forall \rho \in R$ we have $\rho(\sqcap X) = \rho(\sqcap\{\rho(x):x \in X\}) = \sqcap\{\rho(x):x \in X\}$ since ρ is a quasi complete meet morphism. Hence $\rho(\sqcap X) = \sqcap\{\rho(x):x \in X\} = \sqcap X$ for any $\rho \in R$ proving that $\sqcap X \in S'$. $\forall y \in L$, $\rho_{S'}(y) = \sqcap\{x \in S':y \sqsubseteq x\} \in S'$ proving that $\rho_{S'}(L) \subseteq S'$ also $S' \subseteq \rho_{S'}(L)$ (theorem 4.1.1.5) then $\rho_{S'}(L)=S'$. Since $\forall \rho \in R$, $S' \subseteq \rho(L)$ we have $\rho_{S'}(L) \subseteq \rho(L)$ and lemma 4.1.2.2 implies that $\rho \sqsubseteq \rho_{S'}$, therefore $\rho_{S'}$ is an upper bound of R. Let η be another upper bound of R : $\forall \rho \in R$, $\{\rho \sqsubseteq \eta\} \Rightarrow \{\eta(L) \subseteq \rho(L)\} \Rightarrow$ $\{\eta(L) \subseteq \cap\{\rho(L):\rho \in R\} \Rightarrow \{\forall y \in L, \{z \in \eta(L):y \sqsubseteq z\} \subseteq \{z \in \cap\{\rho(L):\rho \in R\}:y \sqsubseteq z\}\}$. But $z \in \eta(L)$ implies $\eta(z)=z$ (lemma 4.1.1.2) and $y \sqsubseteq \eta(z)$ is equivalent to $\eta(y) \sqsubseteq \eta(z)$ (Lemma 4.1.2.3). Therefore $\forall y \in L$, $\eta(y) \in \{z \in \eta(L):\eta(y) \sqsubseteq \eta(z)\}$ so that $\eta(y) \in \{z \in \cap\{\rho(L):\rho \in R\}:y \sqsubseteq z\}$ proving that $\rho_{S'} \sqsubseteq \eta$, $\rho_{S'}$ is the least upper bound $\dot{\sqcup}R$ of R. *End of Proof.*

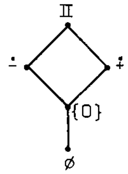
From lemma 4.1.2.2 and theorem 4.1.2.4 the complete lattice $(\mathbb{R},\sqsubseteq,\downarrow,\uparrow,\dot{\sqcup},\dot{\sqcap})$ of closure operators of L is isomorphic with the complete lattice $(\mathbb{L}',\supseteq,L,\{\top\},\cap,\lambda S.[\rho_{US}(L)])$ which is the set of images L' of L by the closure operators belonging to $\mathbb{R}$.

Example : Let $\mathbb{I}$ be the set of integers, $L=2^{\mathbb{I}}$. We denote by $\underline{e}$ the set of integers divisible by two, $\underline{o}=\mathbb{I}-\underline{e}$. We denote by $\dot{+}$ the set of positive or zero integers and by $\dot{-}$ the set of negative or zero integers. We define two closure operators ρ and η on L as follows :

$$\rho = \lambda X.\underline{\text{case}}\ \begin{array}{ll} \{X=\emptyset\} & \to \emptyset; \\ \{\forall x \in X, \underline{\text{even}}(x)\} & \to \underline{e}; \\ \{\forall x \in X, \underline{\text{odd}}(x)\} & \to \underline{o}; \\ \underline{\text{otherwise}} & \to \mathbb{I}; \end{array}\ \underline{\text{esac}};$$

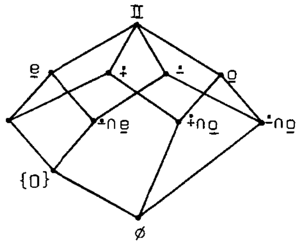
$\rho(L) =$ 

$$\eta = \lambda X.\underline{\text{case}}\ \begin{array}{ll} \{X=\emptyset\} & \to \emptyset; \\ \{X=\{0\}\} & \to \{0\}; \\ \{\forall x \in X, x \geq 0\} & \to \dot{+}; \\ \{\forall x \in X, x \leq 0\} & \to \dot{-}; \\ \underline{\text{otherwise}} & \to \mathbb{I}; \end{array}\ \underline{\text{esac}};$$

$\eta(L) =$ 

The meet and join of ρ and η are defined as follows :

$$\rho\dot{\sqcap}\eta = \lambda X.\underline{\text{case}}\ \begin{array}{ll} \{X=\emptyset\} & \to \emptyset; \\ \{X=\{0\}\} & \to \{0\}; \\ \{\forall x \in X, x \geq 0 \text{ and } \underline{\text{even}}(x)\} & \to \dot{+}\underline{e}; \\ \{\forall x \in X, x \geq 0 \text{ and } \underline{\text{odd}}(x)\} & \to \dot{+}\underline{o}; \\ \{\forall x \in X, x \leq 0 \text{ and } \underline{\text{even}}(x)\} & \to \dot{-}\underline{e}; \\ \{\forall x \in X, x \leq 0 \text{ and } \underline{\text{odd}}(x)\} & \to \dot{-}\underline{o}; \\ \{\forall x \in X, x \geq 0\} & \to \dot{+}; \\ \{\forall x \in X, x \leq 0\} & \to \dot{-}; \\ \{\forall x \in X, \underline{\text{even}}(x)\} & \to \underline{e}; \\ \{\forall x \in X, \underline{\text{odd}}(x)\} & \to \underline{o}; \\ \underline{\text{otherwise}} & \to \mathbb{I}; \end{array}\ \underline{\text{esac}};$$

$\rho\dot{\sqcap}\eta(L) =$ 

Notice the property : $\rho=\rho\circ(\rho\dot{\sqcap}\eta)$ and $\eta=\eta\circ(\rho\dot{\sqcap}\eta)$

$$\rho\dot{\sqcup}\eta = \lambda X.\underline{\text{case}}\ \begin{array}{ll} X=\emptyset & \to \emptyset; \\ \underline{\text{otherwise}} & \to \mathbb{I}; \end{array}\ \underline{\text{esac}};$$

$\rho\dot{\sqcup}\eta(L) =$ 

End of Example.

4.1.3. RELATIVE TOPOLOGY OF $L'=\rho(L)$

Let L' be the image of L by a closure operator ρ. It is a complete lattice $(L',\sqsubseteq',\bot',\top',\sqcup',\sqcap',\bigsqcup',\bigsqcap')$ (theorem 4.1.1.2) which can be considered as a $\bigsqcup'$-topological lattice (§2). But L' is a subset of L which is a $\bigsqcup$-topological lattice so that we can also consider the topology of L' which is the *relativization* of the topology of L. L' is a *subspace* of L if and only if the two topologies of L' coincide.

Let us recall (Kelley[1961], p.51) that given two topological spaces (X,T) and (X',T') with $X'\subseteq X$, T' is the relativization of T means that a set U' is open in T' if and only if there exists an open set U in T such that $U'=U\cap X'$.

THEOREM 4.1.3.1 Let L be a $\bigsqcup$-topological lattice and ρ a closure operator on L, then $L'=\rho(L)$ is a subspace of L.

Proof : Let B be an open set in the base $\mathbb{B}$ of the $\bigsqcup$-topological lattice L. The relativized open base of L' is $\mathbb{B}'=\{B\cap L' : B\in\mathbb{B}\}$. $\forall B'\in\mathbb{B}$, $\exists B : B'=\{x\in B : x=\rho(x)\}$. If $x\in B'$, $y\in L'$ and $x\sqsubseteq' y$ then $x\in B$ and $x\sqsubseteq y$ hence $y\in B$ and $y=\rho(y)$ proving that $y\in B'$. Let $x,y\in B'$ we have $x,y\in B$ hence $(x\sqcap y)\in B$. But $x\sqcap y = x\sqcap' y$ (theorem 4.1.1.2) therefore $x\sqcap' y = \rho(x)\sqcap'\rho(y) = \rho(\rho(x)\sqcap'\rho(y)) = \rho(x\sqcap' y)$ proving that $(x\sqcap' y)\in B'$ and that B' is a dual ideal of L'.

Let $[\![x_\delta : \delta\in\Delta]\!]$ be a net in L' such that $\bigsqcup'_{\Delta} x_\delta\in B'$. We know that $\exists B : \bigsqcup'_{\Delta} x_\delta\in B$ so that $\exists\delta_0\in\Delta : [\![x_\delta : (\delta\in\Delta)\ \underline{\text{and}}\ (\delta\geq\delta_0)]\!]$ is in B and in B' since $\forall\delta\in\Delta, x_\delta\in L'$. $\forall B\in\mathbb{B}$, $B'=B\cap L'$ is an open set in the base of the $\bigsqcup'$-topological lattice L'.

Reciprocally let B' be an open set in the base of L'. Let us define $\rho^{-1}(B') = \{x\in L : \rho(x)\in B'\}$. If $x\in\rho^{-1}(B')$ and $x\sqsubseteq y$ then $\rho(x)\in B'$ and $\rho(x)\sqsubseteq\rho(y)$ hence $\rho(y)\in B'$ and $y\in\rho^{-1}(B')$. Moreover if $(\bigsqcup_{\Delta} x_\delta)\in\rho^{-1}(B')$ then $\rho(\bigsqcup_{\Delta} x_\delta)=\bigsqcup'_{\Delta} x_\delta\in B'$. Hence $\exists\delta_0\in\Delta$ such that $[\![x_\delta : (\delta\geq\delta_0)\ \underline{\text{and}}\ (\delta\in\Delta)]\!]$ is in $B'\subseteq\rho^{-1}(B')$ proving that $[\![x_\delta : \delta\in\Delta]\!]$ is eventually in $\rho^{-1}(B')$. $\rho^{-1}(B')$ is an open set of L such that $B'=\rho^{-1}(B')\cap L'$. Let now U' be an open set of L'. $U'=\bigcup_{i\in\Delta} B'_i$ where $\forall i\in\Delta$, $B'_i=\rho^{-1}(B'_i)\cap L'$ is an open set of L. Hence $U'=\bigcup_{i\in\Delta}(\rho^{-1}(B'_i)\cap L') = (\bigcup_{i\in\Delta}\rho^{-1}(B'_i))\cap L'$ since 2^L is a distributive lattice. But $U=\bigcup_{i\in\Delta}\rho^{-1}(B'_i)$ is an open set of L, proving that for any open set U' there exists an open set U of L such that $U'=U\cap L'$. *End of Proof.*

Let $f\in X\to Y'$ a function on X into Y', $X'\subseteq X$ and $Y'\subseteq Y$ we denote by $(f|X')$ the *restriction* of f to X' and by $(Y|f)$ the *injection* of f into Y. If X' is a topological subspace of X, Y' is a topological subspace of Y and f is continuous then $(f|X')$ and $(Y|f)$ are continuous functions. (Schwartz[1970], p.42).

