I S S N 0250 3638

TOWARDS A NEW ALGEBRAIC FOUNDATION OF FLOWCHART SCHEME THEORY

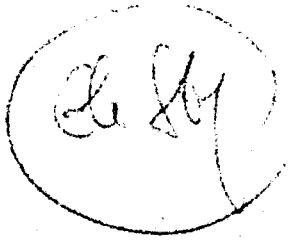
by

Virgil-Emil CĂZĂNESCU and Gh. ŞTEFĂNESCU

PREPRINT SERIES IN MATHEMATICS
No. 43/1987

INSTITUTUL
DE
MATEMATICA

INSTITUTUL NATIONAL
PENTRU CREATIE
STIINTIFICA SI TEHNICA



ISSN 0250 3638

TOWARDS A NEW ALGEBRAIC FOUNDATION OF FLOWCHART SCHEME THEORY

by

Virgil-Emil CĂZĂNESCU and Gh. ŞTEFĂNESCU

PREPRINT SERIES IN MATHEMATICS
No. 43/1987

BUCURESTI

TOWARDS A NEW ALGEBRAIC FOUNDATION OF FLOWCHART SCHEME THEORY

by

Virgil Emil CĂZĂNESCU*) and Gheorghe ŞTEFĂNESCU**)

December 1987

*) *Faculty of Mathematics, University of Bucharest, Str. Academiei 14, 70109 Bucharest, Romania*

**) *Department of Mathematics, The National Institute for Scientific and Technical Creation, Bd. Păcii 220, 79622 Bucharest, Romania.*

TOWARDS A NEW ALGEBRAIC FOUNDATION OF FLOWCHART SCHEME THEORY

Virgil Emil CĂZĂNESCU*) and Gheorghe ŞTEFĂNESCU**)

*) Faculty of Mathematics, University of Bucharest,
Str. Academiei 14, 70109 Bucharest, Romania

**) Department of Mathematics, The National Institute for Scientific and Technical Creation, Bd. Păcii 220, 79622 Bucharest, Romania.

INTRODUCTION

In the study of flowchart schemes we use a new operation called feedback (Figure 4.e) instead of the iteration to model the loops. As in the definition of the iteration appear implicitly an identification of the return points with the inputs, the use of iteration implies the use of tupling (Figure 10 bis), therefore the algebraic theories have had a main place in the study of flowchart schemes. The use of feedback permits to leave out the tupling. Our conviction is that the symmetric strict monoidal categories [9] are the most adequate algebraic structures to study acyclic flowchart schemes. To study flowchart schemes we use a symmetric strict monoidal category endowed with an adequate axiomatized feedback.

The aim of this paper is just to provide motivation. Proofs will be given elsewhere.

1. A FORMAL REPRESENTATION OF FLOWCHART SCHEMES

1.1. A representation by pairs

The usual computation processes may be represented by flowchart pictures as in Figure 1. The meaning of the picture is the usual one: start the computation beginning with the input vertex (START) and execute the statements in the order given by arrows

until an output vertex is reached; in the case a statement has more than one output arrow (exit) its execution gives at the same time the information regarding the output arrow on which the execution is continued.

The (abstract) flowchart schemes will be obtained by a double abstraction of these concrete flowchart pictures: an abstraction of statements and an abstraction of connections.

The first abstraction is easier to understand. It consists of replacing the concrete statements used to label the vertices in flowchart pictures by abstract symbols (variables). Since the statements we use may have more than one entry and one exit, the set of variables is a double-indexed set $\{X(m,n)\}_{m,n\in\mathbf{N}}$. An element $x\in X(m,n)$ is considered as a unknown computation process with m entries and n exits (a still unspecified computation process). Denote by X the disjoint union of this family of variables. Two functions $i,o : X \to \mathbf{N}$ specify the numbers of entries and of exits respectively, corresponding to a variable, [illegible].

The result of this abstraction is the usual notion of "flowchart scheme" studied in the seventies (Manna, Greibach, Kotov): An X-flowchart scheme is a finite, locally ordered, oriented graph whose vertices are coherently labelled by symbols in X. Such an abstraction of the flowchart picture in Figure 1 is given in Figure 2, where $x_1,x_3,x_4,x_5,x_6 \in X(1,1)$ and $x_2 \in X(1,2)$.

The second abstraction is more complicated, and at the present stage of the presentation only a vague definition can be given. Note that every flowchart picture can be rearranged in a normal way be putting on a first level the statements of the scheme and on a second level the connections of the scheme. For example, the scheme in Figure 2 can be arranged in a normal form as in Figure 3. In this way we can image the possibility of using a "theory" for connections. (What "theory" means will be explained later.) In our concrete case, this theory is the theory of finite functions Fn given by the family of sets

$$\mathbf{Fn}(m,n) = \{f \mid f : [m] \to [n] \text{ function}\}, \text{ for } m,n\in\mathbf{N}$$

where ~~typically~~ $[n] = \{1,2,\ldots,n\}$. An element $f\in \mathbf{Fn}(m,n)$ used as a connection indicates

the redirecting of flow of control. For the scheme in Figure 3 the connection $g \in Fn(8,7)$ is given in the following table

j	1	2	3	4	5	6	7	8
g(j)	2	3	4	5	1	6	7	3

At the abstract level we shall use for connections a "support" theory T given by a family of sets $\{T(m,n)\}_{m,n \in N}$. An element $f \in T(m,n)$ is considered as a known computation process with m entries and n exits.

The result of this double abstraction is the concept of representation of an X-flowchart scheme over T. It can be defined as follows. For x in the free monoid X^* we denote by $|x|$ the length of the word x, and for $j \in [|x|]$ we denote by x_j the j-th letter of x. Hence $x = x_1 x_2 \ldots x_{|x|}$. Also we use the notation: $i(x) = i(x_1) + \ldots + i(x_{|x|})$ and $o(x) = o(x_1) + \ldots + o(x_{|x|})$. A representation of an X-flowchart scheme over T with m entries and n exits is defined as a pair

$$F = (x, f)$$

where $x = x_1 \ldots x_{|x|} \in X^*$ specifies the vertices of the scheme, ordered in a linear way, and $f \in T(m + o(x), n + i(x))$ specifies the connection of the scheme. The scheme in Figure 3 may be represented as $(x_1 x_2 x_3 x_4 x_5 x_6, g)$, where $g \in Fn(8,7)$ is the function defined above,

It must be emphasized that there may be more representations which correspond to a flowchart scheme. The difference between these representations is generated by the way in which the statements of the scheme are linearly ordered as a string $x \in X^*$.

We denote by $Fl_{X,T}$ the set of representations of X-flowchart schemes over T. More precisely,

$$Fl_{X,T}(m,n) = \{(x,f) \mid x \in X^*, f \in T(m + o(x), n + i(x))\} .$$

1.2. Operations

While the above representation of schemes by pairs F = (x,f) is convenient for theoretical purposes, for practical purposes it is inconvenient in the sense that it does

not show how the scheme F can be obtained from the components x and f of its representation. To fill in this gap we introduce here operations on flowchart schems.

If we look at the normal representation of schemes given in Figure 3, then we can deduce that every scheme can be obtained from the components of its representations using the operations in Figure 4, called sum, composition and feedback. More precisely, a flowchart scheme F represented by the pair $(x,f)\in Fl_{X,T}(m,n)$ can also be represented by a formal expression

$$((1_m + x_1 + \ldots + x_{|x|}) \cdot f)\uparrow^{i(x)},$$

where $\uparrow^{i(x)}$ denotes the application of the feedback by $i(x)$ times, and $1_m \in T(m,m)$ is the scheme without (internal) vertices which directly connects the i-th entry on the i-th exit.

1.2.1. The elements of T are considered as particular schemes having only connections between entries and exits (i.e., without internal vertices). Therefore, if the operations above have sense in $Fl_{X,T}$ then they must be defined in T, too. The usual flowchart schemes have as support theory a subtheory of the theory of finite relations **Rel** defined by the family of sets

$$\mathbf{Rel}(m,n) = \{r \mid r \subseteq [m]\times[n] \text{ relation}\}, \text{ for } m,n \in \mathbf{N}.$$

Here the operations in Figure 4 have the following meaning.

The operations in **Rel**. For $r \subseteq [m]\times[n]$ and $r' \subseteq [p]\times[q]$ the sum $r + r' \subseteq [m + p]\times[n + q]$ is defined by

$$r + r' = r \cup \{(m + j, n + j') \mid (j,j') \in r'\}.$$

For $r \subseteq [m]\times[n]$ and $r' \subseteq [n]\times[p]$ the composite $r \cdot r' \subseteq [m]\times[p]$ is the usual one defined by

$$r \cdot r' = \{(j,j') \mid \text{there exists } u \in [n] \text{ such that } (j,u) \in r \text{ and } (u,j') \in r'\}.$$

For $r \subseteq [m + 1]\times[n + 1]$ the feedback $r\uparrow \subseteq [m]\times[n]$ is defined by

$$r\uparrow = \{(j,j') \mid (j,j') \in r \text{ or } [(j,n + 1) \in r \text{ and } (m + 1,j') \in r]\}.$$

$\in [\omega]\times[\omega]$

(The meaning of $1_m \in \mathbf{Rel}(m,m)$ is clear: $1_m = \{(j,j) \mid j \in [m]\}$. In the sequel we shall use some distinghuished morphisms of the support theory T, namely $m \leftrightarrow n \in T(m+n, n+m)$, $m \nabla m \in T(m+m, m)$, $0_m \in T(0,m)$, $\perp_n \in T(n,0)$ and $m \Delta m \in T(m, m+m)$, whose meaning in **Rel** is: $m \leftrightarrow n = \{(j, n+j) \mid j \in [m]\} \cup \{(m+j, j) \mid j \in [n]\}$; $m \nabla m = \{(j,j) \mid j \in [m]\} \cup \{(m+j, j) \mid j \in [m]\}$; $0_m = \emptyset$, $\perp_n = \emptyset$, $m \Delta m = \{(j,j) \mid j \in [m]\} \cup \{(j, m+j) \mid j \in [m]\}$.)

