NEW INSTRUMENTS FOR IMPROVING THE ANALYSIS OF INFINITISTIC BEHAVIOUR OF PROGRAMS

by

Denis Therien

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

Research Report CS-76-46

November, 1976

The financial assistance of the National Research Council, under Science 67 Scholarship and grant A-1617 is gratefully acknowledged.

NEW INSTRUMENTS FOR IMPROVING THE ANALYSIS OF INFINITISTIC BEHAVIOUR OF PROGRAMS

Denis Therien

Abstract

Quasinets, as defined by A. Blikle, are algebras in which it is possible to investigate formally properties of computer programs. Briefly, a quasinet is a set U of objects interpreted as meanings of algorithms of a given class (e.g. flowchart, recursive procedures, etc.) together with a partial orderings $\leq$, such that $(U,\leq)$ is a complete lattice, and a binary operation $\circ$ which is continuous and additive with respect to the join of $(U,\leq)$ such that $(U,\circ,e)$ is a monoid for some element e in U. The completeness property of the lattice provides infinitistic operations necessary to deal with iteration and recursion. The finitistic join is to describe branching, and the monoid operation to describe sequencing in algorithms.

This paper introduces two new quasinets which improve the analysis of infinitistic properties of programs (e.g. non-termination). One is a modification of the quasinet of generalized languages (i.e. languages with finite and infinite words); the other is isomorphic to the quasinet of δ-relations but it avoids many of the technical inconveniences appearing in this algebra.

§1 Introduction

If we want to investigate formally the properties of a computer program, we need a precise mathematical model in which programs are representable and which is powerful enough to carry out the required analysis. Throughout this paper, a program is assumed to consist of a finite number of elementary modules (e.g. boxes in flowcharts, elementary instructions, etc.) together with a control mechanism to define the order in which the elementary modules are to be executed (e.g. flowchart structure, recursive calls, coroutine mechanisms, etc.).

The approach developed by Blikle [74,76a,76b], consists of building an algebra in which modules and sequence control are representable. The algebra must provide enough tools to enable us to analyze formally some properties of programs such as correctness, domain of termination (for which class of inputs does the program terminate), domain of looping (for which class of inputs does the program loop infinitely), domain of blocking (for which class of inputs does the program block without reaching the end). The algebras used by Blikle are called quasinets: they consist of a set U of objects representing the modules, a partial ordering $\leq$, such that $(U,\leq)$ is a complete lattice, and a continuous binary operation $\circ$, additive with respect to the join of the lattice, such that $(U,\circ,e)$ is a monoid for a certain unit element e in U. Completeness of the lattice provides infinitistic operations necessary to deal with iteration and recursion. Finitistic join is to describe branching, and the monoid operation is to describe sequencing in algorithms.

Obviously, there are many ways in which one can associate a meaningful object with a given computer program. The first step of the approach is thus to define quasinets which are adequate to investigate some class of properties. In section 2, we discuss informally two types of semantics of programs, that is, ways of associating a "useful meaning" to a given program. Section 3 defines quasinets formally and introduces an important concept, the $\mathbb{C}$-operation. Section 4 introduces a new quasinet in order to deal with symbolic semantics. In section 5, we introduce another new quasinet, this time to deal with input-output semantics. These last two quasinets are superior to previous existing ones, specially to analyze the infinitistic behaviour of programs. In section 6, we prove some properties enjoyed by our new quasinets.

The present paper is self-contained except in section 5 where some basic knowledge of the usual algebra of relations is required. Also, the reader is occasionally referred to some other articles in order to have a more complete discussion of some concepts presented here.

§2 Semantics of Programs

To describe the semantics of any language, we need to define a meaning for every expression in the language in such a way that the expressions will be understood by the users. In the case of programming languages, the expressions are programs but there exist many different ways of associating a meaning with a program. In this paper we use two different semantics of programs, and this section intends to give a brief discussion of each of them.

First, we can identify the meaning of a program with its input-output function (I-O semantics). Given a program π in an arbitrary language

L, and appropriate vectors x and y of data, we can associate with π a partial function f such that $f(x) = y$ whenever the program π, accepting x as its input, processes the data, eventually stops and yields y as its output. In this case, f will be the meaning of π. Consider for example the two following Algol programs:

```
π1 : begin integer a,b;
     loop : a := a-1;
            b := a+1;
            if a < 0 then goto loop
     end
π2 : begin integer a,b;
     loop : b := a;
            a := a-1;
            if a < 0 then goto loop
     end
```

The I-O functions f_1 and f_2 associated respectively with π_1 and π_2 are defined by the equations

$$f_1(x_1,x_2) = f_2(x_1,x_2) = (x_1-1,x_1)$$

where x_1, x_2 are integers and $x_1 \geq 1$. For arguments (x_1,x_2) with $x_1 < 1$, both f_1 and f_2 are undefined since the programs π_1 and π_2 enter an infinite loop. This example shows that in I-O semantics, two different algorithms can have the same meaning, i.e. they can compute the same I-O function. But very often it is important to analyze the different steps required by an algorithm to compute a function. In this case, I-O semantics are not sufficient and we need a different tool.

To this effect, we can use the so-called generalized formal languages. With every instruction of a program is associated a single character which is understood as the name of this instruction (symbolic semantics). Sequences of characters generated along all the control flows in the program are called runs and can be interpreted as symbolic executions of the program. The meaning of a program here is defined to be the set of all its runs; the meaning is therefore a formal language, possibly with infinite words. In the case of programs π_1 and π_2 above, if we establish the following dictionary of characters

character	instruction
A	$a := a-1$
B	$b := a+1$
C	$b := a$
T_1	$a < 0$ (first branch of the test)
T_2	$a \geq 0$ (second branch of the test)

then the generalized language of runs generated by our programs are:

$$(ABT_1)^*ABT_2 \cup (ABT_1ABT_1\ldots)$$

and $$(CAT_1)^*CAT_2 \cup (CAT_1CAT_1\ldots).$$

Of course, if we know the functional meanings of the symbols, then we realize that the only possible executions of π_1 are ABT_2 and $ABT_1ABT_1\ldots$. But we cannot establish this fact dealing with the set of runs since in this semantics we assume to have no access to the functional meanings of the characters.

§3 Quasinets

3.1 Preliminary Notions

Let $(U,\leq)$ be a complete lattice. For any set $P \subseteq U$ and any elements x and y in U, we denote by:

$\cup P$: the l.u.b. of P

$\cap P$: the g.l.b. of P

$x \cup y = \cup\{x,y\}$: the join of x and y

$x \cap y = \cap\{x,y\}$: the meet of x and y

A set $P \subseteq U$ is said to be directed provided that it is non-empty and that every finite subset of P has an upper bound in P.

A function $\phi: U \to U$ is said to be additive if it preserves finite joins, i.e. for any $x,y \in U$, $\phi(x \cup y) = \phi(x) \cup \phi(y)$. It is said to be continuous if it preserves l.u.b. of directed sets, i.e. for any directed set $P \subseteq U$ $\phi(\cup P) = \cup\{\phi(x) | x \in P\}$.