COROLLARY 4.1.3.2 Any closure operator on L is continuous.

Proof : $\rho\in L\to\rho(L)$ is continuous (proof of theorem 4.1.3.2) hence $(L|\rho)$ is continuous. *End of Proof.*

4.1.4. INDUCED FUNCTION SPACE

Let L be a $\bigsqcup$-topological space and $\rho\in L\overset{c}{\to}L$ a closure operation. Let us define $\bar\rho\in(L\overset{c}{\to}L)\to(L\overset{c}{\to}L)$ by $\bar\rho=\lambda f.\rho\circ f\circ\rho$. (Notice that whenever f is continuous $\rho\circ f\circ\rho$ is continuous (4.1.3.2)).

LEMMA 4.1.4.1 $\bar\rho$ is a closure operation on $L\overset{c}{\to}L$.

Proof : Whenever $f\sqsubseteq g$ we have $\rho\circ f\circ\rho\sqsubseteq\rho\circ g\circ\rho$ since ρ is monotone proving that $\bar\rho$ is monotone. Also $f\sqsubseteq\bar\rho(f)$ since $\forall x\in L$, $f(x)\sqsubseteq\rho(f(\rho(x)))$ since ρ is extensive and f monotone. Finally $\bar\rho(\bar\rho(f)) = \bar\rho(\rho\circ f\circ\rho) = \rho\circ\rho\circ f\circ\rho\circ\rho = \rho\circ f\circ\rho = \bar\rho(f)$ proving that $\bar\rho$ is idempotent. *End of Proof.*

COROLLARY 4.1.4.2 $\bar\rho(L\overset{c}{\to}L)$ is a $\bigsqcup$-topological subspace of $L\overset{c}{\to}L$.

4.1.5. INDUCED FUNCTIONALS

LEMMA 4.1.5.1 Let $F\in(L\overset{c}{\to}L)\overset{c}{\to}(L\overset{c}{\to}L)$ be a continuous functional, then $lfp(F)\sqsubseteq lfp(\bar\rho\circ F\circ\bar\rho)$.

Proof : According to theorem 2.10, $lfp(F)$ exists since F is monotone (theorem 2.9) and $L\overset{c}{\to}L$ is a complete lattice (2.12). Also $f=lfp(\bar\rho\circ F\circ\bar\rho)$ exists since $\bar\rho$ is a closure operator on $L\overset{c}{\to}L$. Since $\bar\rho$ is extensive and $\bar\rho\circ F$ monotone we have $F(f)\sqsubseteq\bar\rho\circ F(f)\sqsubseteq\bar\rho\circ F\circ\bar\rho(f)=f$ proving that f is a post fixed point of F which implies that $lfp(F)\sqsubseteq f$, (theorem 2.10). *End of Proof.*

4.1.6. HOMEOMORPH SPACES L^* OF $\rho(L)$

Let L be a $\bigsqcup$-topological lattice, ρ a closure operator on L and $L'=\rho(L)$. Let L^* be the image of L' by a bijection β (one-to-one and onto map). Let us define two operations on L^{*2} by $x\sqcup_* y = \beta(\beta^{-1}(x)\sqcup'\beta^{-1}(y))$ and $x\overset{*}{\sqcap} y = \beta(\beta^{-1}(x)\sqcup'\beta^{-1}(y))$.

THEOREM 4.1.6.1 $(L^*,{}^*\sqsubseteq,\bot_*,\overset{*}{\top},\sqcup_*,\overset{*}{\sqcap},\bigsqcup_*,\overset{*}{\bigsqcap})$ is a complete lattice *isomorphic* with $(L',\sqsubseteq',\bot',\top',\sqcup',\sqcap',\bigsqcup',\bigsqcap')$ by β.

LEMMA 4.1.6.2 The one-to-one map β of the $\sqcup'$-topological space L' onto the $\underset{*}{\sqcup}$- topological space L^* and its inverse β^{-1} are continuous.

THEOREM 4.1.6.3 The $\sqcup'$-topological lattice L' and the $\underset{*}{\sqcup}$-topological lattice L^* are *homeomorphic* and β is the corresponding homeomorphism.

LEMMA 4.1.6.4 Let $F' \in L' \overset{o}{\to} L'$ and $F^* \in L^* \overset{o}{\to} L^*$ be defined by $F^* = \lambda f^*.[\beta \circ F'(\beta^{-1} \circ f^* \circ \beta) \circ \beta^{-1}]$. Then $\mathit{flp}(F^*) = \beta \circ \mathit{lfp}(F') \circ \beta^{-1}$.

Proof : $\beta \circ \mathit{lfp}(F') \circ \beta^{-1}$ is a fixed point of F^* since $F^*(\beta \circ \mathit{lfp}(F') \circ \beta^{-1}) = \beta \circ F'(\beta^{-1} \circ \beta \circ \mathit{lfp}(F') \circ \beta \circ \beta^{-1}) \circ \beta = \beta \circ \mathit{lfp}(F') \circ \beta^{-1}$. It is the least fixed point since whenever $f^* = F^*(f^*)$ we have $\beta^{-1} \circ f^* \circ \beta = F'(\beta^{-1} \circ f^* \circ \beta)$ hence $\mathit{lfp}(F') \leq \beta^{-1} \circ f^* \circ \beta$ that is $\beta \circ \mathit{lfp}(F') \circ \beta^{-1} \leq f^*$. *End of Proof.*

4.1.7. CORRESPONDENCE BETWEEN L AND L^*

A correspondence can be established between L and L^*, $L \overset{o}{\to} L$ and $L^* \overset{o}{\to} L^*$ and $(L \overset{o}{\to} L) \overset{o}{\to} (L \overset{o}{\to} L)$ and $(L^* \overset{o}{\to} L^*) \overset{o}{\to} (L^* \overset{o}{\to} L^*)$ as follows :

$\alpha \in L \overset{o}{\to} L^* = \beta \circ \rho$ (α is surjective and continuous)

$\gamma \in L^* \overset{o}{\to} L = \rho \circ \beta^{-1} = \beta^{-1}$ (γ is injective and continuous)

$\bar{\alpha} \in (L \overset{o}{\to} L) \overset{o}{\to} (L^* \overset{o}{\to} L^*)$

$\bar{\alpha} = \lambda f.[\beta \circ \bar{\rho}(f) \circ \beta^{-1}] = \lambda f.[\alpha \circ f \circ \gamma]$

$\bar{\gamma} \in (L^* \overset{o}{\to} L^*) \overset{o}{\to} (L \overset{o}{\to} L)$

$\bar{\gamma} = \lambda f^*.[\bar{\rho}(\beta^{-1} \circ f^* \circ \beta)] = \lambda f^*.[\gamma \circ f^* \circ \alpha]$

$\bar{\bar{\alpha}} \in ((L \overset{o}{\to} L) \overset{o}{\to} (L \overset{o}{\to} L)) \overset{o}{\to} ((L^* \overset{o}{\to} L^*) \overset{o}{\to} (L^* \overset{o}{\to} L^*))$

$\bar{\bar{\alpha}} = \lambda F.\{\lambda f^*.[\beta \circ \rho \circ F(\beta^{-1} \circ f^* \circ \beta \circ \rho) \circ \beta^{-1}]\}$

$= \lambda F.\{\lambda f^*.[\beta \circ (\bar{\rho} \circ F \circ \bar{\rho})(\beta^{-1} \circ f^* \circ \beta) \circ \beta^{-1}]\}$

$= \lambda F.\{\lambda f^*.[\bar{\alpha}(F(\bar{\gamma}(f^*)))]$

$\bar{\bar{\gamma}} \in ((L^* \overset{o}{\to} L^*) \overset{o}{\to} (L^* \overset{o}{\to} L^*)) \overset{o}{\to} ((L \overset{o}{\to} L) \overset{o}{\to} (L \overset{o}{\to} L))$

$\bar{\bar{\gamma}} = \lambda F^*.\{\lambda f.[\bar{\gamma}(F^*(\bar{\alpha}(f)))]$.

THEOREM 4.1.7.1 $\forall F \in ((L \overset{o}{\to} L) \overset{o}{\to} (L \overset{o}{\to} L))$, $\mathit{lfp}(F) \leq \bar{\gamma}(\mathit{lfp}(\bar{\bar{\alpha}}(F)))$

Proof : The existence of the least fixed points is guaranteed by theorem 2.10. Let F' be $(\bar{\rho} \circ F \circ \bar{\rho})$, then $\bar{\bar{\alpha}}(F) = \lambda f^*.[\beta \circ F'(\beta^{-1} \circ f^* \circ \beta) \circ \beta^{-1}]$). We know that $\mathit{lfp}(\bar{\bar{\alpha}}(F)) = \beta \circ \mathit{lfp}(F') \circ \beta^{-1} = \bar{\gamma}(\mathit{lfp}(F'))$ (lemma 4.1.6.4) that $\mathit{lfp}(F) \leq \mathit{lfp}(F')$ (lemma 4.1.5.1) and that $\bar{\gamma}$ is monotone, therefore $\mathit{lfp}(F) \leq \bar{\gamma}(\mathit{lfp}(\bar{\bar{\alpha}}(F)))$. *End of Proof.*

COROLLARY 4.1.7.2 Let $F^* \in (L^* \overset{o}{\to} L^*) \overset{o}{\to} (L^* \overset{o}{\to} L^*)$ such that $\bar{\bar{\alpha}}(F) \leq F^*$, then $\mathit{lfp}(F) \leq \bar{\gamma}(\mathit{lfp}(F^*))$.

Proof : Immediate from lemma 2.13 and theorem 4.1.7.1. *End of Proof.*

COROLLARY 4.1.7.3 Let F, G $\in (L \overset{o}{\to} L) \overset{o}{\to} (L \overset{o}{\to} L)$ then $\mathit{lfp}(F \circ G) \leq \bar{\gamma}(\mathit{lfp}(\bar{\bar{\alpha}}(F) \circ \bar{\bar{\alpha}}(G)))$.