Note. The subtheory of partial, finite functions in **Rel**, denoted by **Pfn** and defined by the family of sets

$$\mathbf{Pfn}(m,n) = \{f \mid f : [m] \leftrightarrow [n] \text{ partially defined function}\}, \text{ for } m,n \in \mathbf{N}$$

is closed under the aforementioned operations. The theory **Fn** defined in §1.1 is not closed under feedback, hence it is inconvenient to use **Fn** as a support theory for deterministic flowchart schemes (since $\mathbf{Fn}(1,0) = \emptyset$, for the unique function $f \in \mathbf{Fn}(2,1)$ we have $f^{\dagger} \notin \mathbf{Fn}(1,0)$). The use of **Pfn** as support theory in the deterministic case is equivalent to the extension of the concept of usual flowchart scheme to the concept of partial flowchart scheme. A partial flowchart scheme is obtained from a usual flowchart scheme by deleting some arrows, and one interprets such an absence of arrow as a connection to an endless loop. For the sake of contrast, sometimes the usual flowchart schemes (over **Fn**) will be called complete flowchart schemes.

1.2.2. Conversely, in the following section we shall see that it is easy to extend the operations in Figure 4 from T to $\mathrm{Fl}_{X,T}$, supposing T "contains" bijective, finite functions.

We collect these facts as the following slogan:

> In order to define algebra $\mathrm{Fl}_{X,T}$ we have to specify:
> - a double indexed set X;
> - a support theory T containing finite bijections and
> endowed with operations acting as in Figure 4.

1.3. The algebra of representations ($Fl_{X,T}$)

In order to extend our operations from T to $Fl_{X,T}$, T has to contain some distinghished elements $m \leftrightarrow n \in T(m+n, n+m)$ representing the "block transpositions" where $m,n \in \mathbf{N}$, i.e. $\overset{m}{\underset{n}{\times}}\overset{n}{\underset{m}{}}$. In the theory Rel the morphisms $m \leftrightarrow n$ were defined in §1.2.1.

The flowchart scheme in the normal form corresponding to a representation of X-flowchart over T $F = (x,f)$ is illustrated in Figure 5. The operations on flowchart scheme representations can be obtained by applying first the operations in Figure 4 on the pictures corresponding to the given representations, then by rearranging the obtained result in an adequate, normal form, and finally by writing the representation associated to the final picture.

The sum of two normal flowchart schemes illustrates in Figure 6.a can be rearranged in the normal form given in Figure 6.b. Hence, we can formally define the sum of two representations $(x,f) \in Fl_{X,T}(m,n)$ and $(y,g) \in Fl_{X,T}(p,q)$ by

$$(x,f) + (y,g) = (x \dot{+} y, (1_m + p \leftrightarrow o(x) + 1_{o(y)})(f+g)(1_n + i(x) \leftrightarrow q + 1_{i(y)})) .$$

The composite of two normal flowchart schemes illustrated in Figure 7.a can be rearranged in the normal form given in Figure 7.b. Hence, we can formally define the composite of two representations $(x,f) \in Fl_{X,T}(m,n)$ and $(y,g) \in Fl_{X,T}(n,p)$ by

$$(x,f) \cdot (y,g) = (x \dot{+} y, (f + 1_{o(y)})(1_n + i(x) \leftrightarrow o(y))(g + 1_{i(x)})(1_p + i(y) \leftrightarrow i(x))).$$

The feedback of a normal flowchart scheme illustrated in Figure 8.a can be rearranged in the normal form given in Figure 8.b. Hence, we can formally define the feedback of a representation $(x,f) \in Fl_{X,T}(m+1, n+1)$ by

$$(x,f)\uparrow = (x, [(1_m + o(x) \leftrightarrow 1) f (1_n + 1 \leftrightarrow i(x))]\uparrow) .$$

Let us mention that the embeddings of X and T into $Fl_{X,T}$ are given by the following applications:

$E_T(f) = (\varepsilon, f)$ for $f \in T(m,n)$, where $\varepsilon \in X^*$ is the empty word;

$E_X(x) = (x, m \leftrightarrow n)$ for $x \in X(m,n)$.

(The last equality can be extended to embed X^* into $Fl_{X,T}$: $E_X(x) = (x, l(x) \leftrightarrow o(x))$ for $x \in X^*$.)

We do not insist on the algebraic rules satisfied by the flowchart scheme representations since this study is interesting only from a technical viewpoint. We only mention that an algebraic structure, called flow, has been singled out, which is preserved by passing from T to $Fl_{X,T}$, and that $Fl_{X,T}$ satisfies a universal property partially similar to that satisfied by polynomials (those interpretations of X and T in a flow that satisfy a certain supplementary condition can be naturally extended in a unique way to $Fl_{X,T}$).

1.4. Flownomials, flow-calculus

As we pointed out in §1.2 a flowchart scheme represented by a picture in a normal form may also be represented by a formal expression of the particular form $((1_m + x_1 + \ldots + x_k) \cdot f)\uparrow^{i(x_1)+\ldots+i(x_k)}$. The final form of the calculus is obtained by allowing arbitrary formal expressions written with "+", "·" and "↑".

Flownomial expressions. Let X and T be as above. Define the sets $EXP_{X,T}(m,n)$ of flownomial X-expressions over T of type $m \to n$ as follows:

(i) atomic elements $x \in X(m,n)$ and $f \in T(m,n)$ are flownomial expresions of the type $m \to n$;

(ii) ~~combine flownomial~~ compound expressions: if $F^1 : m \to n$, $F^2 : p \to q$, $F^3 : n \to q$ and $F : m+1 \to n+1$ are flownomial expressions of the indicated type then $F^1 + F^2 : m+p \to n+q$, $F^1 \cdot F^3 : m \to q$ and $F\uparrow : m \to n$ are flownomial expressions of the indicated type.

(iii) all flownomial expressions are obtained by using rules (i) and (ii).

A flownomial expression of the form $((1_m + x_1 + \ldots + x_k) \cdot f)\uparrow^r$, where $r = i(x_1) + \ldots + i(x_k)$ is said to be in a normal form; in the sequel we shall use the following standard notation: $\underline{x} = \sum_{j \le k} x_j$, $i(\underline{x}) = \sum_{j \le k} i(x_j)$, $o(\underline{x}) = \sum_{j \le k} o(x_j)$; $\underline{x}' = \sum_{j \le k'} x'_j$ etc. When T is closed with respect to +, · and ↑ and contains the block transpositions $m \leftrightarrow n$, every flownomial expression can be brought to a normal form by using the

following rules:

(R1) replace subexpressions involving only elements in T by the corresponding value computed in T;

(R2) the normal form of $f \in T(m,n)$ is $(1_m \cdot f)\uparrow^n$ and of $x \in X(m,n)$ is $((1_m + x) \cdot m \leftrightarrow n)\uparrow^m$;

(R3) the normal form of $((1_m + x) \cdot f)\uparrow^{j(x)} + ((1_p + x') \cdot f')\uparrow^{j(x')}$ is $[(1_{m+p} + x + x')][(1_m + p \leftrightarrow o(x) + 1_{o(x')})(f + f')(1_n + i(x) \leftrightarrow q + 1_{i(x')})]\uparrow^{i(x+x')}$;

(R4) the normal form of $((1_m + x) \cdot f)\uparrow^{i(x)} \cdot ((1_n + x') \cdot f')\uparrow^{i(x')}$ is $[(1_m + x + x')][(f + 1_{o(x')})(1_n + i(x) \leftrightarrow o(x'))(f' + 1_{i(x)})(1_q + i(x') \leftrightarrow i(x))]]\uparrow^{i(x+x')}$;

(R5) the normal form of $((1_{m+1} + x) \cdot f)\uparrow^{j(x)}\uparrow$ is $[(1_m + x)][(1_m + o(x) \leftrightarrow 1)f(1_n + 1 \leftrightarrow f(x))]\uparrow)\uparrow^{j(x)}$.

Using these rules every flownomial expression can be brought to a unique normal form, hence flownomial expressions in normal form give a complete and independent system of representations for the congruence relation R generated by the rules (R1 - 5) in the algebra of expressions $EXP_{X,T}$. In addition, it can be proved that the algebra of representations $Fl_{X,T}$ is isomorphic to the quotient algebra $EXP_{X,T}/R$. Consequently, in this enlarged frame we have the following identification:

representations by pairs = flownomial expressions in normal form.

The examples we shall give in this paper are related to the flowchart scheme in Figure 9.a. They use the variables $x \in X(1,3)$, $y \in X(1,1)$ and $z \in X(2,1)$. The support theory T is the theory of finite partial functions, i.e., T = **Pfn**. An element $f \in$ **Pfn**(m,n) is represented by the sequence of its values, i.e., $(\bar{f}(1),\bar{f}(2),\ldots,\bar{f}(m))_n$, where $\bar{f}(i)$ = "if f(i) = undefined then $\perp$ else f(i)", for $i \in [m]$. For instance: the function $f \in$ **Pfn**$(4,4)$ given by f(1) = 1, f(2) = f(4) = 3, f(3) = 2 is represented by $(1,3,2,3)_4$; $(1,\perp)_3$ represents the function $f \in$ **Pfn**$(2,3)$ given by f(1) = 1 and f(2) = undefined. (This representation of finite partial functions is not elegant, and is similar to the representation of natural numbers by bars, i.e., 7 = ||||||| etc.)