A function $\phi: U^n \to U$ with $n \geq 1$ is called additive (continuous) if it is additive (continuous) for each argument separately.

By a quasinet we shall mean any system $Q = (U, \leq, \circ, e)$ where U is a set, $\leq$ is a partial ordering in U, $\circ$ is a binary operation in U called composition, e is an element in U and the following conditions are satisfied:

i) $(U,\leq)$ is a complete lattice

ii) $(U,\circ,e)$ is a monoid with unit e

iii) $\circ$ is additive and continuous

iv) for all x in U, $\underline{o} \circ x = \underline{o}$ where $\underline{o} = \cap U$ is the bottom element of the lattice.

For any element x in U, we define the n-th power and the *-iteration:

$$x^0 = e$$
$$x^{n+1} = x^n \circ x$$
$$x^* = \bigcup_{n=0}^{\infty} x^n .$$

3.2 Interpretation

The set U is a set of objects that we associate with algorithms and their modules. From now on we will not distinguish between a module and the element of U to which it is associated. The lattice and the monoid over U should be defined so that join corresponds to alternative branching and composition to sequencing (Fig.3.2.1)

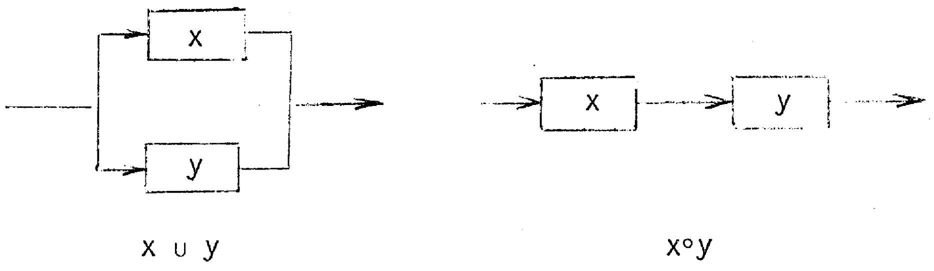

Fig. 3.2.1

This interpretation motivates in a natural way the additivity of composition (Fig.3.2.2).

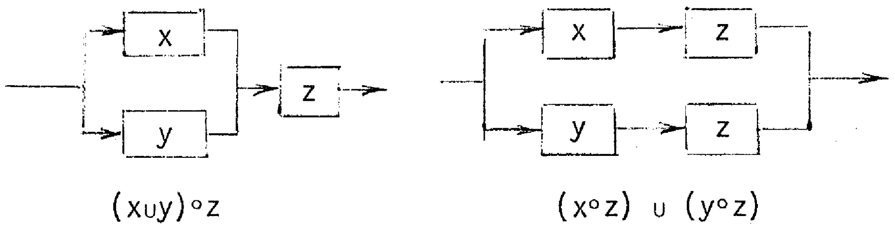

Fig. 3.2.2

The *-iteration corresponds to a denumerable alternative branching, hence can be used to describe finite looping.

The bottom element $\underline{o}$ can be interpreted as the meaning of an "empty algorithm" i.e. an algorithm without any runs. Clearly, if we link sequentially two algorithms x and y and if $x = \underline{o}$ then the compound algorithm $x \circ y$ has no runs at all. Hence we require condition iv) to hold in a quasinet.

Consider now the case $x \circ y$ where $y = \underline{o}$. If control never reaches y (e.g. if x loops infinitely) then the fact that $y = \underline{o}$ is of no concern. On the other hand, whenever x terminates, the compound algorithm $x \circ \underline{o}$ has no runs and $x \circ \underline{o} = \underline{o}$. If we are interested only in the finitistic behaviour of programs, we can require that $x \circ \underline{o} = \underline{o}$ for all $x \in U$. Blikle [74] defines a net as a quasinet in which this last property is satisfied. Thus one should work with nets whenever interested only in finitistic behaviour of programs and in quasinets otherwise.

3.3 Generalized Composition

We now introduce another operation that plays a central role in this approach of analyzing programs. To each sequence of elements in U, finite or infinite, we want to associate an element in U: the operation, denoted by $\mathbb{C}$, should correspond to sequential linking of a finite or infinite number of modules. We require that the following properties hold for our $\mathbb{C}$-operation:

$\mathbb{C}[\lambda] = e$ where λ is the empty sequence

$\mathbb{C}[u_1,\ldots,u_n] = u_1 \circ \mathbb{C}[u_2,\ldots,u_n]$ for $n \leq \infty$

It can be proved that an operation $\mathfrak{C}$ with the two properties above exists in any quasinet, but is not unique. Consequently, when dealing with a quasinet where a particular generalized composition $\mathfrak{C}$ is defined, we have to mention it explicitly and consider an extended system $Q = (U,\leq,\circ,e,\mathfrak{C})$.

Finally, given a $\mathfrak{C}$-operation in Q, we define an infinitary iteration, also called ∞-iteration, in U. For any x in U, define $x^{\infty} = \mathfrak{C}[x,x,\ldots]$. This ∞-iteration corresponds to infinite looping.

§4 Generalized Languages

4.1 Preliminaries

Let A be an arbitrary non-empty set (possibly infinite).

We consider the set of finite and infinite sequences over A; we denote finite sequences by $(a_1,\ldots,a_n)$ and infinite sequences by $(a_1,a_2,\ldots)$; we also call these sequences words (over A). We will sometimes use $(a_1,\ldots,a_n)$ $n \leq \infty$ as a shorthand notation for a sequence that may be finite or infinite.

We denote by:

λ: the empty word which is assumed to be of length 0

$seq^n(A)$: the set of all words of length n; for example, $seq^{\circ}(A) = \{\lambda\}$

$seq^*(A)$: $\bigcup_{n=0}^{\infty} seq^n(A)$: the set of finite words over A

$seq^{\infty}(A)$: the set of infinite words over A

$seq(A)$: $seq^*(A) \cup seq^{\infty}(A)$: the set of all words over A

We define a binary operation on seq(A), denoted by ^, and called concatenation:

i) $(a_1,\dots,a_n)\hat{}(b_1,\dots,b_m) = (a_1,\dots,a_n,b_1,\dots,b_m)$
for $n < \infty$, $m \leq \infty$

ii) $(a_1,a_2,\dots)\hat{}(b_1,\dots,b_m) = (a_1,a_2,\dots)$ for $m \leq \infty$.

It is clear that $(\text{seq}(A),\hat{},\lambda)$ is a monoid. Any subset L of seq(A) is called a generalized language and we denote by:

$$\text{GLan}(A) = \{L \mid L \subseteq \text{seq}(A)\}$$

We extend the operation ^ to the set GLan(A) by defining:
$L_1\hat{}L_2 = \{x_1\hat{}x_2 \mid x_1 \in L_1 \;\&\; x_2 \in L_2\}$.