Proof : Immediate since $\bar{\bar{\alpha}}(F \circ G) = \lambda f^*.[\bar{\alpha}(F(G(\bar{\gamma}(f^*))))] = \lambda f^*.[\bar{\alpha}(F(\bar{\gamma}(\bar{\alpha}(G(\bar{\gamma}(f^*))))))]$ $= \bar{\bar{\alpha}}(F) \circ \bar{\bar{\alpha}}(G)$. *End of Proof.*

COROLLARY 4.1.7.3 Let F, G $\in (L \overset{o}{\to} L) \overset{o}{\to} (L \overset{o}{\to} L)$ and $F^*, G^* \in (L^* \overset{o}{\to} L^*) \overset{o}{\to} (L^* \overset{o}{\to} L^*)$ such that $\bar{\bar{\alpha}}(F) \leq F^*$ and $\bar{\bar{\alpha}}(G) \leq G^*$ then $\mathit{lfp}(F \circ G) \leq \bar{\gamma}(\mathit{lfp}(F^* \circ G^*))$.

Proof : Immediate since $\bar{\bar{\alpha}}(G) \leq G^*$ and $\bar{\bar{\alpha}}(F)$ monotone we have $\bar{\bar{\alpha}}(F \circ G) = \bar{\bar{\alpha}}(F) \circ \bar{\bar{\alpha}}(G) \leq \bar{\bar{\alpha}}(F) \circ G^* \leq F^* \circ G^*$. *End of Proof.*

4.2. DISCOVERY OF APPROXIMATE PROPERTIES OF PROCEDURES

We can now specify what we mean by "approximate properties of programs". Given the set L of semantic properties of a program that is the set of predicates on the variables of that program, a set of approximate properties of that program will be the image of the semantic properties L by a closure operation ρ. The idea is that whenever P is true at some program point then $\rho(P)$ is also true since ρ is extensive that is $P \Rightarrow \rho(P)$. Also ρ must be monotone since it must preserve implication : $\{P \Rightarrow P'\} \Rightarrow \{\rho(P) \Rightarrow \rho(P')\}$. Finally an approximate property $\rho(P)$ approximates itself $\rho(\rho(P)) = \rho(P)$. A closure operation formalizes the idea of approximation in the space L of semantic properties. We have given several characterizations of closure operations (4.1.1.1, 4.1.1.5, 4.1.1.9) which can be used in practice to define interesting approximate properties. Also, we are able to combine different classes of approximate properties (4.1.2). In order to represent the space of approximate properties $\rho(L)$ in a machine we use an homeomorphic space L^* (4.1.6) the elements of which can be represented inside a computer. Given a concrete semantic property P its abstract representation $\alpha(P)$ in L^* is given by the *abstraction function* α. Conversely the image $\gamma(P^*)$ of an abstract property P^* of L^* is given by the *concretization function* γ. The correspondence between concrete predicate transformers $\phi \in (L \overset{o}{\to} L)$ and abstract properties transformers $\phi^* \in (L^* \overset{o}{\to} L^*)$ is given by

$\bar\alpha(\phi)$ and $\bar\gamma(\phi^*)$. Finally, the meaning of a procedure which is given by the least fixed point of a functional F (3.1,3.2) can be approximated by the least fixed point of $\bar{\bar\alpha}(F)$ which is an approximate system of equations in the space $(L^* \circ\!\!\rightarrow L^*) \circ\!\!\rightarrow (L^* \circ\!\!\rightarrow L^*)$. Indeed given a predicate P which holds on entry of the procedure π we know that $lfp(F)(P)$ holds on exit. Since the compiler cannot compute $lfp(F)$ it approximates it by $\bar\gamma(lfp(\bar{\bar\alpha}(F)))$ and will assume instead that $\bar\gamma(lfp(\bar{\bar\alpha}(F)))(P)$ holds on exit of the procedure. This is a correct approximation since $lfp(F)(P) \Rightarrow \bar\gamma(lfp(\bar{\bar\alpha}(F)))(P)$ (4.1.7.1). However the least fixed point of $\bar{\bar\alpha}(F)$ must be machine computable, which is not the case in general since $\bar{\bar\alpha}(F)$ explicitely uses F which works on non machine representable objects. Yet we can in turn approximate $\bar{\bar\alpha}(F)$ by an approximate abstract system of equations F^* constructed so that $\bar{\bar\alpha}(F) \leqslant F^*$ (4.1.7.2). Then 4.1.7.3 gives us a systematic method for constructing F^* by approximation of all the basic functions $f \in (L \circ\!\!\rightarrow L)$ composing F by an abstract function $f^* \in (L^* \circ\!\!\rightarrow L^*)$ chosen such that $f \leqslant \bar\gamma(f^*)$. Finally the least fixed point of F^* can be mechanically computed by successive approximations (2.11).

(Note that we considered systems of equations of a rather restricted format : instead of elements of $((D_1 \circ\!\!\rightarrow D_1')\times\ldots\times(D_n \circ\!\!\rightarrow D_n')) \circ\!\!\rightarrow ((D_1 \circ\!\!\rightarrow D_1')\times\ldots\times(D_n \circ\!\!\rightarrow D_n'))$ we have the simpler form $((D \circ\!\!\rightarrow D) \circ\!\!\rightarrow (D \circ\!\!\rightarrow D))$. This is a restriction imposed for convenience of the sake only. All the results of paragraph 4.1 go through for the more general case but the formalism becomes rather heavy !).

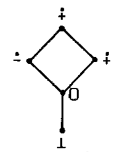
Example : Let us consider the simple procedure :

```
φ1 | proc f( value x∈Π; result y∈Π ) =
φ2 |   if x>1000 then
φ3 |      y:=x;
   |   else
φ4 |      begin new z∈Π  := -2*y ;
φ5 |         f(z;y);
φ6 |         y:=-y;
   |      end;
   |   fi;
```

The system of semantic equations associated with the above procedure is :

$$\left\{\begin{array}{ll}
\phi_1 \in (\Pi\to\mathbb{B}) \circ\!\!\rightarrow (\Pi^2\to\mathbb{B}) & = \lambda P.\{\bar\sigma_3^3(\phi_2(\underline{entry}(\Pi,\Pi)(P)))\} \\
\phi_2 \in (\Pi^3\to\mathbb{B}) \circ\!\!\rightarrow (\Pi^3\to\mathbb{B}) & = \lambda P.\{\phi_3(P\ \underline{and}\ \lambda(a,y,x).[x>1000])\ \underline{\underline{or}}\ \phi_4(P\ \underline{and} \\
 & \qquad \lambda(a,y,x).[x\leq 1000])\} \\
\phi_3 \in (\Pi^3\to\mathbb{B}) \circ\!\!\rightarrow (\Pi^3\to\mathbb{B}) & = \lambda P.\{\underline{assign}(2,\lambda(a,y,x).[x],\Pi^3)(P)\} \\
\phi_4 \in (\Pi^3\to\mathbb{B}) \circ\!\!\rightarrow (\Pi^3\to\mathbb{B}) & = \lambda P.\{\bar\sigma_4^4(\phi_6(\phi_5(\underline{block}(\lambda(a,y,x).[-2*y],\Pi^3,\Pi)(P))))\} \\
\phi_5 \in (\Pi^4\to\mathbb{B}) \circ\!\!\rightarrow (\Pi^4\to\mathbb{B}) & = \lambda P.\{\overset{4}{\Sigma}_{4,2}(\phi_1(\sigma_4^4(P)))\ \underline{\underline{and}}\ \overset{4}{\bar\Sigma}_2(\bar\sigma_2^n(P))\} \\
\phi_6 \in (\Pi^4\to\mathbb{B}) \circ\!\!\rightarrow (\Pi^4\to\mathbb{B}) & = \lambda P.\{\underline{assign}(2,\lambda(a,y,x,z).[-y],\Pi^4)(P)\}
\end{array}\right.$$

The space of abstract properties is $\alpha^m(\Pi^m\to\mathbb{B}) = (L^{*m}, \bot^m, \pm^m, \sqcap^m, \sqcup^m)$ where :

$L^* =$ 

Therefore a predicate on m variables is approximated by the m-tuple of the signs of these variables. More precisely we have $\gamma^m((s_1,\ldots,s_m)) = \underset{i=1}{\overset{m}{\underline{AND}}}\,\gamma^1(s_i)$ where,

$$\begin{array}{l}
\gamma^1 = \lambda s.\underline{case}\ s\ \underline{in} \\
\qquad \bot \to \lambda x.[x=\Omega]; \\
\qquad 0 \to \lambda x.[(x=\Omega)\ \underline{or}\ (x=0)]; \\
\qquad + \to \lambda x.[(x=\Omega)\ \underline{or}\ (x\geq 0)]; \\
\qquad - \to \lambda x.[(x=\Omega)\ \underline{or}\ (x\leq 0)]; \\
\qquad \pm \to \lambda x.\underline{true}; \\
\quad \underline{esac};
\end{array}$$

Note that we ignore the absolute value of variables and the possible relationships between variables.

The abstract system of equations is given by a simple transcription replacing an unknown predicate $P\in(\Pi^n\to\mathbb{B})$ by an unknown tuple $(s_1,\ldots,s_n)\in L^{*n}$, and each basic function f by an abstract function f^* as follows :

- $\bar\sigma_i^{*n}((s_1,\ldots,s_i,\ldots,s_n)) = (s_1,\ldots,s_{i-1},s_{i+1},\ldots,s_n)$
- $\underline{entry}^*(\Pi^n,\Pi^m)((s_1,\ldots,s_n)) = (s_1,\ldots,s_n,\underbrace{\bot,\ldots,\bot}_{m\ \text{times}},s_1,\ldots,s_n)$
- $\underline{and}^* \in L^{*m}\times L^{*m} = \sqcap^m$
- $\underline{or}^* \in L^{*m}\times L^{*m} = \sqcup^m$
- $\underline{assign}^*(i,E,\Pi^n)((s_1,\ldots,s_n)) = (s_1,\ldots,s_{i-1},\underline{val}(E,(s_1,\ldots,s_n)),s_{i+1},\ldots,s_n)$ where $\underline{val}$ is a function which computes the sign of an expression knowing the signs $(s_1,\ldots,s_n)$ of its variables. It essentially transcribes the expression E on L^{*n} using the classical rules of signs such as : $+\oplus+=+$, $-\oplus+=+\oplus-=\pm$, $-\oplus\pm=\pm\oplus-=\pm$, $\bot\oplus s=s\oplus\bot=\bot$, etc.
- $\underline{block}(E,\Pi^n)((s_1,\ldots,s_n)) = (s_1,\ldots,s_n,\underline{val}(E,(s_1,\ldots,s_n)))$
- $\sigma_i^{*n}((s_1,\ldots,s_i,\ldots,s_n)) = s_i$
- $\Sigma_{i,j}^{n*}((s_1,s_2)) = (\pm,\ldots,\pm,s_1,\pm,\ldots,\pm,s_2,\pm,\ldots,\pm)$ where s_1 and s_2 occupy the $i^{\underline{th}}$ and $j^{\underline{th}}$ position in the n-tuple.
- $\bar\Sigma_i^{n*}((s_1,\ldots,s_n)) = (s_1,\ldots,s_{i-1},\pm,s_{i+1},\ldots,s_n)$.