Example. In this example we prove that the following identity holds in flow-calculus:

$$[(1 \vee 1 \cdot x)(1_2 + x)]\uparrow = [(1_1 + x + x)(5,1,2,6,3,4,5)_6]\uparrow^2 ,$$

i.e., the normal form of the left-hand-side expression is the right-hand-side expression.
Note that the left-hand-side expression expresses the top of the figure 3.
Indeed:

(5)

$$\textcircled{1}\ x = [(1_1 + x)1 \leftrightarrow 3]\uparrow = [(1_1 + x)(4,1,2,3)_4]\uparrow ;$$

$$\textcircled{2}\ 1 \vee 1 \cdot x = [1_2 \cdot (1,1)_1]\uparrow^0 \cdot [(1_1 + x)(4,1,2,3)_4]\uparrow^1 =$$

$$= \{(1_2 + x)[((1,1)_1 + 1_3)(1_1 + 0 \leftrightarrow 3)((4,1,2,3)_4 + 1_0)(1_3 + 1 \leftrightarrow 0)]\}\uparrow^{0+1} =$$

$$= [(1_2 + x)(4,4,1,2,3)_4]\uparrow ;$$

$$\textcircled{3}\ 1_2 + x = [1_2 \cdot (1,2)_2]\uparrow^0 + [(1_1 + x)(4,1,2,3)_4]\uparrow^1 =$$

$$= \{(1_3 + x)[(1_2 + 1 \leftrightarrow 0 + 1_3)((1,2)_2 + (4,1,2,3)_4)(1_2 + 0 \leftrightarrow 3 + 1_1)]\}\uparrow^{0+1} =$$

$$= [(1_3 + x)(1,2,6,3,4,5)_6]\uparrow ;$$

$$\textcircled{4}\ (1 \vee 1 \cdot x)(1_2 + x) = [(1_2 + x)(4,4,1,2,3)_4]\uparrow^1 \cdot [(1_3 + x)(1,2,6,3,4,5)_6]\uparrow^1 =$$

$$= \{(1_2 + x + x)[((4,4,1,2,3)_4 + 1_3)(1_3 + 1 \leftrightarrow 3)((1,2,6,3,4,5)_6 + 1_1)(1_5 + 1 \leftrightarrow 1)]\}\uparrow^2 =$$

$$= [(1_2 + x + x)(6,6,1,2,7,3,4,5)_7]\uparrow^2 ;$$

$$\textcircled{5}\ [(1 \vee 1 \cdot x)(1_2 + x)]\uparrow = [(1_2 + x + x)(6,6,1,2,7,3,4,5)_7]\uparrow^2\uparrow =$$

$$= \{(1_1 + x + x)[(1_1 + 6 \leftrightarrow 1)(6,6,1,2,7,3,4,5)_7(1_4 + 1 \leftrightarrow 2)]\uparrow\}\uparrow^2 =$$

$$= [(1_1 + x + x)(5,1,2,6,3,4,5)_6]\uparrow^2 .$$

2. SEMANTIC MODELS

The basic model for the study of semantics of deterministic flowchart schemes has been introduced by C.C. Elgot. It consists in the following: Let S be the set of value-vectors denoting the states of memory in a computing device (the values in the registers of memory). A deterministic flowchart scheme F with m entries and n exits is interpreted via an interpretation I as a partial function $F_I : [m] \times S \leftrightarrow [n] \times S$ with the meaning that "$F_I(j,s)$ is defined and equal to (j',s')" iff "if the execution of the program obtained by interpreting F via I begins at entry j of the program with initial state of memory s, then the execution halts at exit j' of the program, the resulted state of

memory being s'."

If we denote by

$$\mathbf{Pfn}_S(m,n) = \{ f \mid f : [m] \times S \to [n] \times S \text{ partial function} \}, \text{ for } m,n \in \mathbf{N}$$

we obtain a "theory", in a vague sense, $\mathbf{Pfn}_S$ which is the basic semantic model in the deterministic case.

Note that in the particular case when S has exactly one element $\mathbf{Pfn}_S$ can be identified with Pfn defined above (§1.2.1). In this case the stress is laid on flow of control, whereas the memory state remains unchanged.

In a similar way has been introduced the basic semantic model in the nondeterministic case. A nondeterministic flowchart scheme F with m entries and n exits is interpreted, via an interpretation I, as a relation $F_I \subseteq ([m] \times S) \times ([n] \times S)$ with the meaning that "$((j,s),(j',s')) \in F_I$" iff "if the execution of the program obtained by interpreting F via I begins at entry j of the program with initial state of memory s, then the execution may halts, on one variant, at exit j' of the program, the resulted state of memory being s'." If we denote by

$$\mathbf{Rel}_S(m,n) = \{ r \mid r \subseteq ([m] \times S) \times ([n] \times S) \}, \text{ for } m,n \in \mathbf{N}$$

then we obtain a theory $\mathbf{Rel}_S$ which is the basic semantic model in the nondeterministic case.

As above, in the particular case when S has exactly one element, $\mathbf{Rel}_S$ can be identified with Rel defined in §1.2.1.

In $\mathbf{Rel}_S$ many operations and algebraic structures may be considered. The operations which interest us (sum, composition and feedback) have the following definitions.

For $r \in \mathbf{Rel}_S(m,n)$ and $r' \in \mathbf{Rel}_S(p,q)$ the sum $r + r' \in \mathbf{Rel}_S(m+p, n+q)$ is defined by

$$r + r' = r \cup \{((m+j,s),(n+j',s')) \mid ((j,s),(j',s')) \in r'\}.$$

For $r \in \mathbf{Rel}_S(m,n)$ and $r' \in \mathbf{Rel}_S(n,p)$ the composite $r \cdot r' \in \mathbf{Rel}_S(m,p)$ is the usual one, defined by

$$r \cdot r' = \{((j,s),(j',s')) \mid \exists\, (j_o,s_o) \in [n] \times S \text{ with } ((j,s),(j_o,s_o)) \in r \text{ and } ((j_o,s_o),(j',s')) \in r'\}.$$

In order to define the feedback let us note that every relation $v \in \mathbf{Rel}_S(m,n)$ is given by a family of relations $v_{i,j} \subseteq S \times S$, for $i \in [m]$, $j \in [n]$, where $v_{i,j} = \{(s,s') \mid ((i,s),(j,s')) \in v\}$. Denote by v^* the reflexive-transitive closure of a relation $v \subseteq S \times S$, i.e., $v^* = 1_S \cup v \cup v^2 \ldots$, where $1_S = \{(s,s) \mid s \in S\}$. Using these facts, for $r \in \mathbf{Rel}_S(m+1, n+1)$ the feedback $r\uparrow \in \mathbf{Rel}_S(m,n)$ is defined by

$$(r\uparrow)_{i,j} = r_{i,j} \cup \left(r_{i,n+1} \cdot \overbrace{r_{m+1,n+1}}^{*} \cdot r_{m+1,j}\right), \text{ for } i \in [m],\ j \in [n].$$

We finish this section be defining the natural embedding of **Rel** in $\mathbf{Rel}_S$, given by the application

$$r \longmapsto \{((i,s),(j,s)) \mid (i,j) \in r,\ s \in S\}.$$

Particularly, this application shows how various classes of finite relations in **Rel** (bijective functions, injective functions, etc.) can be thought of as being elements in an arbitrary $\mathbf{Rel}_S$.

3. SYNTACTIC MODELS

In order to formalize some aspects regarding the study of flowchart schemes: isomorphism, accessibility, reduction, minimization with respect to the input (step-by-step) behaviour, coaccessibility, minimization with respect to the input-output (step-by-step) behaviour, the flow-calculus, introduced in § 1.4, has to be augmented with some rules of identification for flownomials. It is an important test for this calculus whether the identifications corresponding to the natural aforementioned properties can be (easily) defined. This task can be done. The most interesting fact is that for the above properties there is a unique rule of identification (i.e., the equivalence relation generated by simulation) that has as particular cases the identification rules necessary for each property.

3.1. The simulation relation. Let $F = ((1_m + x_1 + \ldots + x_k) \cdot f)\uparrow^r$ and $F' = ((1_m + x'_1 + \ldots + x'_{k'}) \cdot f')\uparrow^{r'}$ be two flownomial expressions in normal form, having the same type $m \to n$, and $y \in \mathbf{Rel}(k,k')$ (think of y as a relation between the statements

$x_1,\ldots,x_k$ of F and the statements $x'_1,\ldots,x'_{k'}$ of F'). We say that F and F' are in simulation via y (in symbols $F \rightarrow_y F'$) if:

(i) $(j,j') \in y$ implies $x_j = x'_{j'}$;

(ii) the natural "block" extensions of y to the inputs of the statements, denoted i(y), and to the outputs of the statements, denoted o(y), fulfil

$$f \cdot (1_n + i(y)) = (1_m + o(y)) \cdot f' \ .$$

(this equality makes sense when T is closed under composition and the relations i(y), o(y) are "embedded" in T).

Let us explain in more details what we mean by "block" extensions and by "embedding". Suppose we are given the sequences $x_1,\ldots,x_k$ and $x'_1,\ldots,x'_{k'}$, and the relation $y \in \mathbf{Rel}(k,k')$ satisfying (i). Define the block extension of y to the inputs of the statements $i(y) \in \mathbf{Rel}(\Sigma_{j\leq k} i(x_j), \Sigma_{j'\leq k'} i(x'_{j'}))$ as follows: An $s \in [\Sigma_{j\leq k} i(x_j)]$ can be written in a unique way as $s = \Sigma_{j<\alpha(s)} i(x_j) + \beta(s)$, where $\alpha(s) \in [k]$ and $\beta(s) \in [i(x_{\alpha(s)})]$ (read this: s is the input that has the number $\beta(s)$ of the statement that has the number $\alpha(s)$ in the sequence $x_1,\ldots,x_k$). Similarly, every $s' \in [\Sigma_{j'\leq k'} i(x'_{j'})]$ can be written as $s' = \Sigma_{j'<\alpha'(s')} i(x'_{j'}) + \beta'(s')$. Now the relation i(y) is defined by

$$i(y) = \left\{ (s,s') \mid (\alpha(s), \alpha'(s')) \in y \text{ and } \beta(s) = \beta'(s') \right\} .$$

The block extension of y to outputs $o(y) \in \mathbf{Rel}(\Sigma_{j\leq k} o(x_j), \Sigma_{j'\leq k'} o(x'_{j'}))$ is defined in a similar way.