Again, it is clear that $(\text{GLan}(A),\hat{},\{\lambda\})$ is a monoid where the unit $\{\lambda\}$ is the set consisting of the empty word alone. It can easily be shown that ^ is continuous and additive with respect to unions. The system $G = (\text{GLan}(A),\subseteq,\hat{},\{\lambda\})$ is thus a quasinet where $\subseteq$ is the set theoretic inclusion. Moreover, we clearly have that the bottom element of the lattice $(\text{GLan}(A),\subseteq)$ is the empty set ϕ, and this element acts as a two-sided zero in GLan(A), i.e. $L\hat{}\phi = \phi\hat{}L = \phi \quad \forall L \in \text{GLan}(A)$. Thus the system defined above is a net and can be applied to analyze programs only in so far as finitistic behaviour is concerned.

The present paper was written to solve an open problem proposed by Blikle, namely, to transform G in such a way that it can be used to analyze non-finitistic properties of programs as well.

4.2 The New Quasinet of Generalized Languages

We denote by:

$\mathrm{Fin}(L) = L \cap \mathrm{seq}^*(A)$: the set of finite words of L

$\mathrm{Inf}(L) = L \cap \mathrm{seq}^\infty(A)$: the set of infinite words of L

Some properties of the Inf and Fin operators will be needed in this section. From Redziejowski [72], we extract the following:

1) $L = \mathrm{Fin}(L) \cup \mathrm{Inf}(L)$

2) $\mathrm{Fin}(L) \cap \mathrm{Inf}(L) = \phi$

3) $\mathrm{Fin}(\bigcup_{y\in\Gamma} L_y) = \bigcup_{y\in\Gamma} \mathrm{Fin}(L_y)$

4) $\mathrm{Inf}(\bigcup_{y\in\Gamma} L_y) = \bigcup_{y\in\Gamma} \mathrm{Inf}(L_y)$

5) $\mathrm{Fin}(L_1 \hat{} L_2) = \mathrm{Fin}(L_1) \hat{} \mathrm{Fin}(L_2)$

6) $\mathrm{Inf}(L_1 \hat{} L_2) = \mathrm{Inf}(L_1) \hat{} L_2 \cup \mathrm{Fin}(L_1) \hat{} \mathrm{Inf}(L_2)$

We will also need the obvious:

7) $\mathrm{Fin}(\mathrm{Fin}(L)) = \mathrm{Fin}(L)$

8) $\mathrm{Fin}(\mathrm{Inf}(L)) = \phi$

9) $\mathrm{Inf}(\mathrm{Fin}(L)) = \phi$

10) $\mathrm{Inf}(\mathrm{Inf}(L)) = \mathrm{Inf}(L)$

We now proceed to define a new operation on GLan(A), which we denote by $\hat{\hat{}}$ and which we simply call the new concatenation:

$$L_1 \hat{\hat{}} L_2 = \mathrm{Inf}(L_1) \cup \mathrm{Fin}(L_1) \hat{} L_2$$

We can check some of the properties enjoyed by this new operation by applying properties of $\hat{}$, Inf and Fin.

1) $\hat{\wedge}$ is associative

$$(L_1 \hat{\wedge} L_2) \hat{\wedge} L_3 = \mathrm{Inf}(\mathrm{Inf}(L_1) \cup \mathrm{Fin}(L_1)\wedge L_2) \cup \mathrm{Fin}(\mathrm{Inf}(L_1) \cup \mathrm{Fin}(L_1)\wedge L_2)\wedge L_3$$

$$= \mathrm{Inf}(\mathrm{Inf}(L_1)) \cup \mathrm{Inf}(\mathrm{Fin}(L_1)\wedge L_2) \cup \mathrm{Fin}(\mathrm{Inf}(L_1))\wedge L_3 \cup \mathrm{Fin}(\mathrm{Fin}(L_1)\wedge L_2)\wedge L_3$$

$$= \mathrm{Inf}(L_1) \cup \mathrm{Fin}(L_1)\wedge\mathrm{Inf}(L_2) \cup \mathrm{Fin}(L_1)\wedge\mathrm{Fin}(L_2)\wedge L_3$$

$$L_1 \hat{\wedge} (L_2 \hat{\wedge} L_3) = \mathrm{Inf}(L_1) \cup \mathrm{Fin}(L_1)\wedge(\mathrm{Inf}(L_2) \cup \mathrm{Fin}(L_2)\wedge L_3)$$

$$= \mathrm{Inf}(L_1) \cup \mathrm{Fin}(L_1) \wedge \mathrm{Inf}(L_2) \cup \mathrm{Fin}(L_1)\wedge\mathrm{Fin}(L_2)\wedge L_3$$

2) $\{\lambda\}$ is a two-sided unit

$$L \hat{\wedge} \{\lambda\} = \mathrm{Inf}(L) \cup \mathrm{Fin}(L)\wedge\{\lambda\}$$

$$= L$$

$$\{\lambda\} \hat{\wedge} L = \mathrm{Inf}(\{\lambda\}) \cup \mathrm{Fin}(\{\lambda\})\wedge L$$

$$= \phi \cup \{\lambda\}\wedge L$$

$$= L$$

We have thus shown that $(\mathrm{GLan}(A), \hat{\wedge}, \{\lambda\})$ is a monoid.

3) $\hat{\wedge}$ distributes over arbitrary unions

$$L \hat{\wedge} (\cup_y L_y) = \mathrm{Inf}(L) \cup \mathrm{Fin}(L)\wedge(\bigcup_y L_y)$$

$$= \mathrm{Inf}(L) \cup \bigcup_y \mathrm{Fin}(L)\wedge L_y$$

$$= \bigcup_y (\mathrm{Inf}(L) \cup \mathrm{Fin}(L)\wedge L_y)$$

$$= \bigcup_y (L \hat{\wedge} L_y)$$

$$(\bigcup_y L_y) \hat{\wedge} L = \mathrm{Inf}(\bigcup_y L_y) \cup \mathrm{Fin}(\bigcup_y L_y)\wedge L$$

$$= \bigcup_y \mathrm{Inf}(L_y) \cup \bigcup_y (\mathrm{Fin}(L_y)\wedge L)$$

$$= \bigcup_y (\mathrm{Inf}(L_y) \cup \mathrm{Fin}(L_y)\wedge L)$$

$$= \bigcup_y (L_y \hat{\wedge} L)$$

4) Properties of ϕ

$\phi \hat{\hat{}} L = \phi$

$L \hat{\hat{}} \phi = \mathrm{Inf}(L)$

Consider now the system $G' = (\mathrm{GLan}(A), \subseteq, \hat{\hat{}}, \{\lambda\})$, where $\subseteq$ is the set theoretic inclusion on GLan(A); from the above properties, it is clear that $\hat{\hat{}}$ is additive and continuous.