Constant predicates such as $\lambda(a,y,z).[x>1000]$ and $\lambda(a,y,x).[x\leq 1000]$ are translated into constants of L^{*3} that is respectively $(\pm,\pm,+)$ and $(\pm,\pm,\pm)$.

It is clear that the system of abstract equations can be built directly without the intermediate step of semantic equations using rules similar to the ones of paragraph 3.1.

The system of abstract equations is now :

$$\begin{cases} \phi_1 \in L^{*} \,o\!\to L^{*2} = \lambda x.\{\bar{\sigma}_3^{*3}(\phi_2([x,\bot,x]))\} \\ \phi_2 \in L^{*3} \,o\!\to L^{*3} = \lambda(a,y,x).\{\phi_3([a,y,x] \sqcap (\dot{\pm},\dot{\pm},\dot{+})) \sqcup \phi_4([a,y,x] \sqcap (\dot{\pm},\dot{\pm},\dot{\pm}))\} \\ \phi_3 \in L^{*3} \,o\!\to L^{*3} = \lambda(a,y,x).\{(a,x,x)\} \\ \phi_4 \in L^{*3} \,o\!\to L^{*3} = \lambda(a,y,x).\{\bar{\sigma}_4^{*4}(\phi_6(\phi_5([a,y,x,\ominus(\dot{+})\oplus y])))\} \\ \phi_5 \in L^{*4} \,o\!\to L^{*4} = \lambda(a,y,x,z).\{\overset{4*}{\Sigma}_{4,2}(\phi_1(z)) \sqcap (a,\dot{\pm},x,z)\} \\ \phi_6 \in L^{*4} \,o\!\to L^{*4} = \lambda(a,y,x,z).\{(a,\ominus y,x,z)\} \end{cases}$$

For the convenience of hand computations this system can be simplified as follows (the sign "a" of the value parameter "x" on procedure entry can be eliminated since no relationship can be discovered between variables) :

$\phi_1 = \lambda x.\{(x \sqcap \dot{+}) \sqcup \ominus\phi_1(\ominus x)\} = F^*(\phi_1)$.

Solving by successive approximations we get :

$$\begin{aligned} \phi_1^0 &= \bot = \lambda x.\bot \\ \phi_1^1 = F^*(\phi_1^0) &= \lambda x.\{(x \sqcap \dot{+}) \sqcup (\ominus\bot(\ominus x))\} = \lambda x.\{(x \sqcap \dot{+})\} \\ \phi_1^2 = F^*(\phi_1^1) &= \lambda x.\{(x \sqcap \dot{+}) \sqcup \ominus((\ominus x) \sqcap \dot{+})\} \\ &= \lambda x.\{(x \sqcap \dot{+}) \sqcup (x \sqcap \dot{-})\} \text{ since } (\ominus x) \sqcap \dot{+} = \ominus(x \sqcap \dot{-}) \text{ and } \ominus\ominus x = x \\ &= \lambda x.\{(x \sqcap (\dot{+} \sqcup \dot{-}))\} = \lambda x.\{x \sqcap \dot{\pm}\} = \lambda x.x \\ \phi_1^3 = F^*(\phi_1^2) &= \lambda x.\{(x \sqcap \dot{+}) \sqcup (\ominus\ominus x)\} = \lambda x.x \end{aligned}$$

Therefore the sign of f(x) is the sign of x. It is certainly difficult to automatize the above algebraic manipulations. However in practive we do not need to know ϕ_1 but only $\phi_1(x)$ for a given x, hence we can avoid formal computations.

End of Example.

4.2.1. CHAOTIC ITERATIONS FOR SOLVING FIXED POINT EQUATIONS IN FINITE SPACES

Let us consider a system of functional equations $\phi=F(\phi)$ which can be detailed as :

$$\begin{cases} \phi_i \in D_i \,o\!\to D'_i = F(\phi) \\ \qquad\qquad = \lambda(\psi_1,\ldots,\psi_n).\{\lambda P.[f_i(P,\psi_1,\ldots,\psi_n)]\}(\phi_1,\ldots,\phi_n) \\ i = 1,\ldots,n \end{cases}$$

We define an *index* to be a vector $(J_1,\ldots,J_n)$ such that $\forall i\in[1,n]$, $J_i \subseteq D_i$. Then we denote by F_J the functional defined by $F_J(\phi)=\psi$ where $\forall i\in[1,n]$, $\psi_i = \lambda X.$ <u>if</u> $X \in J_i$ <u>then</u> $F_i(\phi)(X)$ <u>else</u> $\phi_i(X)$ <u>fi</u>.

Informally, we say that the evaluation of $F_j(\phi)(Y)$ necessitates the evaluation of $\phi_i(X)$ which we denote by $F_j(\phi)(Y) \mapsto \phi_i(X)$ if and only if the term $F_j(\phi)$ is of the syntactic form $\lambda P.[a\ \ \phi_i(f(P,\phi))\ b]$ and the evaluation of $f(y,\phi)$ yields the value X.

Let V be $(V_1,\ldots,V_n)$ where $\forall i\in[1,n]$, $V_i \subseteq D_i$. We denote by ϕ-F-*closure*(V) the least vector $\bar{V}=(\bar{V}_1,\ldots,\bar{V}_n)$ such that $\forall i\in[1,n]$,
$\bar{V}_i = V_i \cup \{X : (\exists j\in[1,n],\ \exists Y\in\bar{V}_j : F_j(\phi)(X) \mapsto \phi_i(X))\}$

A *chaotic iteration sequence* corresponding to the functional F for a set $V=(V_1,\ldots,V_n)$ of values and initialized with a given ϕ^0 is a sequence $\phi^0,\phi^1,\ldots,\phi^k,\ldots$ of functions such that $\phi^0 \leq F(\phi^0)$ and $\phi^0 \leq lfp(F)$ and for any $k\geq 1$, $\phi^k = F_{J^{k-1}}(\phi^{k-1})$ where $J^0,J^1,\ldots,J^k,\ldots$ is an admissible indexing sequence that is satisfying the condition $\{\exists m\geq 0 : \{(\forall i\in[1,n]),\ (\forall X\in V_i),\ (\forall k\geq 0),\ (\exists l\in[1,m]) :$ $(X \in J_i^{k+1})$ <u>and</u> $(\forall j\in[1,n],\ \bar{W}_j \subseteq \bigcup_{p=0}^{l-1} J_j^{k+p})$ <u>where</u> $(W_i=\{X\})$ <u>and</u> $(\forall j\in[1,n]:j\neq i, W_j=\emptyset)$ <u>and</u> $(\bar{W}=\phi^{k+1}\text{-F-}closure(W))\}\}$.

Example : Consider Ackermann's function over the natural numbers :

```
φ1 | proc f( value x,y∈ℕ ; result a∈ℕ ) =
   |   if x=0 then
   |      a:=y+1;
   |   else if y=0 then
   |          f(x-1,1;a);
   |        else
   |          begin new t∈ℕ := Ω;
   |             f(x,y-1;t);
φ2 |             f(x-1,t;a);
   |          end;
   |        fi;
   |   fi;
```

The space of abstract properties is chosen to be L^* = the lattice with $\top$ on top, 0 and $+$ in the middle and $\bot$ at the bottom, and after simplifications the corresponding system of equations is :

$$\begin{cases} \phi_1 = \lambda(x,y).[\mathit{incr}(y) \sqcup \phi_1(\mathit{decr}(x \sqcap +),+) \sqcup \phi_2(x \sqcap +, \phi_1(x \sqcap +, \mathit{decr}(y \sqcap +)))] \\ \phi_2 = \lambda(x,t).[\phi_1(\mathit{decr}(x),t)] \end{cases}$$

where, *incr* = $\lambda x.$<u>case</u> x <u>in</u> $\bot\to\bot$; $0\to+$; $+\to+$; $\top\to+$ <u>esac</u>;
decr = $\lambda x.$<u>case</u> x <u>in</u> $\bot\to\bot$; $0\to\bot$; $+\to\top$; $\top\to\top$ <u>esac</u>;

The value of $\phi_1(\top,\top)$ can be computed by a chaotic iteration sequence corresponding to the above system of equations for a set $V=(\{(\top,\top)\},\emptyset)$ of values and starting with a given initial approximation ϕ^0 defined by :

$$\begin{bmatrix} \phi_1^0 = \lambda(x,y).\bot \\ \phi_2^0 = \lambda(x,y).\bot \end{bmatrix}$$

Step 1, $J_0 = (\{(\top,\top),(+,\top),(\top,+)\},\emptyset)$

$\phi_1^1(\top,\top) = + \sqcup \phi_1^0(\top,+) \sqcup \phi_2^0(+,\phi_1^0(+,\top)) = +$

$\phi_1^1(+,\top) = + \sqcup \phi_1^0(\top,+) \sqcup \phi_2^0(+,\phi_1^0(+,\top)) = +$

$\phi_1^1(\top,+) = + \sqcup \phi_1^0(\top,+) \sqcup \phi_2^0(+,\phi_1^0(+,\top)) = +$

and ϕ^1-F-*closure*$((\{(\top,\top)\},\emptyset)) = (\{(\top,\top),(\top,+),(+,\top)\},\{(+,\bot)\})$

Step 2, $J_1=(\emptyset,\{(+,+)\})$

$\phi_2^2(+,+) = \phi_1^1(\top,+) = +$

Step 3, $J_1 = (\{(\top,\top),(+,\top),(\top,+)\},\emptyset)$

$\phi_1^3(\top,\top) = + \sqcup \phi_1^2(\top,+) \sqcup \phi_2^2(+,\phi_1^2(+,\top))$

$= + \sqcup \phi_1^1(\top,+) \sqcup \phi_2^2(+,\phi_1^1(+,\top)) = +$

$\phi_1^3(+,\top) = + \sqcup \phi_1^2(\top,+) \sqcup \phi_2^2(+,\phi_1^2(+,\top)) = +$

$\phi_1^3(\top,+) = + \sqcup \phi_1^1(\top,+) \sqcup \phi_2^2(+,\phi_1^1(+,\top)) = +$

and ϕ^3-F-*closure*$((\{(\top,\top)\},\emptyset)) = (\{(\top,\top),(\top,+),(+,\top)\},\{(+,+)\})$

Step 4, $J_3 = (\emptyset,\{(+,+)\})$

$\phi_2^4(+,+) = \phi_1^3(\top,+) = +$

Step 5, $J_4 = (\{(\top,\top)\})$

$\phi_1^5(\top,\top) = + \sqcup \phi_1^4(\top,+) \sqcup \phi_2^4(+,\phi_1^4(+,\top))$

$= + \sqcup \phi_1^3(\top,+) \sqcup \phi_2^4(+,\phi_1^3(+,\top)) = +$

and ϕ^5-F-*closure*$((\{(\top,\top)\},\emptyset)) = (\{(\top,\top),(\top,+),(+,\top)\},\{(+,+)\})$

The iteration process stabilizes so that $\phi_1(\top,\top)=+$ proving that Ackermann's function yields a strictly positive natural number. *End of Example.*

LEMMA 4.2.1.1. A chaotic iteration sequence $\phi^0,\phi^1,\ldots,\phi^k,\ldots$ is an increasing chain : $\{\forall k\geq 0,\ \phi^k \subseteq \phi^{k+1} \subseteq F(\phi^k) \subseteq lfp(F)\}$.