At a first stage we can translate "embedding" by "inclusion". Later on we shall give a more general meaning to "embedding" that contains, as a particular case, the embedding of **Rel** in $\mathbf{Rel}_s$ defined in § 2.[8]

The meaning of $F \rightarrow_y F'$ depends on the type of y and will be given below for each particular class of relations used for y. We only mention here that this notion of simulation is the result of an historical process aiming to formalize some flowchart scheme properties. Initially we had found that isomorphism and reduction can be captured using simulations via bijective and surjective function, respectively. Later on we found that accessibility could also be modelled by simulation, namely by simulation

8) See also the Appendix below.

via injective functions, and the input (step-by-step) behaviour could be captured using simulations via functions. Concessibility can be modelled by simulation via converses of injective functions, and the input-output (step-by-step) behaviour, in the deterministic case, can be captured using simulations via partially defined functions.

3.2. Equivalences generated by simulations. For a subset A of **Rel** let us denote by $\rightarrow_A$ the simulation via A-relations, namely "$F \rightarrow_A F'$ iff there exists γ in A such that $F \rightarrow_\gamma F'$ ", and by $=_A$ the equivalence relation generated by $\rightarrow_A$. By the above comments it follows that the most interesting subsets A of **Rel** are: **Bi** (bijective functions), **In** (injective functions), **Sur** (surjective functions), **Pn** (functions), $\mathbf{In}^{-1}$ (converses of injective functions), **Pfn**, $\mathbf{Sur}^{-1}$ (converses of surjective functions), and **Rel**.

In the case when A is closed with respect to sum and composition, $=_A$ is a congruence relation, hence the operations can be defined in the quotient structure $Fl_{X,T}/=_A$. The resulted algebraic structures $Fl_{X,T}/=_A$, for certain X,T and A, are the basic syntactic models for flowchart scheme theory.

4. FLOWCHART SCHEMES

In section §1.1 we emphasized that more representations by pairs (or equivalently, flownomial expressions in normal form) corresponds to a flowchart scheme, the difference being generated by the way the statements of the scheme are linearly ordered. This observation suggests the identification of a flowchart scheme with the class of its representations. The mathematical formulation of the fact that two representations represent the same flowchart-picture is capturated by the simulation via bijective functions.

4.1. The simulation via bijective functions (isomorphism). Suppose the support theory T "contains" bijective finite functions. The meaning of the wording "contain" will be specified later on. In the usual case T is a subtheory of **Rel**, hence the meaning is

clear: $T \supseteq Bi$.

The definition of simulation via bijective functions is obtained form the general definition, given in §3.1, using for γ morphisms in **Bi**.

In the particular case when T is a subtheory of **Rel** the meaning of the simulation "$F \to_\gamma F'$ with γ in **Bi**" is "F and F' represent the same flowchart scheme, the bijection γ doing the connection between the linearly ordered statements of F and of F'." Therefore, the simulation via bijective functions can be named "isomorphism".

Now we turn back to the general setting. For the congruence relation $=_{Bi}$ generated by $\to_{Bi}$, the following equivalent characterizations can be given:

(i) $=_{Bi} = \to_{Bi}$ (hence $\to_{Bi}$ is a congruence);

(ii) $=_{Bi}$ is the congruence relation generated by the identifications $(\leftrightarrow X)\ (x + x') \cdot n \leftrightarrow q =_{Bi} m \leftrightarrow p \cdot (x' + x)$, where $x \in X(m,n)$ and $x' \in X(p,q)$ (see Figure 10.a).

4.2. The mathematical concept of flowchart schemes. The above facts show that, in the case $T \subseteq$ **Rel**, a flowchart scheme can be identified with an element in the quotient structure $Fl_{X,T}/=_{Bi}$. Generalizing, we say

> the elements in a $Fl_{X,T}/=_{Bi}$ are (abstract) flowchart schemes.

4.3. The algebra of flowchart schemes (biflow). We had selected some identities, written in terms of "+", "·", "↑", 1_m and $m \leftrightarrow n$, and satisfied by flowchart schemes, in order to define an algebraic structure, called biflow. The identities are listed in table 1 and illustrated in Figure 11. The main point is that this set of identities is complete, i.e. they suffice to prove that flownomial expressions over **Rel**, which represent the same flowchart scheme, are equal. Consequently, the identities, listed in table 1, completely characterizes flowchart schemes from the algebraic point of view.

In more details, a **biflow** B is an abstract structure given by:

a family of sets $\{B(m,n)\}_{m,n \geq 0}$; the distinguished morphisms $1_n \in B(n,n)$,

$m \leftrightarrow n \in B(m + n, n + m)$; three operations: composition $\cdot : B(m,n) \times B(n,p) \to B(m,p)$, sum $+ : B(m,n) \times B(p,q) \to B(m + p, n + q)$ and feedback $\uparrow : B(m + 1, n + 1) \to B(m,n)$

and satisfying the identities listed in table 1. The axioms (B1-6) show that a biflux B is a strict monoidal category; (B7-10) show that B is a symmetric strict monoidal category and the finite bijective functions are embedded in B; (B11-15) axiomatize the feedback.

(B1) $(fg)h = f(gh)$	(B9) $m \leftrightarrow (n + p) = (m \leftrightarrow n + 1_p)(1_n + m \leftrightarrow p)$
(B2) $1_m f = f = f 1_n$	(B10) $(f + g) \cdot n \leftrightarrow q = m \leftrightarrow p \cdot (g + f)$
(B3) $(f + g) + h = f + (g + h)$	for $f : m \to n$, $g : p \to q$
(B4) $1_0 + f = f = f + 1_0$	(B11) $f(g\uparrow^p)h = ((f + 1_p)g(h + 1_p))\uparrow^p$
(B5) $1_m + 1_n = 1_{m+n}$	(B12) $(f + g)\uparrow^p = f + g\uparrow^p$
(B6) $(f + g)(u + v) = fu + gv$	(B13) $(f(1_n + g))\uparrow^p = ((1_m + g)f)\uparrow^q$
for $m \xrightarrow{f} n \xrightarrow{u} p$, $m' \xrightarrow{g} n' \xrightarrow{v} p'$	for $f : m + p \to n + q$, $g : q \to p$
(B7) $m \leftrightarrow n \cdot n \leftrightarrow m = 1_{m+n}$	(B14) $1_1\uparrow = 1_0$
(B8) $0 \leftrightarrow n = 1_n = n \leftrightarrow 0$	(B15) $1 \leftrightarrow 1\uparrow = 1_1$

Table 1. These axioms define a biflow

Semantic models: $\mathbf{Rel}_s$ and all of its subtheories, which contain the embedding of **Bi** in $\mathbf{Rel}_s$ (cf. §2), are biflows. Particularly, **Bi**, **In**, **PSur** (partial, surjective functions) **Pfn** and **Rel** are biflows.

Syntactic models: If T is a biflow, then $Fl_{X,T}/_{=_{Bi}}$ is a biflow.

Generally, the support theory T for the flowchart schemes which interest us, has at least a structure of biflow. Since **Bi** is an initial biflow (in the sense of category theory: for every biflow B there exists a unique morphism of biflows from **Bi** to B), the initial wording "the support theory T contains bijections" gets a precise meaning, when T is a biflow.

4.4. The universal property. In order to get an interpretation of flownomial expressions in $EXP_{X,T}$ (or representations in $Fl_{X,T}$) in a biflow B we have to interpret the variable in X using a rank-preserving application $I_X : X \to B$ (i.e., $x \in X(m,n) \mapsto I_X(x) \in B(m,n)$) and the morphisms in T using a morphism of biflows $I_T : T \to B$ (i.e., I_T is given by a family of applications $I_T : T(m,n) \to B(m,n)$ which preserve the constants 1_m, $m \leftrightarrow n$ and the operations "+", "·", and "$\uparrow$"). Now the interpretation of a flownomial expression in normal form $F = ((1_m + x_1 + \ldots + x_k) \cdot f)\uparrow^r \in EXP_{X,T}(m,n)$ is $(I_X,I_T)^f(F) \in B(m,n)$, given by

$$(I_X,I_T)^f(F) = ((1_m + I_X(x_1) + \ldots + I_X(x_k)) \cdot I_T(f))\uparrow^r .$$

(Of course, the restriction to normal form is inessential.)

The above formula makes sense in each abstract structure B endowed with 1_m, "+", "·", and "$\uparrow$". We have taken a biflow B in order that the interpretation $(I_X,I_T)^f$ commute with the operations and in order that the $=_{Bi}$ - equivalent flownomial expressions have the same interpretation. The latter statement shows that the extension $(I_X,I_T)^f$ makes sense for $Fl_{X,T}/=_{Bi}$ too, and in that case we denote the corresponding application by $(I_X,I_T)^{bf} : Fl_{X,T}/=_{Bi} \to B$.

In the usual cases T is a subtheory of **Rel**, B is a subtheory in a $\mathbf{Rel}_s$, I_X gives the semantics for each statement $x \in X$, and I_T is the restriction to T of the embedding of **Rel** into $\mathbf{Rel}_s$. In this cases the interpretation $(I_X,I_T)^f(F)$ gives the behaviour of the program obtained by interpreting via I_X the flowchart scheme corresponding to the flownomial expression F.

Let (E_X^b, E_T^b) be the embedding of (X,T) into $Fl_{X,T}/=_{Bi}$ obtained using the embedding (E_X, E_T) of (X,T) in $Fl_{X,T}$, defined in §1.3, and the canonical projection from $Fl_{X,T}$ to $Fl_{X,T}/=_{Bi}$. The universal property satisfied by $Fl_{X,T}/=_{Bi}$ is similar to that satisfied by the polynomials, namely

> "for every biflow B and every interpretation (I_X, I_T) of (X,T) in B there exists a unique morphism of biflows $I^{bf} : Fl_{X,T}/=_{Bi} \to B$ (namely, $(I_X, I_T)^{bf}$ defined above) such that $E_X^b \cdot I^{bf} = I_X$ and $E_T^b \cdot I^{bf} = I_T$."