By property 4), we see that G' is a quasinet but not a net. Our new operation has eliminated the undesirable effect of ϕ as a right zero but otherwise has conserved all the properties needed to analyze programs.

4.3 Generalized Concatenation

We now proceed to define a $\mathbb{C}$-operation in the new quasinet. First we need the following operation, $C:\mathrm{seq}(\mathrm{seq}(A)) \to \mathrm{seq}(A)$ defined by

i) $C[\lambda] = \lambda$

$C[x_1,\dots,x_n] = x_1 \hat{} C[x_2,\dots,x_n]$

ii) $C[x_1,x_2,\dots]$ is the shortest word in seq(A) with the following property

$\forall n \geq 1 \;\; x_1 \hat{} x_2 \hat{} \dots \hat{} x_n$ <u>pre</u> $C[x_1,x_2,\dots]$ where x <u>pre</u> y iff $\exists z \in \mathrm{seq}(A)$ such that $x \hat{} z = y$.

Alternatively, since $\{C[x_1], C[x_1,x_2],\dots\}$ clearly forms a chain with respect to <u>pre</u>, we can define $C[x_1,x_2,\dots] = \mathrm{l.u.b.}\{C[x_1], C[x_1,x_2],\dots\}$ and clearly both definitions are in agreement.

We extend this last operation to the following:

$\mathbb{C}: \mathrm{seq}(\mathrm{GLan}(A)) \to \mathrm{GLan}(A)$

where i) $\mathbb{C}[\lambda] = \{\lambda\}$

$\mathbb{C}[L_1,\ldots,L_n] = L_1 \hat{\wedge} \mathbb{C}[L_2,\ldots,L_n] \quad n < \infty$

ii) $\mathbb{C}[L_1,L_2,\ldots] = \{C[x_1,x_2,\ldots] \mid (\forall i \geq 1)\ x_i \in \mathrm{Fin}(L_i)\}$
$\cup \bigcup_{n=1}^{\infty} \{C[x_1,\ldots,x_n] \mid x_i \in \mathrm{Fin}(L_i) \text{ for } i = 1,\ldots,n-1 \ \&\ x_n \in \mathrm{Inf}(L_n)\}$

We call $\mathbb{C}$ the generalized concatenation of languages and clearly the properties of a $\mathbb{C}$-operation are satisfied.

Now let the n-th power, the *-iteration and the ∞-iteration have the meaning defined in an abstract quasinet (see section 3.1 and 3.3). The following properties can be checked to hold for the ∞ operator:

1)	$\phi^{\infty} = \phi$	4)	$\mathrm{Inf}(L)^{\infty} = \mathrm{Inf}(L)$
2)	$\{\lambda\}^{\infty} = \{\lambda\}$	5)	$L_1 \subseteq L_2 \Rightarrow L_1^{\infty} \subseteq L_2^{\infty}$
3)	$L \hat{\wedge} L^{\infty} = L^{\infty}$		

Among these properties, the third is specially important, for we can show that if L is λ-free (i.e. $\lambda \notin L$), then L^{∞} is the greatest solution of the equation $X = L \hat{\wedge} X$. To get this result, we need some lemmas:

<u>Lemma 4.3.1</u> If R is a solution of $X = L \hat{\wedge} X$, then $(\mathrm{Fin}(L))^* \hat{\wedge} \mathrm{Inf}(L) \subseteq R$.

<u>Proof</u> We show by induction on i that $(\mathrm{Fin}(L))^i \hat{\wedge} \mathrm{Inf}(L) \subseteq R$.

For $i = 0$, since $R = L \hat{\wedge} R = \mathrm{Inf}(L) \cup \mathrm{Fin}(L) \wedge R$, we have $\mathrm{Inf}(L) = (\mathrm{Fin}(L))^{\circ} \hat{\wedge} \mathrm{Inf}(L) \subseteq R$.

Assume the lemma to be true for an arbitrary $i \geq 0$. We have $(\mathrm{Fin}(L))^i \hat{\wedge} \mathrm{Inf}(L) \subseteq R$ by induction hypothesis. It is easily verified that the inclusion stays valid if we multiply on both sides by any language. In particular, $\mathrm{Fin}(L)\hat{\,}(\mathrm{Fin}(L))^i\hat{\,}\mathrm{Inf}(L) \subseteq \mathrm{Fin}(L)\hat{\,}R$. But $\mathrm{Fin}(L)\hat{\,}R = \mathrm{Fin}(L)\char`^R \subseteq R$. Thus $(\mathrm{Fin}(L))^{i+1}\hat{\,}\mathrm{Inf}(L) \subseteq R$. This proves that

$$\bigcup_{n=0}^{\infty} (\mathrm{Fin}(L))^n\hat{\,}\mathrm{Inf}(L) \subseteq R$$

or equivalently

$$(\mathrm{Fin}(L))^*\hat{\,}\mathrm{Inf}(L) \subseteq R. \qquad \square$$

<u>Lemma 4.3.2</u> $L^\infty = (\mathrm{Fin}(L))^*\hat{\,}\mathrm{Inf}(L) \cup (\mathrm{Fin}(L))^\infty$

<u>Proof</u> By definition

$$L^\infty = \{C[x_1,x_2,\ldots]\,|\,(\forall i\geq 1)\ x_i \in \mathrm{Fin}(L)\}$$
$$\cup \bigcup_{n=1}^{\infty} \{C[x_1,\ldots,x_n]\,|\,x_i \in \mathrm{Fin}(L) \text{ for } i = 1,\ldots,n-1 \ \& \ x_n \in \mathrm{Inf}(L)\}.$$

It is a simple matter to check that the first term of the right-hand side union is $(\mathrm{Fin}(L))^\infty$ and the second term is $(\mathrm{Fin}(L))^*\hat{\,}\mathrm{Inf}(L)$. $\square$

<u>Lemma 4.3.3</u> If L is λ-free, any solution of $X = L\hat{\,}X$ is λ-free.

<u>Proof</u> Let $R = L\hat{\,}R = \mathrm{Inf}(L) \cup \mathrm{Fin}(L)\char`^R$ and suppose that $\lambda \in R$. This implies that $\lambda \in \mathrm{Fin}(L)\char`^R$, and hence λ should be in $\mathrm{Fin}(L)$ which contradicts the fact that L is λ-free. $\square$

<u>Theorem 4.3.1</u> If L is λ-free, L^∞ is the greatest solution of $X = L\hat{\,}X$.

<u>Proof</u> Suppose that R is an arbitrary solution, i.e. $R = L\hat{\,}R = \mathrm{Inf}(L) \cup \mathrm{Fin}(L)\char`^R$.

By Lemma 4.3.1, $R = (\mathrm{Fin}(L))^*\hat{\,}\mathrm{Inf}(L) \cup B$, where we can assume that $B \cap (\mathrm{Fin}(L))^*\hat{\,}\mathrm{Inf}(L) = \phi$. By Lemma 4.3.2, it is sufficient to show that $B \subseteq (\mathrm{Fin}(L))^\infty$.