Proof : Basis : $\phi^0 \subseteq F(\phi^0) \subseteq lfp(F)$ implies $\forall i\in[1,n]$, $\forall X\in D_i$, $\phi_i^0(X) \sqsubseteq F_i(\phi^0)(X) \sqsubseteq lfp(F)_i(X)$. If $X\in J_i^0$ then $\phi_i^0(X) \sqsubseteq \phi_i^1(X)=F_i(\phi^0)(X)$ otherwise $X\notin J_i^0$ and $\phi_i^0(X)=\phi_i^1(X) \sqsubseteq F_i(\phi^0)(X)$.

Induction step : assume that for some $k>0$ we have $\phi^{k-1} \subseteq \phi^k \subseteq F(\phi^{k-1}) \subseteq lfp(F)$. Now $\forall i\in[1,n]$, $\forall X\in D_i$ then either $X\in J_i^{k-1}$ and then $\phi_i^k(X)=F_i(\phi^{k-1})(X) \sqsubseteq F_i(\phi^k)(X) \sqsubseteq lfp(F)_i(X)$ since F_i is monotone otherwise $X\notin J_i^{k-1}$ and then $\phi_i^k(X)=\phi_i^{k-1}(X) \sqsubseteq F_i(\phi_i^{k-1})(X)$ by induction hypothesis. Moreover $F_i(\phi^{k-1})(X) \sqsubseteq F_i(\phi^k)(X) \sqsubseteq lfp(F)_i(X)$ proving that $\phi_i^k \subseteq F_i(\phi^k) \subseteq lfp(F)$. Now $\forall i\in[1,n]$, $\forall X\in D_i$ then if $X\in J_i^k$ then $\phi_i^k(X) \sqsubseteq F_i(\phi^k)(X)=\phi_i^{k+1}(X)$ else $\phi_i^k(X)=\phi_i^{k-1}(X) \sqsubseteq F_i(\phi^k)(X)$ proving that $\phi_i^k \subseteq \phi_i^{k+1} \subseteq F_i(\phi^k) \subseteq lfp(F)$, the lemma is established by recurrence on k. *End of Proof.*

In practice we are only interested in chaotic iteration sequences which *stabilizes*, that is the chain $\phi^0,\phi^1,\ldots,\phi^k,\ldots$ is not an infinite strictly increasing chain : $\{\exists s\geq 0 : \forall k\geq s,\ \phi^s=\phi^k\}$.

THEOREM 4.2.1.2 Let $\phi^0,\phi^1,\ldots,\phi^k,\ldots$ be a chaotic iteration sequence corresponding to the functional F for a set $V=(V_1,\ldots,V_n)$ of values. Assume that the sequence stabilizes after s steps then,

$\{(\forall i\in[1,n]),\ (\forall X\in V_i),\ \phi_i^s=lfp(F)_i(X)\}$

Proof : Let W^s be ϕ^s-F-*closure*(V). Assume that $X\in V_i \subseteq W_i^s$. By definition of an admissible indexing sequence $\forall s\geq 0$, $\exists l$ such that $X\in J_i^{s+l}$ hence $\phi_i^{s+l+1}(X)=F_i(\phi^{s+l})(X)$ and since $\phi^s=\phi^{s+l+1}=\phi^{s+l}$ we conclude $\phi_i^s(X)=F_i(\phi^s)(X)$. Assume now that $X\in(W_i^s-V_i)=W_i^{s+m}-V_i)$. Hence $\exists Y\in V_j$ such that $F_j(\phi^{s+m})(Y) \bullet\!\!\rightarrow \phi_i^{s+m}(X)$, therefore $\exists l$ such that $Y\in J_i^{s+l}$ and since $\phi^{s+m}=\phi^{s+l}$, X belongs to the i-th component of the ϕ^{s+l}-F-*closure* of $(\emptyset,\ldots,\{Y\},\ldots,\emptyset)$ (where $\{Y\}$ is in the j-th position). Then $\exists p$ such that $X\in J_i^{s+p}$ in which case $\phi_i^{s+p+1}(X)=F_i(\phi^{s+p})(X)$ proving that $\phi_i^s(X)=F_i(\phi^s)(X)$ since stabilization occurs after s steps. In short we have proved that $\forall i\in[1,n]$ $(\phi_i|W_i^s) = (F_i|((W_1^s \circ\!\!\rightarrow W_1'^s)\times\ldots\times(W_n^s \circ\!\!\rightarrow W_n'^s)))$ where $W_i'^s=\phi_i(W_i^s)$.

Let us now define ψ^0 to be a vector $(\psi_1^0,\ldots,\psi_n^0)$ of functions such that $\forall i\in[1,n]$, $\psi_i^0=\lambda X.[\underline{\text{if}}\ X\in W_i^s\ \underline{\text{then}}\ \phi_i^s(X)\ \underline{\text{else}}\ \bot\ \underline{\text{fi}}]$. $\forall X\in D_i$, if $X\in W_i^s$ then we have $\psi_i^0(X)=F_i(\psi^0)(X)$ else $X\in(D_i-W_i^s)$ and then $\psi_i^0(X)=\bot \sqsubseteq F_i(\psi^0)(X)$. Therefore $\psi^0 \subseteq F(\psi^0)$ and according to lemma 4.2.1.1 $\psi^0 \subseteq lfp(F)$. Therefore $lfp(F)=\psi^\infty=\lim_{k\to\infty}\psi^k$ where $\psi^k=F^k(\psi^0)$ (theorem 2.11). Let us prove by recurrence that $\{\psi^0$-F-*closure*$(W) = \psi^k$-F-*closure*$(W) = W\}$ and $\{\forall i\in[1,n],\ \forall X\in W_i,\ \psi_i^k(X)=\psi_i^0(X)\}$. The basis k=0 is trivial, the induction step k+1 follows : we have $\psi_i^{k+1}(X)=F_i(\psi^k)(X)$. $\forall Z\in D_j$ such that $F_i(\psi^k)(X) \bullet\!\!\rightarrow \psi_j^k(Z)$, Z belongs to W_j by induction hypothesis and therefore $\psi_j^k(Z)=\psi_j^0(Z)$ so that $\psi_i^{k+1}(X) = F_i(\psi^k)(X) = F_i(\psi^0)(X) = \psi_i^0(X)$. Moreover, ψ^{k+1}-F-*closure*(W) $=W$ since otherwise $\exists i\in[1,n]$, $\exists X\in(D_i-W_i)$, $\exists j\in[1,n]$, $\exists Y\in W_j$ such that $F_j(\psi^{k+1})(Y) \bullet\!\!\rightarrow \psi_i^{k+1}(X)$. Hence by induction on the complexity of the term $F_j(\psi^{k+1})(Y)$ this implies $\exists h\in[1,n]$, $\exists Z\in W_h$ such that $\psi^{k+1}(Z)\neq\psi^0(Z)$ which has been proved to be impossible. Finally, $\forall k\geq 0$, $\forall X\in W_i$, $\psi_i^k(X)=\psi_i^0(X)$ thus passing to the limit $lfp(F)_i(X)=\psi_i^\infty(X)=\psi_i^0(X)$ $=\phi_i^s(X)$. *End of Proof.*

From the above proof notice that a chaotic iteration sequence $\phi^0,\ldots,\phi^s,\ldots,\phi^{s+m}$ which is stable after s steps : $\phi^s=\phi^{s+1}=\ldots=\phi^{s+m}$ is definitely stabilized.

The cause of stabilization of a chaotic iteration sequence depends on F and on the domains $D_1,\ldots,D_n$ and $D_1',\ldots,D_n'$. For example, whenever each D_i' is a *lattice of locally finite length* (the length of every strictly increasing chain is bounded) stabilization is guaranteed. Yet this may imply that any admissible indexing sequence must contain an index with an infinite component. In practice this untractable situation is ruled out when for example the domains D_i are finite.

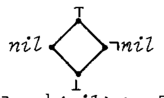
The last practical question is the one of constructing an admissible indexing sequence , this is done dynamically during the iteration process. For example, we can use the following algorithm : at each step k we evaluate $\phi_i(X)$ for all $X \in V_i$. Whenever $F_i(X,\phi) \mapsto \phi_j(Y)$ we determine the value Z of $\phi_j(Y)$ as follows :

- if $\phi_j(Y)$ has already been evaluated at step s, Z is taken to be the corresponding value,
- else if $\phi_j(Y)$ is beeing evaluated (that is $F_j(Y,\phi) \mapsto \phi_j(Y)$) then Z is taken to be the value of $\phi_j(Y)$ computed at step s-1 if $\phi_j(Y)$ has been evaluated at step s-1 otherwise Z is equal to the infimum $\perp$ of D_j,
- else Z is taken to be the value of $F_j(Y,\phi)$.

Example : Consider the reverse function over linked linear lists :

```
        type node = record val∈Π ; next∈↑node end;
φ1 |    proc rev(value x,y∈↑node ; result z∈↑node) =
   |        if x=nil then
   |            z:=y;
   |        else
   | φ2 |       begin new t∈↑node := allocate(node);
   |    |           t.val:=x.val;
   |    |           t.next:=y;
   |    | φ3 |      rev(x.next,t;z);
   |    |       end;
   |        fi;
```

The abstract space of pointer values is chosen to be :

$L^* =$ 

so that $\gamma^1(\perp)=\lambda p.[p=\Omega]$, $\gamma^1(nil)=\lambda p.[(p=\underline{nil})\underline{or}(p=\Omega)]$, $\gamma^1(\neg nil)=\lambda p.[p\in\uparrow node-\{\underline{nil}\}]$, $\gamma^1(\top)=\lambda p.[p\in\uparrow node]$.