In a categorical language this property shows that $Fl_{X,T}/=_{Bi}$ is the coproduct of the biflow T and the one freely generated by X in the category of biflows.

4.5. Bi-flow-calculus. The calculus with flownomials associated to flowchart schemes, called bi-flow-calculus, is obtained by adding to the rules (R1-5), that define the flow-calculus (i.e., the calculus for representations introduced in §1.4), the rule which consists in the identification of $=_{Bi}$ - echivalent flownomial expressions.

Another method to define the same bi-flow-calculus is to consider flownomial X - expresions over T together with the algebraic rules that define a biflow. More precisely, the calculus is defined by the rule (R1) in §1.4 and (B1-4, B6, B10-13) in table 1. (Since T is a biflow, the other rules (B5, B7-9, B14-15) are covered by (R1).)

Example. In this example we shall prove that the following identity holds in bi--flow-calculus

$$(1V1 \cdot x(1_1 + y + x))\uparrow (1_3 + y)(1,3,2,3)_3 = (1V1 \cdot x(1_2 + x))\uparrow(1,3,2,4)_4(1_2 + (y + y)1V1) .$$

(a) Proof. Using normal forms: As in example in §1.4 the normal form of the left-hand side expression is $NF_1 = [(1_1 + x + y + x + y)(4,1,5,6,3,2,7,4,3)_7]\uparrow^4$ and of the right-hand side expression is $NF_2 = [(1_1 + x + x + y + y)\ (4,1,6,5,2,7,4,3,3)_7]\uparrow^4$. We shall prove that $NF_1 \rightarrow_y NF_2$ for the bijection $y = (1,3,2,4)_4$. Note that y preserves the statements with respect to the sequences (x,y,x,y) and (x,x,y,y), hence condition (i) in definition §3.1 holds. The extension of y to inputs is $(1,3,2,4)_4$ and to outputs is $(1,2,3,7,4,5,6,8)_8$. Since $(4,1,5,6,3,2,7,4,3)_7(1_3 + (1,3,2,4)_4) = (4,1,6,5,3,2,7,4,3)_7 = (1_1 + (1,2,3,7,4,5,6,8)_8) \cdot (4,1,6,5,2,7,4,3,3)_7$ condition (ii) in definition §3.1 holds. too.

(b) Proof. Using the algebraic rules (without marking the application of the rules (R1) and (B1-4)):

$$(1V1 \cdot x(1_1 + y + x))\uparrow (1_3 + y)(1,3,2,3)_3 =$$

$$= [1V1 \cdot x(1_2 + x)((1_1 + y + 1_2) + 1_1)]\uparrow (1_3 + y)(1,3,2,3)_3 \qquad \text{by B6}$$

$$= (1V1 \cdot x(1_2 + x))\uparrow (1_1 + y + 1_2)(1_3 + y)(1,3,2,3)_3 \qquad \text{by B11}$$

$= (1V1 \cdot x(1_2 + x)) \uparrow (1_1 + y + 1_1 + y)(1_1 + 1 \leftrightarrow 1 + 1_1)(1_2 + 1V1)$ by B6

$= (1V1 \cdot x(1_2 + x)) \uparrow (1_1 + 1 \leftrightarrow 1(1_1 + y) + y)(1_2 + 1V1)$ by B6, B10

$= (1V1 \cdot x(1_2 + x)) \uparrow (1,3,2,4)_4(1_2 + (y + y) \cdot 1V1)$ by B6 .

5. ACCESSIBILITY [*]

A flowchart scheme is a notation of a sequential computation process. In the process of computation only the vertices that can be reached by paths going from inputs matter; these vertices form the accessible part of the scheme. Here we regard as equivalent two flowchart schemes that have the same accessible part. In a formal approach accessibility is captured by simulation via injective functions.

5.1. The simulation via injective functions; the resulted congruence. Suppose that the support theory T "contains" injective finite functions. In the case $T \subseteq$ **Rel** this means $T \supseteq$ **In**.

The definition of simulation via injective functions is obtained from the general definition, given in §3.1, using for y morphisms in **In**.

In the particular case when T is a subtheory of **Rel** the meaning of the simulation "$F \xrightarrow[y]{} F'$ with y in **In**" is "F' can be obtained from F by adding a part inaccessible from F, namely that corresponding to the vertices that are not in the image of y". Of course, the relation $\rightarrow_{\mathbf{In}}$ is not symmetric, the meaning of the converse relation $F' \xleftarrow[y]{} F$ being "F can be obtained from F' by deleting the part corresponding to the complement of the image of y; this part is not accessible from the remained one".

Now we turn back to the general setting. For the congruence relation $=_{\mathbf{In}}$ generated by $\rightarrow_{\mathbf{In}}$, the following equivalent characterizations can be given:

- (i) $=_{\mathbf{In}} = {\rightarrow_{\mathbf{In}}}^{-1} \cdot \rightarrow_{\mathbf{In}}$;
- (ii) $=_{\mathbf{In}}$ is the congruence relation generated by the identifications ($\leftrightarrow X$) in §4.1 and the identifications

$$((1_m + x)f) \uparrow^{i(x)} = ((1_m + x + y)g) \uparrow^{i(x+y)} \quad \text{when } f(1_n + 1_{i(x)} + 0_{i(y)}) =$$
$$= (1_m + 1_{o(x)} + 0_{o(y)})g ,$$

[*] In automata and system theory the analogous property is called "reachability" (cf. Arbib-Manes).

where x and y are finite sums of variables;

(iii) $=_{In}$ is the congruence relation $\sim$ satisfying

(P_{In}) "$F(1_n + y) \sim (1_m + y)G \Rightarrow F\uparrow^p \sim G\uparrow^q$ where

$F : m + p \to n + p$, $G : m + q \to n + q$ and $y \in In(p,q)$"

generated by the identifications ($\leftrightarrow$ X) and the identifications

(OX) $O_m \cdot x = O_n$, where $x \in X(m,n)$ (see Figure 10.b).

Comments. By (i) two flowchart schemes are $=_{In}$ - equivalent iff they can be transformed into the same scheme by deleting inaccessible parts. In (ii), by using separate simulations via bijective functions, we can suppose that the injective function y has the particular form $1_r + O_s$, and, in this case, the meaning of the formula of simulation is much clearer. Much more interesting is the characterization (iii), since it reduces the generators to $(\leftrightarrow X) + (OX)$ by restricting the class of the congruence relations, used to generate $=_{In}$, to those satisfying (P_{In}).

5.2. The mathematical concept of accessible flowchart scheme. The above facts show that, in the case $T \subseteq Rel$, every equivalence class with respect to $=_{In}$, contains an accessible flowchart scheme, unique up to an isomorphism. Consequently we can identify an accessible flowchart scheme to its $=_{In}$ - equivalence class. Generalizing we say

the elements in a $Fl_{X,T}/=_{In}$ are accessible flowchart schemes.

5.3, 5.4. We do not insist on the algebraic rules satisfied by accessible flowchart schemes. We only mention that the corresponding algebraic structure, called inflow, is a biflow, contains injections (in order to generate injections we use the distinguished morphisms $O_n : 0 \to n$), and satisfies:

(11) $O_m \cdot f = O_n$, for $f : m \to n$;

(12) $f(1_n + y) = (1_m + y)g \Rightarrow f\uparrow^p = g\uparrow^q$,

where $f : m + p \to n + p$, $g : m + q \to n + q$ and y is an injection: $p \to q$.

5.5. In flow-calculus. The calculus with flownomials associated to accessible flowchart schemes, called in-flow-calculus, is obtained by adding to the rules that define the bi-flow-calculus in §4.5 the rule which consists in the identification of $=_{In}$ - equivalent expressions.

For the algebraic version, we add the rules (I1-2) above to the rules (R1, B1-4, B6, B10-13) in §4.5 that define algebraically the bi-flow-calculus.

Example. In in-flow-calculus the following identity holds;

$$((1_3 + y + x + y)(1,3,2,3,2,5,4,4)_5)\uparrow^2 = (1_3 + y)(1,3,2,3)_3 .$$

(a) Proof. Using normal forms: The normal form of left-hand side expression (G in Figure 9) is $NF_1 = [(1_4 + y + x + y)(1,3,2,4,3,2,6,5,5)_6]\uparrow^3$ and of $(1_3 + y)(1,3,2,3)_3$ is $NF_2 = [(1_4 + y)(1,3,2,4,3)]_4\uparrow^3$. We shall prove that $NF_2 \to_y NF_1$ for the injection $y = 1_1 + O_2 = (1)_3$. Note that y preserves the statements with respect to the sequences (y) and (y,x,y), hence condition (i) in definition §3.1 holds. The extension of y to input is $(1)_3$ and to outputs is $(1)_5$. Since $(1,3,2,4,3)_4(1_3 + (1)_3) = (1,3,2,4,3)_6 = (1_4 + (1)_5)\cdot(1,3,2,4,3,2,6,5,5)_6$ the condition (ii) in definition §3.1 holds, too.

(b) Proof. Using algebraic rules (marking the application of the new rules (I1-2) only): Note that $O_2(x + y) = (O_1 + O_1)(x + y) = O_1x + O_1y =$ (by I1) $O_3 + O_1 = O_4$, hence

$$(1_4 + \boxed{O_2})(1_3 + y + x + y)(1,3,2,3,2,5,4,4)_5 = (1_3 + y + O_4)(1,3,2,3,2,5,4,4)_5 =$$
$$= (1_3 + y)(1_4 + O_4)(1,3,2,3,2,5,4,4)_5 = (1_3 + y)(1,3,2,3)_3(1_3 + \boxed{O_2}).$$

Using (I2) we obtain

$$[(1_3 + y + x + y)(1,3,2,3,2,5,4,4)_5]\uparrow^2 = [(1_3 + y)(1,3,2,3)_3]\uparrow^0$$

hence the conclusion follows.