If $B = \phi$, there is nothing to prove. Else, let $y \in B$: we define an infinite sequence $\{x_1, x_2, \ldots\}$ as follows:

Since $y \in R$, we have $y \in \mathrm{Inf}(L)$ or $y \in \mathrm{Fin}(L)\hat{}R$. The first case is ruled out by the assumption that B and $(\mathrm{Fin}(L))^{*}\hat{}\mathrm{Inf}(L)$ are disjoint. Thus $y = x_1\hat{}y_1 = C[x_1]\hat{}y_1$, where $x_1 \in \mathrm{Fin}(L)$, $x_1 \neq \lambda$ since L is λ-free, and $y_1 \in R$. Also $y_1 \neq \lambda$ by Lemma 4.3.3.

Now assume that we have $x_1, \ldots, x_n$ such that $x_i \in \mathrm{Fin}(L)$ for $i = 1, \ldots, n$ and there exists $y_n \in R$ such that $y = x_1\hat{}\ldots\hat{}x_n\hat{}y_n = C[x_1, \ldots, x_n]\hat{}y_n$. We have either $y_n \in \mathrm{Inf}(L)$ or $y_n \in \mathrm{Fin}(L)\hat{}R$. Again the first case is ruled out and the second case yields $y_n = x_{n+1}\hat{}y_{n+1}$, $x_{n+1} \in \mathrm{Fin}(L)$, $y_{n+1} \in R$. Also

$$\begin{aligned} y &= C[x_1, \ldots, x_n]\hat{}x_{n+1}\hat{}y_{n+1} \\ &= C[x_1, \ldots, x_{n+1}]\hat{}y_{n+1}. \end{aligned}$$

By passing to the limit, we have an infinite sequence $(x_1, x_2, \ldots)$ such that for all i, $x_i \in \mathrm{Fin}(L)$. By construction, for all $n \geq 1$, $x_1\hat{}\ldots\hat{}x_n$ <u>pre</u> y and thus $y = C[x_1, x_2, \ldots]$ or equivalently $y \in (\mathrm{Fin}(L))^{\infty}$.

We conclude that $B \subseteq (\mathrm{Fin}(L))^{\infty}$ and the theorem is proved. □

The motivation to introduce our **C**-operation is as follows: consider a program π which has been splitted in a number of modules $\pi_1, \ldots, \pi_m$. For each module π_i, we associate to the set of its internal runs a generalized language L_i (cf: symbolic semantics of section 2). Runs of π in terms of π_i's are finite or infinite sequences of these languages. Given such a run, $(L_{i_1}, \ldots, L_{i_n})$ $n \leq \infty$, the set of the corresponding runs of π described in terms of instructions is of course $\mathbf{C}[L_{i_1}, \ldots, L_{i_n}]$. If $L_i \neq \phi$ for all i, this reduces to the analysis made in the net G. But if a module π_i has no computations ($L_{i_k} = \phi$), then the set of computations of π will be $Inf(C[L_{i_1},\ldots,L_{i_{k-1}}])$ instead of ϕ. The quasinet G' can thus be used to analyze infinitistic behaviour of a program in a more adequate way than the net G.

§5 I-O Relations

5.1 Preliminaries

The previous section was concerned with symbolic semantics. In this chapter, we show how we can use similar tools to deal with I-O semantics. We assume that the reader is familiar with basic notions of the algebra of binary relations.

First of all, we describe briefly the algebra of δ-relations as defined in the works of Blikle. Let D be an arbitrary non-empty set and let δ be an element not in D. We define

$$D_\delta = D \cup \{\delta\}$$

$$\Delta = \{(a,\delta) | a \in D_\delta\}$$

$$Rel_\delta(D) = \{Q \cup S | Q \subseteq D\times D \ \& \ S \subseteq \Delta \ \& \ (\delta,\delta) \in S\}$$

The elements of $Rel_\delta(D)$ are called δ-relations over D. They are just ordinary relations with the following assumptions:

i) $(\delta,\delta) \in R$ for all $R \in Rel_\delta(D)$

ii) there is no pair $(\delta,a) \in R$ with $a \neq \delta$

If $R = Q \cup S$ as defined above and R is an I-O relation of some program, then Q describes the transmission of inputs into outputs of the program and S describes the domain of looping: the element $(\delta,\delta) \in S$ has no interpretation and is required only for technical reasons.

Let $\circ$ represent the usual composition of relations, $\subseteq$ be the set theoretic inclusion and $I_\delta = \{(a,a) | a \in D_\delta\}$: then it can be easily shown that $Q = (\mathrm{Rel}_\delta(D), \subseteq, \circ, I_\delta)$ is a quasinet (but not a net) in which the bottom element is $\{(\delta,\delta)\}$.

5.2 The New Quasinet of I-O Relations

Let D, δ and D_δ be as in 5.1. By using a new operation on relations, very similar to the new concatenation of languages defined in section 4, we build a quasinet which is isomorphic to Q and which avoids the carry over of a dummy element (δ,δ) which has no meaning in terms of the general interpretation of the quasinet. We denote by $\mathrm{Rel}(D,D_\delta) = \{R | R \subseteq D \times D_\delta\}$; we will also use the notation aRb to describe the fact that $(a,b) \in R$ and I will stand for $\{(d,d) | d \in D\}$.

The usual composition of relations is associative in $\mathrm{Rel}(D,D_\delta)$ and I acts as a one-sided left unit but there is no right unit. We denote by

$$\mathrm{Fin}(R) = R \cap D \times D$$

$$\mathrm{Inf}(R) = R \cap D \times \{\delta\}$$

Properties 1-10 of section 4.2 have direct analogous in this case. We add another one

11) $\mathrm{Inf}(R_1) \circ R_2 = \phi$.

We define the new composition of relations, denoted by $\circledcirc$, on $\mathrm{Rel}(D,D_\delta)$:

$$R_1 \circledcirc R_2 = \mathrm{Inf}(R_1) \cup \mathrm{Fin}(R_1) \circ R_2.$$

We define a mapping ϕ from $\mathrm{Rel}_\delta(D)$ onto $\mathrm{Rel}(D_\delta)$ by $\phi(R) = R\text{-}\underline{o}$ for all $R \in \mathrm{Rel}_\delta(D)$. Clearly, ϕ is a bijection between the two sets: moreover, the ordering and arbitrary joins are preserved, i.e. $R_1 \subseteq R_2$ implies $\phi(R_1) \subseteq \phi(R_2)$ and $\phi(\bigcup_i R_i) = \bigcup_i \phi(R_i)$. To show that composition is also preserved, we proceed as follows:

$$\phi(R_1 \circ R_2) = R_1 \circ R_2 - \underline{o}$$
$$R_1 \circledcirc R_2 = \mathrm{Inf}(R_1) \cup \mathrm{Fin}(R_1) \circ R_2;$$

$(a,b) \in R_1 \circ R_2\text{-}\underline{o}$ implies that $(\exists c)\ aR_1cR_2b$ and $a \neq \delta$. If $c \neq \delta$, then $(a,c) \in \mathrm{Fin}(R_1)$ and $(a,b) \in \mathrm{Fin}(R_1) \circ R_2$. If $c = \delta$, then $b = \delta$ and $(a,b) = (a,\delta) \in \mathrm{Inf}(R_1)$. Thus $R_1 \circ R_2\text{-}\underline{o} \subseteq R_1 \circledcirc R_2$.