The system of equations associated with *rev* is :

$$\begin{cases} \phi_1 = \lambda(x,y).[\underline{case}\ x\ \underline{in}\ \perp\to\perp; nil\to y; \neg nil\to\sigma_3^{*3}(\phi_2(\neg nil,y,\perp)); \top\to y\sqcup\sigma_3^{*3}(\phi_2(\top,y,\perp)); \underline{esac}] \\ \phi_2 = \lambda(x,y,z).[\bar\sigma_4^{*4}(\phi_3(x,y,z,\neg nil))] \\ \phi_3 = \lambda(x,y,z,t).[(x,y,\phi_1(next(x),t),t)] \end{cases}$$

Notice that the allocation of a record yields a non-nil pointer value. The statements t.val:=x.val; and t.next:=y; have no effect on the value of t. Finally the auxiliary function *next* is derived from the declaration of the type *node* :

$next = \lambda p.$case p $\underline{in}$ $\perp\to\perp$; $nil\to\perp$; $\neg nil\to\top$; $\top\to\top$; $\underline{esac}$;

Let us now evaluate $\phi_1(\neg nil, nil)$,

Step 1 :

$$\begin{aligned} \phi_1(\neg nil,nil) &= \sigma_3^{*3}(\phi_2(\neg nil,nil,\perp)) \\ &= \sigma_3^{*3}(\bar\sigma_4^{*4}(\phi_3(\neg nil,nil,\perp,\neg nil))) \\ &= \sigma_3^{*3}(\bar\sigma_4^{*4}(\neg nil,nil,\phi_1(\top,\neg nil),\neg nil)) \\ &= \sigma_3^{*3}(\bar\sigma_4^{*4}(\neg nil,nil,(\neg nil\sqcup\sigma_3^{*3}(\phi_2(\top,\neg nil,\perp))),\neg nil)) \\ &= \sigma_3^{*3}(\bar\sigma_4^{*4}(\neg nil,nil(\neg nil\sqcup\sigma_3^{*3}(\bar\sigma_4^{*4}(\phi_3(\top,\neg nil,\perp,\neg nil)))),\neg nil)) \\ &= \sigma_3^{*3}(\bar\sigma_4^{*4}(\neg nil,nil(\neg nil\sqcup\sigma_3^{*3}(\bar\sigma_4^{*4}(\top,\neg nil,\phi_1(\perp,\neg nil),\neg nil))),\neg nil)) \end{aligned}$$

Observe that $\phi_1(\top,\neg nil)$ is beeing evaluated and was not computed at the previous step

$$\begin{aligned} &= \sigma_3^{*3}(\bar\sigma_4^{*4}(\neg nil,nil(\neg nil\sqcup\sigma_3^{*3}(\bar\sigma_4^{*4}(\top,\neg nil,\perp,\neg nil))),\neg nil)) \\ &= \neg nil \end{aligned}$$

Step 2 :

$$\begin{aligned} \phi_1(\neg nil,nil) &= \sigma_3^{*3}(\bar\sigma_4^{*4}(\neg nil,nil(\phi_1(\top,\neg nil),\neg nil)) \\ &= \sigma_3^{*3}(\bar\sigma_4^{*4}(\neg nil,nil,(\neg nil\sqcup\sigma_3^{*3}(\bar\sigma_4^{*4}(\top,\neg nil,\phi_1(\top,\neg nil),\neg nil))),\neg nil)) \end{aligned}$$

$\phi_1(\top,\neg nil)$ is being evaluated, its value is approximated by the value at step 1.

$$\begin{aligned} &= \sigma_3^{*3}(\bar\sigma_4^{*4}(\neg nil,nil,(\neg nil\sqcup\sigma_3^{*3}(\bar\sigma_4^{*4}(\top,\neg nil,\neg nil,\neg nil))),\neg nil)) \\ &= \neg nil \quad \text{stabilization, stop.} \end{aligned}$$

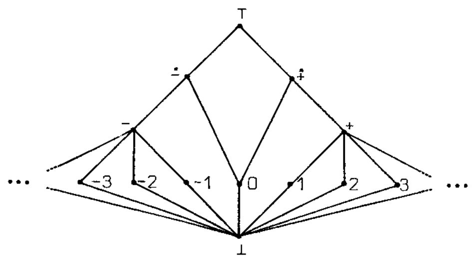
Notice that the above computations can be reordered to fit with the definition of a chaotic iteration sequence. Also this simple analysis shows that rev(L,nil,CL) returns a non nil pointer CL whenever L is not nil. *End of Example.*

4.2.2. STRENGTHENED CHAOTIC ITERATIONS FOR APPROXIMATING THE SOLUTION TO FIXED POINT EQUATIONS IN INFINITE SPACES

Example : Let us consider the following procedure :

```
proc fact( value x,y,z∈Π; result f∈Π ) =
   if x=y then
      f:=z;
   else
      fact(x,y+1,z*(y+1);f);
   fi;
```

such that fact(x,0,1,f) yields f=x! whenever x≥0. We consider the following infinite space of abstract properties :

$L^* =$ 

which allows the distinction between constant and non-constant integers and can be used for compile-time elimination of constant computations. Ignoring the test $(x=y)$ the simplified system of equations corresponding to the procedure *fact* is merely :

$\phi = \lambda(x,y,z).[z \sqcup \phi(x, y \oplus 1, z \otimes (y \oplus 1))]$

It is clear that $\phi(7,0,1) \mapsto \phi(7,1,1) \mapsto \phi(7,2,2) \mapsto \ldots \mapsto \phi(7,k,k!) \mapsto \ldots$ so that although L^* is of finite length any admissible indexing sequence must contain an infinite index. However corollary 4.1.7.2 shows that we can approximate ϕ by $\tilde{\phi}$ such that $\phi \sqsubseteq \tilde{\phi}$ therefore we choose :

$\tilde{\phi} = \lambda(x,y,z).[z \sqcup \tilde{\phi}(x \sqcup x, y \sqcup (y \oplus 1), z \sqcup (y \oplus 1))]$

Now whenever $\tilde{\phi}(x,y,z) \mapsto \tilde{\phi}(x',y',z')$ we have $(x,y,z) \sqsubseteq (x',y',z')$ and since any strictly increasing chain of L^* is finite this derivation process must terminate.

For example :

$\tilde{\phi}(7,0,1) = 1 \sqcup \tilde{\phi}(7, 0 \sqcup 1, 1 \sqcup 1) = 1 \sqcup \tilde{\phi}(7, \dot{+}, 1)$

$= 1 \sqcup (1 \sqcup \tilde{\phi}(7, \dot{+}, +))$

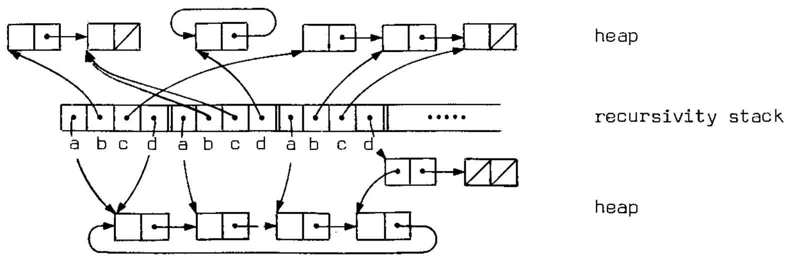
$= 1 \sqcup (1 \sqcup (+ \sqcup \tilde{\phi}(7, \dot{+}, +))) = 1 \sqcup (1 \sqcup (+ \sqcup \bot)) = +$

$\tilde{\phi}(7,0,1) = 1 \sqcup (1 \sqcup \tilde{\phi}(7, \dot{+}, +))$

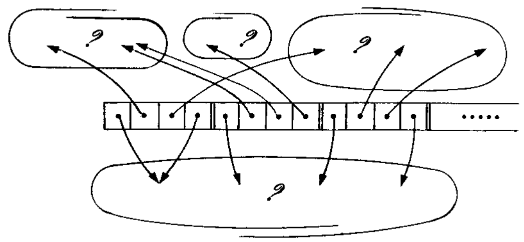
$= 1 \sqcup (1 \sqcup (+ \sqcup \tilde{\phi}(7, \dot{+}, +))) = 1 \sqcup (1 \sqcup (+ \sqcup +)) = +$

More generally $\tilde{\phi}(\top,0,1)$ is $+$ proving that the factorial is a strictly positive number (whenever *fact* terminates). *End of Example.*

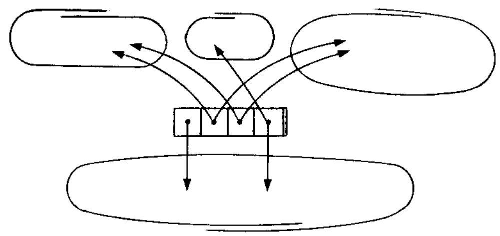
Example : The previous example can be intuitively understood as consisting in merging the frames of the execution recursivity stack. As another example let us consider the approximation of a heap data organization by *collections* (Cousot[1977b]) The following schema is assumed to be a snapshot of an actual data organization during execution of some recursive program :



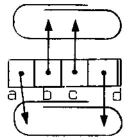
Approximating the heap data organization by collections that is each collection consists in the set of stack pointers which point in a connected subgraph on the heap, we get :



Approximating again by folding the stack frames we get :



this necessitates merging the overlapping collections, so that we finally get the following approximate representation of the actual data organization :



We represent collections as a relation between the stack pointer variables : /a,d/b,c/. They form a finite complete lattice with infimum /a/d/b/c/ supremum /a,d,b,c/. The join is the usual union of relations : /a,b,c/d,e/ $\sqcup$ /a,f,g,/h/ = /a,b,c,f,g/d,e/h/, the meet of collections is such that /a,b,c/d,e/ $\sqcap$ /a,b/c,d/e/ = /a,b/c/d/e/. We define $\underline{\varepsilon}(x,C) = C \sqcap /x/\{y : y \neq x\}/$. For example $\underline{\varepsilon}$(b,/a,b,c/d,e/) = /a,b,c/d,e/ $\sqcap$ /b/a,c,d,e/ = /b/a,c/d,e/.