6. REDUCTION £)

We repeat: a flowchart scheme is a notation of a sequential computation process.

£) In automata and system theory the similar property is called "observability" (cf. Arbib-Manes, J. Pure Appl. Algebra 6(1975) 317-[illegible]).

Hence the result of the computation depends on the sequences of statements to be executed only. The (step-by-step) behaviour of a vertex in a flowchart scheme is the set of all finite and infinite sequences of statements that can be executed beginning with the given vertex. In a flowchart scheme we can identify the vertices that have the same behaviour and obtain a flowchart scheme that denotes the same computation process. A flowchart scheme will be called reduced if it has no different vertices having the same behaviour. Here we regard as equivalent two flowchart schemes that can be reduced to the same scheme by identifying vertices with the same behaviour. In a formal approach reduction is captured by simulation via surjective functions.

6.1. The simulation via surjective functions; the resulted congruence. Suppose that the support theory T "contains" surjective, finite functions. In the case $T \subseteq \mathbf{Rel}$ this means $T \supseteq \mathbf{Sur}$.

The definition of simulation via surjective functions is obtained from the general definition, given in § 3.1, by using for y morphisms in **Sur**.

In the particular case when T is a subtheory of **Rel** the meaning of the simulation "$F \xrightarrow[y]{} F'$ with y in **Sur**" is "F' can be obtained from F by identifying vertices which have the same label and whose output connections are equal after identification". Of course, the relation $\rightarrow_{\mathbf{Sur}}$ is not symmetric, the meaning of the converse relation $F' \xleftarrow[y]{} F$ being "F can be obtained from F' by (partially) unfolding same vertices".

Now we turn back to the general setting. For the congruence relation $=_{\mathbf{Sur}}$ generated by $\rightarrow_{\mathbf{Sur}}$, the following equivalent characterizations can be given:

(i) $=_{\mathbf{Sur}} = \rightarrow_{\mathbf{Sur}} \cdot {}_{\mathbf{Sur}}\leftarrow$;

(ii) $=_{\mathbf{Sur}}$ is the congruence relation $\sim$ satisfying

$(P_{\mathbf{Sur}})$ "$F(1_n + y) \sim (1_m + y)G \Rightarrow F\uparrow^p \sim G\uparrow^q$,

where $F : m + p \to n + p$, $G : m + q \to n + q$ and $y \in \mathbf{Sur}(p,q)$"

generated by the identifications ($\leftrightarrow X$) and the identifications

(VX) $mVm \cdot x = (x + x) \cdot nVn$, where $x \in X(m,n)$ (see Figure 10 c).

Comments. By (i) two flowchart schemes are $=_{Sur}$ - equivalent iff they can be reduced to the same scheme by identifying certain vertices. The characterization (ii) gives very simple generators for $=_{Sur}$ by restricting the class of congruence relations used to generate $=_{Sur}$.

6.2. The mathematical concept of reduced flowchart scheme. The above facts show in the case $\mathcal{T} \subseteq \mathbf{Rel}$ every equivalence class, with respect to $=_{Sur}$, contains a reduced flowchart scheme, unique up to an isomorphism. Consequently, we can identify a reduced flowchart scheme to its $=_{Sur}$ - equivalence class. Generalizing we say:

> the elements in a $Fl_{X,T}/=_{Sur}$ are reduced flowchart schemes.

6.3, 6.4. We do not insist on the algebraic rules satisfied by reduced flowchart schemes. We only mention that the corresponding algebraic structure, called surflow, is a biflow, contains surjections (in order to generate surjections we use the distinghuished morphisms $mVm : m + m \to m$), and satisfies:

(S1) $mVm \cdot f = (f + f) \cdot nVn$, for $f : m \to n$;

(S2) $f(1_n + y) = (1_m + y)g \Rightarrow f \uparrow^p = g \uparrow^q$,
where $f : m + p \to n + p$, $g : m + q \to n + q$ and y is a surjection: $p \to q$.

every surjection is a functional morphism.

6.5. Sur-flow-calculus. The calculus with flownomials associated to reduced flowchart schemes, called sur-flow-calculus, is obtained by adding to the bi-flow-calculus the rule which consists in the identification of $=_{Sur}$ - equivalent expressions.

For the algebraic version, we add the rules (S1-2) above to the rules that algebraically define the bi-flow-calculus in §4.5.

Examples. In sur-flow-calculus the following identities hold:

(a) $(1V1 \cdot x(1_2 + x)) \uparrow (\bot, 1, \bot, 1)_1 = (1V1 \cdot x) \uparrow (\bot, 1)_1$;

(b) $(1V1 \cdot x(1_1 + y + x)) \uparrow (1_3 + y)(1,3,2,3)_3 = (1V1 \cdot x(1_2 + x)) \uparrow (1,3,2,3)_3 (1_2 + y)$.

Proof of (a) using normal forms: The normal form of the left-hand side expression is $NF_1 = [(1_1 + x + x)(2,\perp,1,3,\perp,1,2)_3]\uparrow^2$ and that of the right-hand side expression is $NF_2 = [(1_1 + x)(2,\perp,1,2)_2]\uparrow^1$. We shall prove that $NF_1 \rightarrow_y NF_2$ for the surjection $y = (1,1)_1$. Note that y preserves the statements with respect to the sequences (x,x) and (x), hence the condition (i) in definition § 3.1 holds. The extension of y to inputs is $(1,1)_1$ and to outputs is $(1,2,3,1,2,3)_3$. Since $(2,\perp,1,3,\perp,1,2)_3(1_1 + (1,1)_1) =$ $= (2,\perp,1,2,\perp,1,2)_2 = (1_1 + (1,2,3,1,2,3)_3)(2,\perp,1,2)_2$ the condition (ii) in definition § 3.1 holds, too.

Proof of (b) using algebraic rules. By the example in § 4.5 the left-hand side expression is equal to

$$(1V1 \cdot x(1_2 + x))\uparrow(1,3,2,4)_4(1_2 + (y + y) \cdot 1V1)\uparrow$$

$$= (1V1 \cdot x(1_2 + x))\uparrow(1,3,2,4)_4(1_2 + 1V1 \cdot y) \qquad \text{by (S1)}$$

$$= (1V1 \cdot x(1_2 + x))\uparrow(1,3,2,3)_3(1_2 + y).$$

7. THE INPUT BEHAVIOUR (COMPLETE MINIMIZATION)

A flowchart scheme denotes a sequential computation process. For an input of the scheme let us consider the set of finite and infinite sequences of statements that can be executed beginning with this input. (This set can be identified with the tree obtained by completly unfolding the scheme beginning with the given input.) By (step-by-step) input behaviour of a flowchart scheme we mean the tuple of the sets obtained as above for each input. It is natural to regard as equivalent two flowchart schemes that have the same input behaviour. In the class of the flowchart schemes that have a given input behaviour there is a minimal one, unique up to an isomorphism. This minimization preserves the completeness of the scheme, namely the minimal flowchart scheme in $Fl_{X,Pfn}$ of a scheme over Fn is over Fn, too - hence the name. In a formal approach the (step-by-step) input behaviour is captured by simulation via functions.

7.1. The simulation via functions; the resulted congruence. Suppose that the support theory T "contains" functions. In the case $T \subseteq \mathbf{Rel}$ this means $T \supseteq \mathbf{Fn}$.

The definition of simulation via functions is obtained from the general definition, given in §3.1, by using for γ morphisms in **Fn**.

In the general case $\rightarrow_{\mathbf{Fn}} \subseteq \rightarrow_{\mathbf{Sur}} \cdot \rightarrow_{\mathbf{In}}$ hence in the case $T \subseteq \mathbf{Rel}$ the meaning of the simulation "$F \rightarrow_{\gamma} F'$ with γ in **Fn**" is "F' can be obtained from F in two steps: first by identifying vertices with common labels and coherent continuations, and second by adding inaccessible vertices". The meaning of the equivalence relation $=_{\mathbf{Fn}}$, generated by $\rightarrow_{\mathbf{Fn}}$, is "$F =_{\mathbf{Fn}} F'$" iff "F and F' have the same (step-by-step) input behaviour (or equivalently, by completly unfolding F and F' we get same tuple of trees)" iff "by identifying vertices and deleting inaccessible ones F and F' can be transformed into the same minimal flowchart (with respect to the input behaviour)".

For the congruence relation $=_{\mathbf{Fn}}$, generated by $\rightarrow_{\mathbf{Fn}}$, the following equivalent characterization can be given:

(i) $=_{\mathbf{Fn}} = \rightarrow_{\mathbf{Sur}} \cdot \mathbf{In}^{\leftarrow} \cdot \rightarrow_{\mathbf{In}} \cdot \mathbf{Sur}^{\leftarrow}$;

(ii) $=_{\mathbf{Fn}}$ is the congruence relation ~ satisfying

$(P_{\mathbf{Fn}})$ "$F(1_n + y) \sim (1_m + y)G \Rightarrow F\uparrow^p \sim G\uparrow^q$,

where $F : m + p \rightarrow n + p$, $G : m + q \rightarrow n + q$ and $y \in \mathbf{Fn}(p,q)$"

generated by the identifications ($\leftrightarrow$ X) in §4.1, (OX) in §5.1 and (VX) in §6.1.

Comments. By (i) two flowchart schemes are $=_{\mathbf{Fn}}$ - equivalent iff by identifying vertices and deleting inaccessible ones they can be transformed into the same scheme. Again in (ii) we get very simple generators (now for $=_{\mathbf{Fn}}$) restricting the class of congruence relations used for generation.