By a similar argument, the reverse inclusion also holds. Hence $\phi(R_1 \circ R_2) = R_1 \circledcirc R_2$.

Thus by isomorphism, $Q' = (\mathrm{Rel}(D,D_\delta), \subseteq, \circledcirc, I)$ is a quasinet, but not a net (note that $\phi(I_\delta) = I$).

5.3 Generalized Composition

We now proceed to define a $\mathbf{C}$-operation in the new quasinet.

$$\mathbf{C}: \mathrm{seq}(\mathrm{Rel}(D,D_\delta)) \to \mathrm{Rel}(D,D_\delta)$$

i) $\mathbf{C}[\lambda] = I$

$\mathbf{C}[R_1,\ldots,R_n] = R_1 \circledcirc \mathbf{C}[R_2,\ldots,R_n]$

ii) $\mathbf{C}[R_1,R_2,\ldots] = \{(a,\delta) \mid (\exists (a_1,a_2,\ldots) \in \mathrm{seq}^\infty(D))\ a = a_1$
$\&\ \forall i \geq 1\ \ a_i \mathrm{Fin}(R_i)\ a_{i+1}\}$
$\cup\ \{(a,\delta) \mid aR_1\ldots R_i\delta \text{ for some } i \geq 1\}$

This is clearly a valid $\mathfrak{C}$-operation. As usual, we use the notation R^n, R^*, R^∞ to denote the n-th power, the *-iteration and the ∞-operation. One can check the following properties of the ∞ operator:

1)	$\phi^\infty = \phi$	5)	$R^\infty = \mathrm{Inf}(R^\infty)$
2)	$I^\infty = D\times\{\delta\}$	6)	$R_1 \subseteq R_2 \Rightarrow R_1^\infty \subseteq R_2^\infty$
3)	$R \circledcirc R^\infty = R^\infty$	7)	$R^\infty = (\mathrm{Fin}(R))^* \circledcirc \mathrm{Inf}(R) \cup (\mathrm{Fin}(R))^\infty$
4)	$R^\infty \circledcirc Q = R^\infty$		

Note the difference with the case of generalized concatenation of languages in property 2): the unit of $\mathrm{Rel}(D,\delta)$ is carried over in $D\times\{\delta\}$ when raised to power ∞, while the unit of $\mathrm{GLan}(A)$ is carried over in itself.

Theorem 4.3.1 has an analogous part in this section, a bit weaker in its statement.

Theorem 5.3.1 R^∞ is a solution of the equation $X = R \circledcirc X$ and for any other solution P, the domain of P is included in the domain of R^∞.

Proof By property 3) above, $R^\infty = R \circledcirc R^\infty$ and thus R^∞ is a solution of the equation $X = R \circledcirc X$.

Let P be another solution: $P = R \circledcirc P = \mathrm{Inf}(R) \cup \mathrm{Fin}(R)\circ P$.

If $P = \phi$, there is nothing to prove. Else let $(a_1,b) \in P$; we define inductively a sequence $(a_1,a_2,\ldots,a_n)$, $n \leq \infty$, over $D \cup \{\delta\}$, as follows:

$(a_1,b)\in P$ implies $(a_1,b)\in \mathrm{Inf}(R)$ or $(\exists a_2 \in D)$ $a_1\mathrm{Fin}(R)a_2P\ b$: if the first case holds, we put $a_2 = b = \delta$ and we stop: if the second case holds, a_2 becomes the second element of the sequence and we do the induction step.

Suppose we have defined $(a_1,a_2,\ldots,a_n)$ such that $a_i \mathrm{Fin}(R) a_{i+1}$ for $i = 1,\ldots,n-1$ and $a_n \; P \; b$; then either $(a_n,b) \in \mathrm{Inf}(R)$ or $(\exists a_{n+1} \in D)$ $a_n \mathrm{Fin}(R) a_{n+1} \; P \; b$: if the first case holds, we put $a_{n+1} = b = \delta$ and we stop: else a_{n+1} becomes the n+1st element of the sequence and we go back to the induction step.

Clearly, two cases can happen:

i) the sequence $(a_1,\ldots,a_n)$ $n \geq 2$ is finite; then by construction $a_n = \delta$ and $a_1 \mathrm{Fin}(R) a_2 \ldots a_{n-2} \mathrm{Fin}(R) a_{n-1} \mathrm{Inf}(R) \delta$. This means that $(a_1,\delta) \in R^\infty$ and a_1 is in the domain of R^∞.

ii) otherwise the set of sequences over D $\{(a_1),(a_1,a_2),(a_1,a_2,a_3),\ldots\}$ has no maximal element with respect to <u>pre</u>. In this case, for all $i \geq 1$, we have by construction $a_i \mathrm{Fin}(R) a_{i+1}$ and thus $(a_1,\delta) \in R^\infty$ by the definition of the ∞ operator. Again this shows that a_1 is in the domain of R^∞.

In both cases the theorem is proved. $\square$

§6 Adequacy of the $\mathfrak{C}$-operations in the New Quasinets

6.1 Preliminaries

Let $Q = (U,\leq,\circ,e,\mathfrak{C})$ be a quasinet with specified $\mathfrak{C}$-operation. We introduce another function $\underset{\sim}{\mathfrak{C}}:(\mathrm{GLan}(U)) \to U$ defined by

$$\forall S \subseteq \mathrm{seq}(U) \quad \underset{\sim}{\mathfrak{C}}(S) = \cup\{\mathfrak{C}(t) \mid t \in S\}.$$

The motivation for introducing this operation is as follows: each run of a given program defines a sequence of elements in U which are just the objects associated with each module encountered in that particular run; by applying $\mathfrak{C}$ to that sequence, we can find an element of U which corresponds to that run; the operation $\underset{\sim}{\mathfrak{C}}$ thus gives us an element of U which corresponds to all possible runs of a given program.

A sequence $t \in seq(U)$ is said to be proper with respect to the given $\mathfrak{C}$-operation if it is either finite or whenever infinite has the property:

$$\forall u \in U \quad \mathfrak{C}(t)\circ u = \mathfrak{C}(t).$$

A set $S \subseteq seq(U)$ is proper if all its elements are proper sequences.