We have {C}<u>if</u> x=<u>nil</u> <u>then</u>{$\underline{\varepsilon}$(x,C)} ... <u>else</u>{C}... <u>fi</u>; since a nil pointer references no record at all. The same way {C}x:=<u>allocate</u>(node){$\underline{\varepsilon}$(x,C)} since x is the only pointer referencing the newly allocated record. An assignment such as z:=y; will cause z to be disconnected from its collection and be connected to a record in the collection of y, hence {C} z:=y; {$\underline{\varepsilon}$(z,C) $\sqcup$ /z,y/}. An assignment such as t.next:=y; may cause t and y to indirectly point to a common record, hence {C} t.next:=y; {C $\sqcup$ /t,y/}. Finally, an assignment such as x:=x.next; has no effect on the organization of the collections. These informal remarks allow the understanding of the following equations associated with the procedure *rev* given at paragraph 4.2.1 :

$$\begin{cases}\phi_1 = \lambda C.[\phi_2(C,C)] & \text{(A copy of C will allow the folding of stack frames)}\\ \phi_2 = \lambda(C,C').[\phi_3(C,\underline{\varepsilon}(x,C'))\sqcup\phi_4(C,C')] \\ \phi_3 = \lambda(C,C').[\underline{\varepsilon}(z,C')\sqcup /z,y/] \\ \phi_4 = \lambda(C,C').[\phi_5(C,\underline{\varepsilon}(t,C'))] \\ \phi_5 = \lambda(C,C').[\phi_6(C,C'\sqcup /t,y/)] \\ \phi_6 = \lambda(C,C').[\phi_1(C\sqcup(C'\sqcup /y,t/))]\end{cases}$$

For the convenience of hand computations this system is simplified :

$\phi_1 = \lambda C.[(\underline{\varepsilon}(z,\underline{\varepsilon}(x,C))\sqcup /z,y/)\sqcup\phi_1(C\sqcup(\underline{\varepsilon}(t,C)\sqcup /y,t/))]$

Assume now that we want to determine the effect of a call rev(a,b;c) where a and b are pointer variables referencing disjoint collections, we have :

$$\begin{aligned}\phi_1(/a,x/b,y/z/) &= (/a/x/b,y,z/)\sqcup\phi_1(/a,x/b,t,y/z/)\\ &= (/a/x/b,y,z/)\sqcup((/a/x/b,t,y,z/)\sqcup\phi_1(/a,x/b,t,y/z/))\\ &= (/a/x/b,y,z)\sqcup((/a/x/b,t,y,z/)\sqcup(/a/x/b/t/y/z/))\\ &= (/a/x/b,t,y,z/)\end{aligned}$$

$$\begin{aligned}\phi_1(/a,x/b,y/z/) &= (/a/x/b,y,z/)\sqcup((/a/x/b,t,y,z/)\sqcup\phi_1(/a,x/b,t,y/z/))\\ &= (/a/x/b,y,z/)\sqcup((/a/x/b,t,y,z/)\sqcup(/a/x/b,t,y,z/)\\ &= (/a/x/b,t,y,z/)\end{aligned}$$

proving that whenever a and b reference disjoint collections then rev(a,b;c) may cause b and c to indirectly reference the same record whereas a shares no record with b or c. *End of Example.*

The ideas which have been sketched in the above introductory examples are now generalized.

Let $\phi=F(\phi)$ be a system of functional equations. A strengthened version of F is an operator $\tilde{F}$ such that $F\sqsubseteq\tilde{F}$.

Let $\nabla\in L\times L\to L$ be an operation such that $\{\forall x,y\in L,\ x\sqcup y\sqsubseteq x\nabla y\}$.

Let J be an index. We defined $\tilde{F}_J(\phi)=\psi$ where $\forall i\in[1,n]$, $\forall X\in D_i$, $\psi_i(X)=$ if $X\notin J_i$ then $\phi_i(X)$ else $\phi_i(X)\nabla\tilde{F}_i(\phi)(X)$ fi. (Note that according to this definition $\tilde{F}_J$ is extensive]

A *strengthened chaotic iteration sequence* corresponding to the functional F strengthened by $\tilde{F}$ for a set $V=(V_1,\dots,V_n)$ of values and initialized with a given ϕ^0 such that $\phi^0\sqsubseteq F(\phi^0)\sqsubseteq lfp(F)$ is a sequence $\phi^0,\phi^1\dots,\phi^k,\dots$ of functions such that for any $k\geq1$, $\phi^k=\tilde{F}_{J^{k-1}}(\phi^{k-1})$ where $J^0,J^1,\dots,J^{k-1},\dots$ is an admissible indexing sequence that is satisfying the condition $\{\exists m\geq0 : \{(\forall i\in[1,n]),\ (\forall X\in V_i),\ (\forall k\geq0),\ (\exists l\in[1,m]) : (X\in J_i^{k+1})$ and $(\forall j\in[1,n],\overline{W}_j\subseteq\bigcup_{p=0}^{l-1}J_i^{k+p})$ where $(W_i=\{X\})$ and $(\forall j\in[1,n]:j\neq i,W_j=\emptyset)$ and $(\overline{W}=\phi^{k+1}\text{-}\tilde{F}\text{-}closure(W))\}\}$.

THEOREM 4.2.2.1 The limit ϕ^s of any strengthened chaotic iteration sequence which stabilizes after s steps is such that :

$$\{(\forall i\in[1,n]),\ (\forall X\in V_i),\ lfp(F)_i(X)\sqsubseteq\phi_i^s(X)\}$$

Proof : The sequence of strenghtened chaotic iterations is an increasing chain since $\forall k\geq0$, $\tilde{F}_{Jk}$ is extensive and since it is assumed to stabilize $\forall k\geq s$, $\phi^k=\phi^s$. Let $W^s=\phi^s\text{-}\tilde{F}\text{-}closure(V)$, we show that $(\phi^s|W^s)$ is a post fixed point of $(\tilde{F}|(W^1\to W'^1)\times\dots\times(W^n\to W'^n))$ where $W'^i=\phi_i(W^i)$. $\forall i\in[1,n]$, if $X\in V_i\subseteq W_i^s$ then $\forall s\geq0$, $\exists l\leq m$ such that $X\in J_i^{s+l}$ and therefore $\phi_i^{s+l+1}(X)=\tilde{F}_i(\phi^{s+l})(X)$. Since $\phi^s=\phi^{s+l}=\phi^{s+l+1}$ we have $\phi_i^s(X)=\phi_i^s(X)\nabla\tilde{F}_i(\phi)(X)$ therefore $\phi_i^s(X)\sqsubseteq\phi_i^s(X)\sqcup\tilde{F}_i(\phi)(X)$ and $\tilde{F}_i(\phi)(X)\sqsubseteq\phi_i^s(X)$. Assume now that $X\in W_i^s-V_i=W_i^{s+m}-V_i$. Hence $\exists Y\in V_j$ such that $\tilde{F}_j(\phi^{s+m})(Y)\mapsto\phi_i^{s+m}(X)$, so that according to the definition of an admissible indexing sequence $\exists l\leq m$ such that $Y\in J_j^{s+l}$ and since $\phi^{s+m}=\phi^{s+l}$, X belongs to the i-th component of the $\phi^{s+l}\text{-}\tilde{F}\text{-}closure$ of $(\emptyset,\dots,\{Y\},\dots,\emptyset)$ where $\{Y\}$ is in the j-th position of the tuple. Then $\exists p$ such that $X\in J_i^{s+p}$ and since $\phi_i^{s+p+1}=\phi_i^{s+p}=\phi^s$ we necessarily have $\phi_i^s(X)=\phi_i^s(X)\nabla\tilde{F}_i(\phi^s)(X)$ proving that $\tilde{F}_i(\phi^s)(X)\sqsubseteq\phi_i^s(X)$.

Let $\psi=(\psi_1,\dots,\psi_n)$ defined by $\psi_i(X)=$ if $X\in W_i^s$ then $\phi_i^s(X)$ else $\top$ fi; ψ is a post fixed point of $\tilde{F}$ hence since $F(\psi)\sqsubseteq\tilde{F}(\psi)\sqsubseteq\psi$ it is also a post fixed point of F so that according to theorem 2.10 $lfp(F)\sqsubseteq\psi$ proving that $\forall i\in[1,n]$, $\forall X\in V_i$, $lfp(F)_i(X)\sqsubseteq\phi_i(X)$. *End of Proof.*

Example : Let L^* be the set of couples $[a,b]$ where $a,b\in\mathbb{N}\cup\{-\infty,+\infty\}$ and $a\leq b$ augmented by $\perp$ (see 4.1.1.b) with the interpretation $\gamma^1([a,b])=\lambda x.[(a\leq x\leq b)\ \underline{or}\ (x=\Omega)]$.

Consider the two functional equations :

$$\begin{cases}\phi_1\in L^*\to L^* = \lambda x.\{\phi_1(x+[1,1])\}\\ \phi_2\in L^*\to L^* = \lambda x.\{[0,0]\sqcup([1,1]+\phi_2(x))\}\end{cases}$$

They illustrate two phenomenons which may be observed when considering an infinite space L^*. On one hand (i), $\phi_1([0,255])\mapsto\phi_1([1,256])\mapsto\phi_1([2,257])\mapsto\dots$ so that this infinite derivation requires any admissible indexing sequence to contain indexes with infinite components. On the other hand (ii), we have $\phi_2([0,255]) = [0,0]\sqcup([1,1]+\phi_2([0,255])$ and therefore we have to solve by successive approximations the equation $x=[0,0]\sqcup([1,1]+x)$ the approximation sequence of which converges after infinitely many steps : $[0,0],[0,1],[0,2],\dots,[0,+\infty]$.