7.2. Computation processes (or minimal flowchart schemes with respect to the input behaviour). In the case $T \subseteq \mathbf{Rel}$ every $=_{\mathbf{Fn}}$ - equivalence class has a minimal flowchart, unique up to an isomorphism. Since two schemes are $=_{\mathbf{Fn}}$ - equivalent iff

they have the same computation sequences, we can identify such a class to a computation process that cousist in finite and infinite sequences of statemetns. Generalizing we say

> The elements in a $Fl_{X,T}/= \mathbf{Fn}$ are:
>
> - minimal flowchart schemes with respect to the input behaviour;
>
> - computation processes.

7.3. The algebra of minimal flowchart scheme (with respect to the input behaviour). We had selected some identities satisfied by such minimal schemes (namely, the identities listed in Table 2 and illustrated in Figure 12), in order to define an algebraic structure, called funflow. The main point is that the set of identities (B1-15) + (F1 - 6), suffices to prove that flownomial expressions over **Pfn**, which represent the same computation process, are equal.

Rigorously, a funflux (formerly a strong iteration algebraic theory) is a biflow B, with some distinguished morphisms $O_m \in B(0,m)$ and $mVm \in B(m+m,m)$, and satisfying the algebraic rules listed in Table 2 (it should be emphasized that (F6) is not an equation, but an implication). The axioms (B1-10) + (F1 - 5) give a presentation of algebraic theories - in the sense of Lawvere - in terms of sum and composition, hence finite functions are embedded in each funflow.

(F1) $O_0 = 1_0$	(F5) $m \leftrightarrow n = (O_n + 1_m + 1_n + O_m) \cdot (n+m)V(n+m)$
(F2) $O_m \cdot f = O_n$	(F6) $f(1_n + y) = (1_m + y)g \Rightarrow f\uparrow^p = g\uparrow^q$
(F3) $(nVn + 1_n) \cdot nVn = (1_n + nVn) \cdot nVn$	for $f : m+p \to n+p$, $g : m+q \to n+q$,
(F4) $mVm \cdot f = (f + f) \cdot mVn$	and $y \in Fn(p,q)$

Table 2. The axioms in Table 1 together with these ones define a funflow

Semantic models: $\mathbf{Rel}_S$ and all of its subtheories, which contain the embedding of

Fn in $\mathbf{Rel}_S$, are funflows. Particularly, **Pfn** and **Rel** are funflows, **Pfn** being an initial funflow.

Syntactic models: If T is a funflow, then $Fl_{X,T}/=_{\mathbf{Fn}}$ is a funflow.

7.4. The universal property. Let (E^f_X, E^f_T) be the embedding of (X,T) into $Fl_{X,T}/=_{\mathbf{Fn}}$ obtained by using the embedding (E_X, E_T) of (X,T) into $Fl_{X,T}$, defined in § 1.3, and the canonical projection from $Fl_{X,T}$ to $Fl_{X,T}/=_{\mathbf{Fn}}$. The universal property satisfied by $Fl_{X,T}/=_{\mathbf{Fn}}$ is

"for every funflow F and every interpretation (I_X, I_T) of (X,T) in F there exists a unique morphism of funflows $I^{ff} : Fl_{X,T}/=_{\mathbf{Fn}} \to F$ such that $E^f_X \cdot I^{ff} = I_X$ and $E^f_T \cdot I^{ff} = I_T$".

The axiom (F6) ensures that the interpretation $(I_X, I_T)^f$, defined in § 4.4, identifies $=_{\mathbf{Fn}}$ - equivalent flownomial expressions. The morphism I^{ff} above is that induced by $(I_X, I_T)^f$ in the quotient structure $Fl_{X,T}/=_{\mathbf{Fn}}$.

7.5. Fn-flow-calculus. The calculus with flownomials associated to computation processes (or to minimal flowchart schemes, with respect to the input behaviour) called fn-flow-calculus, is obtained by adding to the bi-flow-calculus the rule which consists in the identification of $=_{\mathbf{Fn}}$ - equivalent expressions.

For the algebraic version, we add the rules (F2, F4, F6) in Table 2 to the rules that algebraically define the bi-flow-calculus.

Examples. In fn-flow-calculus the following identity holds

$$F \cdot G \cdot H = [1V1 \cdot x(1_2 + x)]\uparrow (1,3,2,3)_3[((1_1 + 1V1 \cdot y)z)\uparrow + y],$$

where $F := [1V1 \cdot x(1_1 + y + x)]\uparrow$, $G := [(1_3 + y + x + y)(1,3,2,3,2,5,4,4)_5]\uparrow^2$ and $H := ((1_1 + 1V1 \cdot y)z)\uparrow + 1_1$. Moreover, the left-hand side expression is $=_{\mathbf{Fn}}$-minimal.

Indeed, by example §5.5 $G = (1_3 + y)(1,3,2,3)_3 =: G'$ and by example §6.5.b $F \cdot G' = [IV1 \cdot x(1_2 + x)] \uparrow (1,3,2,3)_3(1_2 + y)$. Hence the identity holds. The left-hand side expression is $=_{Fn}$ - minimal since the associated flowchart scheme is reduced and accessible.

8. COACCESSIBILITY

Sometimes in a computation process we are interested in successful computation sequences only (i.e., computation paths that finesh normally by reaching an output). In that case, in the execution process only the vertices that belong to paths going to outputs matter; these vertices form the coaccessible part of the scheme. Here we regard as equivalent two flowchart scheme that have the same coaccessible part. In a formal approach coaccessibility is captured by simulation via relations whose converses represent injective functions.

The study of coaccessibility can be reduced to the study of accessibility, made in § 5, by using a principle of duality: The dual flowchart scheme associated to a scheme F with m inputs and n outputs, is the scheme F^o, with n inputs and m outputs, obtained by reversing arrows of F (in the abstract case this method consists in taking the dual composition in the dual category). In this way the coaccessible part of a scheme F is the accessible part of the dual scheme F^o.

For this reason we omit any details here.

9. THE INPUT-OUTPUT BEHAVIOUR (DETERMINISTIC MINIMIZATION)

The input-output behaviour of a scheme is the restriction of the (step-by-step) input behaviour to the successful (terminal) paths. Here we regard as equivlaent two flowchart schemes that have the same input-output behaviour. In the class of the schemes that have ~~a given~~ the same input-output behaviour there is a minimal one, unique up to an isomorphism. The minimization with respect to the input-output behaviour does not preserve the completeness of a scheme, i.e., the minimal scheme associated to a scheme over **Fn** may be over **Pfn** $\setminus$ **Fn**. However, this minimization preserves the

determinism of a scheme, i.e., the minimal scheme in $Fl_{X,Rel}$ of a scheme over **Pfn** is over **Pfn**, too - hence the name. Formally the input-output behaviour is captured by simulation via partially defined functions.

9.1. The simulation via partial functions; the resulted congruence. Suppose that the support theory T "contains" partial functions. In the case $T \subseteq$ **Rel** this means $T \supseteq$ **Pfn**.

The definition of simulation via partial functions is obtained from the general definition, given in §3.1, by using for γ morphisms in **Pfn**.

In the general case $\rightarrow_{\mathbf{Pfn}} \subseteq \rightarrow_{\mathbf{In}}^{-1} \cdot \rightarrow_{\mathbf{Sur}} \cdot \rightarrow_{\mathbf{In}}$, hence in the case $T \subseteq$ **Rel** the meaning of the simulation "$F \rightarrow_{\gamma} F'$ with γ in **Pfn**" is "F' can be obtained from F in three steps: first by deleting noncoaccessible vertices, second by identifying vertices with common labels and coherent continuations, and finally by adding inaccessible vertices".

The meaning of the equivalence relation $=_{\mathbf{Pfn}}$, generated by $\rightarrow_{\mathbf{Pfn}}$, is "$F =_{\mathbf{Pfn}} F'$" iff "F and F' have the same input-output behaviour" iff "by deleting noncoaccessible vertices, identifying vertices with common labels and coherent continuations, and deleting inaccessible vertices F and F' can be transformed into the same minimal scheme (with respect to the input-output behaviour)".

For the congruence relation $=_{\mathbf{Pfn}}$, gnerated by $\rightarrow_{\mathbf{Pfn}}$, the following equivalent characterization can be given:

(i) $=_{\mathbf{Pfn}} = \rightarrow_{\mathbf{In}}^{-1} \cdot \rightarrow_{\mathbf{Sur}} \cdot \rightarrow_{\mathbf{In}} \cdot \leftarrow_{\mathbf{In}} \cdot \rightarrow_{\mathbf{Sur}} \cdot \leftarrow_{\mathbf{In}}^{-1}$;

(ii) $=_{\mathbf{Pfn}}$ is the congruence relation $\sim$ satisfying

$(P_{\mathbf{Pfn}})$ "$F(1_n + y) \sim (1_m + y)G \Rightarrow F \uparrow^p \sim G \uparrow^q$,
where $F : m + p \rightarrow n + p$, $G : m + q \rightarrow n + q$ and $y \in \mathbf{Pfn}(p,q)$"

generated by the identifications ($\leftrightarrow X$) in §4.1, (OX) in §5.1, (VX) in §6.1 and

$(\perp X)$ $x \cdot \perp_n = \perp_m$, where $x \in X(m,n)$ (see figure 10 d).

Comments. By (i) two flowchart schemes are $=_{\mathbf{Pfn}}$ - equivalent iff by deleting

noncoaccessible vertices, identifying vertices and deleting inaccessible ones they can be transformed into the same scheme. In (ii) we get very simple generators for $=_{\mathbf{Pfn}}$, by restricting the clas of congruence relations used to generate $=_{\mathbf{Pfn}}$.

9.2. Successful computation processes (or minimal flowchart schemes with respect to the input-output behaviour). In the case $T \leq \mathbf{Rel}$ every $=_{\mathbf{Pfn}}$ - equivalence class has a minimal scheme, unique up to an isomorphism. Since two schemes are $=_{\mathbf{Pfn}}$ - equivalent iff they have the same successful computation processes, we can identify such a class to a successful computation process that consists in finite terminal sequences of statements. Generalizing we say:

> The elements in a $\mathrm{Fl}_{X,T}/=_{\mathbf{Pfn}}$ are:
>
> - minimal flowchart schemes with respect to the input-outpur behaviour;
>
> - successful computation processes.