Now let S be a set of sequences of elements of U corresponding to all possible runs of a given program. If this program loops infinitely on itself, we obtain S^{∞} as the set of all possible runs of that new program, and the element associated to it is $\underset{\sim}{\mathfrak{C}}(S^{\infty})$. For consistency, we would like it to be equal to $(\underset{\sim}{\mathfrak{C}}(S))^{\infty}$ since $\underset{\sim}{\mathfrak{C}}(S)$ is the element of U corresponding to all possible runs of the former program. Thus we make the following definition: $\mathfrak{C}$ is said to be adequate in a given quasinet if for any proper set $S \subseteq seq(U)$ with $\lambda \notin S$, we have the property:

$$\underset{\sim}{\mathfrak{C}}(S^{\infty}) = (\underset{\sim}{\mathfrak{C}}(S))^{\infty}.$$

It is important to check that property separately since examples can be given of quasinets with valid $\mathfrak{C}$-operation in which the above equality is not satisfied.

Before proving the adequacy property in the new quasinets, we give two lemmas which hold in any quasinet. The first one is stated without proof.

Lemma 6.1.1 For any family $\{S_p | p \in P\}$ with $S_p \subseteq seq(U)$, we have $\underset{\sim}{\mathfrak{C}}(\bigcup_{p\in P} S_p) = \bigcup_{p\in P} \underset{\sim}{\mathfrak{C}}(S_p)$.

<u>Lemma 6.1.2</u> For any finite or denumerable sets $S_1, S_2 \subseteq seq(U)$, if S_1 is proper then $\underset{\sim}{\mathbf{C}}(S_1 \mathbin{\hat{\wedge}} S_2) = \underset{\sim}{\mathbf{C}}(S_1) \circ \underset{\sim}{\mathbf{C}}(S_2)$, except the case where $S_2 = \phi$ and $\underset{\sim}{\mathbf{C}}(Fin(S_1)) \circ \underline{o} \neq \underline{o}$

<u>Proof</u>

$$\begin{aligned}
\underset{\sim}{\mathbf{C}}(S_1 \mathbin{\hat{\wedge}} S_2) &= \underset{\sim}{\mathbf{C}}(Inf(S_1) \cup Fin(S_1)^\wedge S_2) \\
&= \underset{\sim}{\mathbf{C}}(Inf(S_1)) \cup \underset{\sim}{\mathbf{C}}(Fin(S_1)^\wedge S_2) \qquad \text{by Lemma 6.1.1} \\
&= \cup\{\mathbf{C}(t_1) \mid t_1 \in Inf(S_1)\} \cup \cup\{\mathbf{C}(t) \mid t \in Fin(S_1)^\wedge S_2\}
\end{aligned}$$

Using the fact that S_1 is proper and that $\circ$ is additive, we can rewrite the first term of the union

$$\begin{aligned}
\cup\{\mathbf{C}(t_1) \mid t_1 \in Inf(S_1)\} &= \cup\{\mathbf{C}(t_1) \circ \mathbf{C}(t_2) \mid t_1 \in Inf(S_1), t_2 \in S_2\} \\
&= \cup\{\mathbf{C}(t_1) \mid t_1 \in Inf(S_1)\} \circ \cup\{\mathbf{C}(t_2) \mid t_2 \in S_2\} \\
&= \underset{\sim}{\mathbf{C}}(Inf(S_1)) \circ \underset{\sim}{\mathbf{C}}(S_2)
\end{aligned}$$

The second term of the union can also be rewritten using the second property of the $\mathbf{C}$-operation and the additivity of $\circ$:

$$\begin{aligned}
\cup\{\mathbf{C}(t) \mid t \in Fin(S_1)^\wedge S_2\} &= \cup\{\mathbf{C}(t_1{}^\wedge t_2) \mid t_1 \in Fin(S_1), t_2 \in S_2\} \\
&= \cup\{\mathbf{C}(t_1) \circ \mathbf{C}(t_2) \mid t_1 \in Fin(S_1), t_2 \in S_2\} \\
&= \cup\{\mathbf{C}(t_1) \mid t_1 \in Fin(S_1)\} \circ \cup\{\mathbf{C}(t_2) \mid t_2 \in S_2\} \\
&= \underset{\sim}{\mathbf{C}}(Fin(S_1)) \circ \underset{\sim}{\mathbf{C}}(S_2)
\end{aligned}$$

Note that in this chain of equalities we need the hypothesis that we do not consider the case $S_2 = \phi$ (hence $\underset{\sim}{\mathbf{C}}(S_2) = \underline{o}$) and $\underset{\sim}{\mathbf{C}}(Fin(S_1)) \circ \underline{o} \neq \underline{o}$.

Going back to the expression for $\underset{\sim}{\mathbf{C}}(S_1 \mathbin{\hat{\wedge}} S_2)$, we get

$$\begin{aligned}
\underset{\sim}{\mathbf{C}}(S_1 \mathbin{\hat{\wedge}} S_2) &= \underset{\sim}{\mathbf{C}}(Inf(S_1)) \circ \underset{\sim}{\mathbf{C}}(S_2) \cup \underset{\sim}{\mathbf{C}}(Fin(S_1)) \circ \underset{\sim}{\mathbf{C}}(S_2) \\
&= \underset{\sim}{\mathbf{C}}(Inf(S_1) \cup Fin(S_1)) \circ \underset{\sim}{\mathbf{C}}(S_2) \\
&= \underset{\sim}{\mathbf{C}}(S_1) \circ \underset{\sim}{\mathbf{C}}(S_2). \qquad \square
\end{aligned}$$

6.2 Adequacy of $\mathbb{C}$ in $(GLan(A), \subseteq, \hat{}, \{\lambda\}, \mathbb{C})$

From now on, we consider a proper set $S \subseteq seq(U)$ with $\lambda \notin S$.

Lemma 6.2.1 $\mathbb{C}(S^\infty)$ is a solution of $X = \mathbb{C}(S) \hat{} X$.

Proof S and S^∞ satisfy the hypothesis of Lemma 6.1.2. Thus we obtain

$$\begin{aligned} \mathbb{C}(S) \hat{} \mathbb{C}(S^\infty) &= \mathbb{C}(S \hat{} S^\infty) \\ &= \mathbb{C}(S^\infty) \end{aligned}$$

The following lemma is quite obvious and will not be proved.

Lemma 6.2.2 If $S \subseteq seq(GLan(A))$ is a proper set with $\lambda \notin S$, then

i) $Inf(\mathbb{C}(S)) = \mathbb{C}(Inf(S)) \cup Inf(\mathbb{C}(Fin(S)))$

ii) $Fin(\mathbb{C}(S)) = Fin(\mathbb{C}(Fin(S)))$.

Lemma 6.2.3 $(Fin(\mathbb{C}(S)))^\infty \subseteq \mathbb{C}(S^\infty)$.

Proof Let x be in $(Fin(\mathbb{C}(S)))^\infty$.

By definition, this means

$(\exists x_1, x_2, \ldots)(x_i \in Fin(\mathbb{C}(S)))\ x = C[x_1, x_2, \ldots]$.