The definition of chaotic iteration sequences has been designed to cope with these problems. Let us define the widening operation ∇ on L^* by $\{\forall x\in L^*$, $\perp\nabla x=x\nabla\perp=\perp\}$ and $\{[a,b]\nabla[c,d]=[$if $c<a$ then $-\infty$ else a fi, if $d>b$ then $+\infty$ else b fi;$]$. We clearly have $[a,b]\sqcup[c,d]=[\underline{min}(a,c),\underline{max}(b,d)]\sqsubseteq[a,b]\nabla[c,d]$. Note that any strictly increasing chain $c_0,c_1,\dots,c_k,\dots$ of the form $c_0=i_0$, $c_1=c_0\nabla i_1$, ..., $c_k=c_{k-1}\nabla i_k$, ... is finite for arbitrary intervals $i_0,i_1,\dots,i_k,\dots$

Approximating the above "unsolvable" system of equations, we get :

$$\begin{cases}\tilde{\phi}_1 = \lambda x.\{\tilde{\phi}_1(x\nabla(x+[1,1]))\}\\ \tilde{\phi}_2 = \lambda x.\{\tilde{\phi}_2(x)\nabla([0,0]\sqcup([1,1]+\tilde{\phi}_2(x)))\}\end{cases}$$

We can now solve as usual :

Step 1 : $\tilde{\phi}_1^1([0,255]) = \tilde{\phi}_1([0,255]\nabla[1,256])$
$= \tilde{\phi}_1^1([0,+\infty])$
$= \tilde{\phi}_1([0,+\infty]\nabla[2,+\infty]) = \tilde{\phi}_1^0([0,+\infty]) = \bot$

Step 2 : $\tilde{\phi}_1^2([0,255]) = \tilde{\phi}_1^2([0,+\infty]) = \tilde{\phi}_1^1([0,+\infty]) = \bot$

proving that the corresponding program never terminates. Let us solve now the functional equation defining $\tilde{\phi}_2$:

Step 1 : $\tilde{\phi}_2^1([0,255]) = \tilde{\phi}_2^0([0,255])\nabla([0,0]\sqcup([1,1]+\tilde{\phi}_2^0([0,255])))$
$= \bot\nabla([0,0]\sqcup\bot) = [0,0]$

Step 2 : $\tilde{\phi}_2^2([0,255]) = \tilde{\phi}_2^1([0,255])\nabla([0,0]\sqcup([1,1]+\tilde{\phi}_2^1([0,255])))$
$= [0,0]\nabla([0,0]\sqcup[1,1]) = [0,+\infty]$

Step 3 : $\tilde{\phi}_2^3([0,255]) = \tilde{\phi}_2^2([0,255])\nabla([0,0]\sqcup([1,1]+\tilde{\phi}_2^2([0,255])))$
$= [0,+\infty]\nabla([0,0]\sqcup[1,+\infty]) = [0,+\infty]$

Hence $\phi_2([0,255]) \sqsubseteq \tilde{\phi}_2^2([0,255]) = [0,+\infty]$. *End of Example.*

Notice that both phenomenons (i) and (ii) can occur at the same time. Also whenever L^* is of finite length, phenomenon (ii) is impossible and we can cope with (i) by taking ∇ to be $\sqcup$. More generally we do not need to perform the widening operation after one level of derivation and various *strategies* can be used in the computation tree. For example if L^* is such that all totally unordered subsets $(S\subseteq L^* : (x\in S, (y\in S)$ and $(x\neq y)) \Rightarrow$ not$(x\sqsubseteq y))$ are finite we can perform the widening operation when two values in the derivation sequence are comparable. In order to illustrate the benefit of using refined strategies for widening in the computation tree, let us consider the following example :

$\phi = \lambda x.\{(x\sqcap[-\infty,0])\sqcup\phi((x\sqcap[1,10])-2)\}$

Solving with a coarse strategy which consists in widening with respect to the root of the derivation tree we get :

$\phi^1([0,10]) = [0,0]\sqcup\phi([-1,8])$
$\sqsubseteq [0,0]\sqcup\phi([0,10]\nabla[-1,8]) = [0,0]\sqcup\phi^1([-\infty,10])$
$= [0,0]\sqcup([-\infty,0]\sqcup\phi([-1,8]))$
$\sqsubseteq [0,0]\sqcup([-\infty,0]\sqcup\phi^0([-\infty,10])) = [0,0]\sqcup([-\infty,0]\sqcup\bot)$
$= [-\infty,0]$

$\phi^2([0,10]) \sqsubseteq [0,0]\sqcup([-\infty,0]\sqcup\phi^1([-\infty,10])) = [-\infty,0]$

Solving now with a finer strategy which consists in widening with respect to the immediate ancestor in the derivation tree we get a best approximation of the result :

$\phi^1([0,10]) = [0,0]\sqcup\phi^1([-1,8])$
$= [0,0]\sqcup([-1,0]\sqcup\phi([-1,6]))$
$\sqsubseteq [0,0]\sqcup([-1,0]\sqcup\phi([-1,8]\nabla[-1,6]))$
$= [0,0]\sqcup([-1,0]\sqcup\phi^0([-1,8]))$
$= [-1,0]$

$\phi^2([0,10]) = [0,0]\sqcup\phi^2([-1,8])$
$= [0,0]\sqcup([-1,0]\sqcup\phi([-1,6]))$
$\sqsubseteq [0,0]\sqcup([-1,0]\sqcup\phi^1([-1,8]\nabla[-1,6]))$
$= [0,0]\sqcup([-1,0]\sqcup[-1,0])$
$= [-1,0]$

Let us end up this sketch of methods for approximating the solution to functional fixed point equations by a last academical example :

Example : The interpretation of MacCarthy's 91-function on intervals leads to the following equation :

$\phi_1 = \lambda x.[((x\sqcap[101,+\infty])-[10,10])\sqcup\phi_1(\phi_1((x\sqcap[-\infty,100])+[11,11]))]$

Solving by a coarse strategy that we can express by :

$\tilde{\phi}_1(x) = \lambda x.[\tilde{\phi}_1(x)\nabla\{((x\sqcap[101,+\infty])-[10,10])\sqcup\tilde{\phi}_1(x\nabla\tilde{\phi}_1(x\nabla((x\sqcap[-\infty,100])+[11,11])))\}]$

we get $\tilde{\phi}_1([-\infty,+\infty])=[91,+\infty]$. *End of Example.*

5. CONCLUSION

This work is presently extended in three main directions :

- More complicated language constructs are considered (such as arbitrary jumps out of procedures or procedure passing as value or result parameter). The problem is the one of associating a simple enough system of equations with the program.
- The examples illustrating this paper are rather academical and were given to provide an intuitive support to the formal parts. More elaborated examples and of practical interest are forthcoming.
- In fact our main concern is in discovering methods for solving or approximating the solution to functional fixed point equations in infinite spaces. Considerable progress can be made with regard to efficiency of the computation methods and above all with regard to the preciseness of the result. The abstract properties spaces used in practice have many more properties than assumed in this paper. It appears that taking account of these particular properties can be very useful in specific applications.

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

6. BIBLIOGRAPHY

K. Abdali [1976]. *A lambda-calculus model of programming languages - I. Simple constructs - II. Jumps and procedures*, Journal of Computer Languages, Pergamon Press, North Ireland, Vol.1, (1976), 287-301 and 303-320.

G. Birkhoff [1967]. *Lattice theory*, 3rd ed., Colloquium Publications, Vol.XXV, American Mathematical Society, Providence, R.I., 1967.

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

P. Cousot and R. Cousot [1977b]. *Static determination of dynamic properties of generalized type unions*, Proc. of the ACM Conference on Language Design for Reliable Software, Raleigh, North-Carolina, March 1977. Also SIGPLAN Notices 12,3(March 1977), 77-94.

P. Cousot and R. Cousot [1977c]. *Automatic synthesis of optimal invariant assertions : mathematical foundations*, Proc. of the ACM Symposium Artificial Intelligence and Programming Languages, Rochester, New-York, August 1977.

J.W. De Bakker [1976]. *Least fixed points revisited*, Theorical Computer Science, 2(1976), 155-181.

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

O. Frink [1942]. *Topology in lattices*, Trans. Amer. Math. Soc., 51(1942), 569-582.

G. Grätzer [1971]. *Lattice theory*, W.H. Freeman and Co., San Francisco, 1971.

J.L. Kelley [1961]. *General topology*, Graduate Texts in Mathematics, Springer Verlag, New-York, 1961.

J. Morgado [1962]. *A characterization of the closure operators by means of one axiom.* Portugaliae Math., 21(1962), 155-156.

L. Schwartz [1970]. *Topologie générale et analyse fonctionnelle*, Hermann, Paris, 1970.

D. Scott [1972]. *Continuous lattices*, Proc. 1971 Dalhousie Conference, Lecture Notes in Math., Vol. 274, Springer-Verlag, New-York, 1971, 97-136.

D. Scott [1976]. *Data types as lattices*, SIAM Journal Computing 5, 3(1976), September, 522-587.

M. Sintzoff [1972]. *Calculating properties of programs by valuations on specific models*, Proc. ACM Conf. on Proving Assertions about Programs, Also SIGPLAN Notices 7, 1(1972), 203-207.

A. Tarski [1955]. *A lattice-theorical fixpoint theorem and its applications*, pacific Journal Math., 5(1955), 285-310.

M. Ward [1942]. *The closure operators of a lattice*, Annals Math., 43(1942), 191-196.

B. Wegbreit [1975]. *Property extraction in well-founded property sets*, IEEE Trans. on Soft. Eng., Vol. SE-1, N°3, September 1975, 270-285.

DISCUSSION

Edward Blum: You said the halting problem is reducible to that of solving your general fixed-point equation. How do you know that the approximate problem is decidable?

Cousot: It does not have to be decidable. We know some decidable classes, but we have no general characterization of when they are decidable. The undecidability of the general problem was proven by Kam and Ullman (Monotone data flow analysis frameworks, Acta Informatica, 1977).

Blum: Do you have any way of measuring how good an approximation is?

Cousot: We can compare approximations by the relative fineness of their topologies, and say one is better than another, but not how much better, that is, we do not give a numerical measure.

Andrzej Blikle: What is the output from your automatic system? Is this a formula?

Cousot: At every point of the program we give an approximate invariant which is implied by the semantics of this program.

Daniel Berry: It appears you are trying to do what is normally considered run time checking at compile time.

Cousot: The applications of our work were not considered in the paper, but that is a possible one.

IFIP TC-2 Working Conference on
Formal Description of Programming Concepts
St. Andrews, N. B., Canada, August 1–5, 1977

Organized by
IFIP Technical Committee 2
Programming
International Federation for Information Processing

Program Committee
H, Bekic, J. deBakker, A. Ershov, M. Hammer, T. Hoare
S. Igarashi, R. Milner, M. Paul, C. Pair

NORTH-HOLLAND PUBLISHING COMPANY
AMSTERDAM • NEW YORK • OXFORD

FORMAL DESCRIPTION OF PROGRAMMING CONCEPTS

Proceedings of the IFIP Working Conference on
Formal Description of Programming Concepts
St. Andrews, N. B., Canada, August 1-5, 1977

edited by
ERICH J. NEUHOLD
University of Stuttgart,
Stuttgart, Germany

1978

NORTH-HOLLAND PUBLISHING COMPANY
AMSTERDAM • NEW YORK • OXFORD