Using Lemma 6.2.2, we have

$$(\exists x_1, x_2, \ldots)(x_i \in Fin(\mathbb{C}(Fin(S))))\ x = C[x_1, x_2, \ldots].$$

Thus we can write

$$(\exists t_1, t_2, \ldots)(t_i \in Fin(S))(\exists x_i \in Fin(\mathbb{C}(t_i)))\ x = C[x_1, x_2, \ldots].$$

Since $t_i \in Fin(S)$ for all i, we can change the indices of the languages in each sequence by putting $t_i = L_{n_i}, \ldots, L_{n_{i+1}-1}$ and now $x_i \in Fin(\mathbb{C}(t_i))$ implies that $(\exists t_1, t_2, \ldots)(t_i \in Fin(S))\ x \in \mathbb{C}[C[t_1, t_2, \ldots]]$. Hence $x \in \mathbb{C}(S^\infty)$, using the definition of $\mathbb{C}$.

$\square$

Theorem 6.2.1 $\underset{\sim}{\mathbf{C}}(S^{\infty}) = (\underset{\sim}{\mathbf{C}}(S))^{\infty}$

Proof By Lemma 6.2.1, $\underset{\sim}{\mathbf{C}}(S^{\infty})$ is a solution of $X = \underset{\sim}{\mathbf{C}}(S) \wedge X$.
By Theorem 4.3.1, we thus have

$$\underset{\sim}{\mathbf{C}}(S^{\infty}) \subseteq (\underset{\sim}{\mathbf{C}}(S))^{\infty}.$$

Lemma 4.3.1 tells us that $(\mathrm{Fin}(\underset{\sim}{\mathbf{C}}(S)))^{*} \wedge \mathrm{Inf}(\underset{\sim}{\mathbf{C}}(S)) \subseteq \underset{\sim}{\mathbf{C}}(S^{\infty})$.
Lemma 6.2.3 yields that $(\mathrm{Fin}(\underset{\sim}{\mathbf{C}}(S))^{\infty}$ is also included in $\underset{\sim}{\mathbf{C}}(S^{\infty})$.
Using the fact that $(\underset{\sim}{\mathbf{C}}(S))^{\infty} = (\mathrm{Fin}(\underset{\sim}{\mathbf{C}}(S)))^{*} \wedge \mathrm{Inf}(\underset{\sim}{\mathbf{C}}(S)) \cup (\mathrm{Fin}(\underset{\sim}{\mathbf{C}}(S)))^{\infty}$ by Lemma 4.3.2, we conclude that $(\underset{\sim}{\mathbf{C}}(S))^{\infty} \subseteq \underset{\sim}{\mathbf{C}}(S^{\infty})$. Thus $\underset{\sim}{\mathbf{C}}(S^{\infty}) = (\underset{\sim}{\mathbf{C}}(S))^{\infty}$.

6.3 Adequacy of $\mathbf{C}$ on $(\mathrm{Rel}(D,\delta),\subseteq,\circledcirc,I,C)$

The proof of adequacy of $\mathbf{C}$ in this second quasinet does not differ very much from the one in the previous section.

We first make an observation: since we assume that $\lambda \notin S$, S^{∞} yields only infinite sequences of relations and thus $\underset{\sim}{\mathbf{C}}(S^{\infty}) \subseteq D\times\{\delta\}$. Obviously, it is also true that $(\underset{\sim}{\mathbf{C}}(S))^{\infty} \subseteq D\times\{\delta\}$. Hence we need only to compare the domain of the two relations.

Lemma 6.3.1 $\underset{\sim}{\mathbf{C}}(S^{\infty})$ is a solution of $X = \underset{\sim}{\mathbf{C}}(S) \circledcirc X$.

Proof As in Lemma 6.2.1. □

Lemma 6.3.2 $(\mathrm{Fin}(\underset{\sim}{\mathbf{C}}(S)))^{\infty} \subseteq \underset{\sim}{\mathbf{C}}(S^{\infty})$.

Proof As in Lemma 6.2.3. □

<u>Theorem 6.3.1</u> $\boldsymbol{\mathfrak{C}}(S^{\infty}) = (\boldsymbol{\mathfrak{C}}(S))^{\infty}$.

<u>Proof</u> By Lemma 6.3.1 and Theorem 5.3.1, the domain of $\boldsymbol{\mathfrak{C}}(S^{\infty})$ is included in the domain of $(\boldsymbol{\mathfrak{C}}(S))^{\infty}$.

By the proof of Theorem 5.3.1, it is obvious that $(\mathrm{Fin}(R))^{*} \circledcirc \mathrm{Inf}(R)$ is included in any solution of $X = R \circledcirc X$. In this case, we can conclude because of Lemma 6.3.1 that $(\mathrm{Fin}(\boldsymbol{\mathfrak{C}}(S)))^{*} \circledcirc \mathrm{Inf}(\boldsymbol{\mathfrak{C}}(S)) \subseteq \boldsymbol{\mathfrak{C}}(S^{\infty})$. Lemma 6.3.2 also yields that $(\mathrm{Fin}(\boldsymbol{\mathfrak{C}}(S)))^{\infty} \subseteq \boldsymbol{\mathfrak{C}}(S^{\infty})$.

Since $(\boldsymbol{\mathfrak{C}}(S))^{\infty} = (\mathrm{Fin}(\boldsymbol{\mathfrak{C}}(S)))^{*} \circledcirc \mathrm{Inf}(\boldsymbol{\mathfrak{C}}(S)) \cup (\mathrm{Fin}(\boldsymbol{\mathfrak{C}}(S)))^{\infty}$ by property 7 of ∞ operator in section 5.3, we have that $(\boldsymbol{\mathfrak{C}}(S))^{\infty} \subseteq \boldsymbol{\mathfrak{C}}(S^{\infty})$ and hence the domain of $(\boldsymbol{\mathfrak{C}}(S))^{\infty}$ is included in the domain of $\boldsymbol{\mathfrak{C}}(S^{\infty})$.

Thus $\boldsymbol{\mathfrak{C}}(S^{\infty})$ and $(\boldsymbol{\mathfrak{C}}(S))^{\infty}$ have equal domain and by the observation made before Lemma 6.3.1, they are equal. $\square$

References

Blikle [74]: An extended approach to mathematical analysis of programs, CCPAS Reports, 1974.

Blikle [76a]: An introduction to the mathematical theory of programs, Lecture notes of a course given at the University of Waterloo, 1976.

Blikle [76b]: An analysis of programs by algebraic means, in the Mathematical Foundations of Computer Science (Proc. MFCS Sem. of the Int. Math. S. Banach Center in Warsaw, Feb.1-June 26, 1974), Polish Scientific Publishers, Warsaw (to appear in 1976).

Redziejawski [72]: The theory of general events and its application to parallel programming, World Trade Corporation, IBM Nordic Laboratory, Sweden, TP 18.220, 1972.