9.3, 9.4. We only mention that the algebraic structure corresponding to successful computation processes, called parfunflow, is a funflow, contains partial functions (in order to generate partial functions we use the distinguished morphisms $\perp_m : m \to 0$), and satisfies:

(P1) $f \cdot \perp_n = \perp_m$, for $f : m \to n$;

(P2) $f(1_n + y) = (1_m + y)g \Rightarrow f\uparrow^p = g\uparrow^q$,

where $f : m + p \to n + p$, $g : m + q \to n + q$ and $y \in \mathbf{Pfn}(p,q)$.

9.5. Pfn-flow-calculus. The calculus with flownomials associated to successful computation processes (or, to minimal flowchart schemes, with respect to the input-output behaviour), called pfn-flow-calculus, is obtained by adding to the bi-flow-calculus the rule which consists in the identification of $=_{\mathbf{Pfn}}$ - equivalent expressions.

For the algebraic version, we add the rules (P1-2) above to the rules that

algebraically defined pfn-flow-calculus in §7.5.

Example. In pfn-flow-calculus the following identity holds

$$F \cdot G \cdot H = (1V1 \cdot x)\uparrow(\perp_1 + y),$$

where F,G,H are those defined in example §7.5. Moreover, the left-hand side expression is $=_{\mathbf{Pfn}}$ - minimal.

Indeed $1_0 = \perp_0$, hence $[(1_1 + 1V1 \cdot y)z]\uparrow = [(1_1 + 1V1 \cdot y)z]\uparrow \cdot 1_0 = [(1_1 + 1V1 \cdot y)z]\uparrow \cdot \perp_0 = (\text{by P1})\ \perp_2$. Consequently, using the example §7.5 we obtain

$$F \cdot G \cdot H = [1V1 \cdot x(1_2 + x)]\uparrow(1,3,2,3)(\perp_2 + y) = (1V1 \cdot x(1_2 + x))\uparrow(\perp,1,\perp,1)_1 y\ .$$

Finally, using the example §6.5.a the desired identity follows easily. The left-hand side expression is $=_{\mathbf{Pfn}}$-minimal since the associated flowchart scheme is coaccessible, reduced and accessible.

10. COREDUCTION

In the case we are interested in the study of input-output behaviours we can successfully use the duality defined in §8. The input behaviour is not preserved by duality, while the input-output behaviour is preserved. More exactly, the input-output behaviour of a dual scheme contains the same computation sequences as the given scheme, but having the statements concatenated in the reverse order. Let us call (step-by-step) cobehaviour of a vertex in a scheme, the (step-by-step) behaviour of the corresponding vertex in the dual scheme, defined as in §6. In a scheme we can identify vertices that have the same cobehaviour, without changing the input-output behaviour of the scheme. A flowchart scheme will be called coreduced if it has no different vertices having the same cobehaviour. In a formal approach coreduction is captured by simulation via relations whose converses represent surjective functions.

This coreduction connot be used properly in the context of deterministic flowchart schemes. The reason is the following. By reduction we identify vertices provided after identification they have the same output arrows and bring together the

corresponding input arrows. By coreduction we have to identify vertices provided after identification they have the same input arrows and bring together the corresponding output arrows; hence a nondeterministic choice between different output arrows of continuation can appear.

More details can be obtained from §6 by duality.

11. NONDETERMINISTIC FLOWCHART SCHEMES

The feature of flowchart scheme we take now into account is "nondeterministic choice", i.e., the possibility in a point of a scheme (input, or continuation after a statement) to have more arrows of continuation for the flow of control, and the execution process chooses one variant in a random way. Consequently, in the context of usual flowchart schemes, represented as in §1.1, the basic support theory for this nedeterministic case is **Rel**, while in the deterministic case the basic support theory was **Pfn**. In the presence of the nondeterministic choice we are interested in considering the input-output behaviour, rather than the input behaviour. For modeling the input-output behaviour, in this nondeterininistic case we can try to apply simulation via relations. This syntactic transformation of flowchart schemes is again useful: it is correct, in the sense it preserves the input-output behaviour, but at the present time we do not know whether it is complete, i.e. we do not know whether two usual nondeterministic schemes, having the same input-output behaviour, can be connected by a chain of simulations.

11.1. The simulation via relations; the resulted congruence. Suppose that the support theory T "contains" finite relations. In the case of usual flowchart schemes this means T = **Rel**. The definition of the simulation via relations was given in §3.1.

In the general case $\rightarrow_{\mathbf{Rel}} \subseteq \rightarrow_{\mathbf{In}^{-1}} \cdot \rightarrow_{\mathbf{Sur}^{-1}} \cdot \rightarrow_{\mathbf{Sur}} \cdot \rightarrow_{\mathbf{In}}$, hence in the case T = **Rel** the meaning of the simulation "$F \rightarrow F'$" is "F' can be obtained from F in four steps: first by deleting noncoaccessible vertices, second by multiplying vertices

keepping fix the inputs and sharing the outputs, then by identifying vertices that give the same outputs and bring together the corresponding inputs, and finally by adding inaccessible vertices". The meaning of the equivalence relation $=_{\mathbf{Rel}}$, generated by $\rightarrow_{\mathbf{Rel}}$, is still unclear. We conjecture that "$F =_{\mathbf{Rel}} F'$", iff "F and F' have the same input-output behaviour".

For the congruence relation $=_{\mathbf{Rel}}$, generated by $\rightarrow_{\mathbf{Rel}}$, the following equivalent characterization can be given:

(i) $=_{\mathbf{Rel}} = \overleftarrow{\mathbf{In}} \cdot \overleftarrow{\mathbf{Sur}} \cdot \overrightarrow{\mathbf{In}}^{-1} \cdot \overrightarrow{\mathbf{Sur}}^{-1} \cdot \overrightarrow{\mathbf{Sur}} \cdot \overrightarrow{\mathbf{In}} \cdot \overleftarrow{\mathbf{Sur}}^{-1} \cdot \overleftarrow{\mathbf{In}}^{-1}$;

(ii) $=_{\mathbf{Rel}}$ is the congruence relation $\sim$ satisfying

$(P_{\mathbf{Rel}})$ "$F(1_n + y) \sim (1_m + y)G \Rightarrow F\uparrow^p \sim G\uparrow^q$,

where $F : m + p \rightarrow n + p$, $G : m + q \rightarrow n + q$ and $y \in \mathbf{Rel}(p,q)$"

generated by the identifications $(\Leftrightarrow X)$ in §4.1, (OX) in §5.1, (VX) in §6.1, $(\perp X)$ in §9.1 and

$(\wedge X)$ $x \cdot n \wedge n = m \wedge m \cdot (x + x)$, where $x \in X(m,n)$ (see Figure 10 e).

As we do not know the semantic meaning of $=_{\mathbf{Rel}}$ we do not insist on this syntactic study which has been done as a natural extension of the above ones.

HISTORICAL COMMENTS

It is well known that the operations of "structured programming", i.e., composition, **if-then-else** and **while-do** are not enough for representing all flowchart scheme behaviours, essentially due to their one-input/one-exit feature. However they suffice, provided additional memory is permitted.

The basic Elgot's idea in [3] is to use many-input/many-exit flowchart schemes, having **composition, tupling** and **scalar iteration** as basic operations. These operations suffice for representing all flowchart scheme behaviours. More precisely, every flowchart scheme is "strongly equivalent" (i.e., is equivalent with respect to the input

behaviour) to a scheme built up from atomic schemes and trivial ones by means of these operations. However, for representing all flowchart schemes (pictures) in this setting one needs a vectorial iteration, which is not obtained by a repeated application of the scalar iteration.

The feedback operation was introduced in [12]. It is a "scalar" operation and it seems that this operation is more adequate to study (cyclic) flowchart schemes than scalar iteration. One reason is the following: All flowchart schemes (pictures) can be built up from atomic schemes and trivial ones by means of **composition**, (separated) **sum** and (scalar) **feedback**.

(1) The representation of flowchart schemes by pairs (or by triples, provided that the connection morphism is splitted into its "input" part and its "transfer" part) is due to Elgot; see [3,14,4,2]. At that stage the schemes were over **Fn** [3], **Sur** [4], or **Pfn** [2] (although it was not thought of connections as being morphisms in a "theory"), and the operations on flowchart schemes were verbally defined. In [8] the connections were thought of as being morphisms in an "algebraic theory with iterate". Particularly, this condition implies that in the case of usual flowchart schemes one has to replace **Fn** by its closure with respect to iteration, namely **Pfn**. The operations on flowchart schemes were defined formally by extending those of the theory of connections. The representation of flowchart schemes by flownomial expressions in normal form was introduced in [12, 13]. The extension to arbitrary flownomial expressions was given in [7].

(2) The results of Section 1 and 4 are new. The details for Section 1 were given in [6]. Without axiomatizing finite bijections the result of Section 4 was sketched in [13]. The results in § 1 and § 4 are stronger and much more natural than those in [8,5,2]. Actually a theory with iterate, as introduced in [8], may be defined as a biflow over an algebraic theory; see [6]. The main result in [8] characterizes the representations in $Fl_{X,T}$, where T is a theory with iterate, as being the "T-modul with iterate" freely generated by X. The extension to schemes in $Fl_{X,T/=_{Bi}}$ was given in [5]. The main obstruction in obtaining a natural result regarding the algebraic characterization of

flowchart schemes in [8,5] was the using of an algebraic theory as support theory. Indeed, the flowchart schemes do not have a structure of an algebraic theory but only of a strict monoidal category. This comment applies also to [2].

(3) The results of Section 7 are the translation in terms of feedback of the results in [10]. In the translation of these results the use of the new set of operations (composition-sum-feedback) allows to separate the study of accessibility, given in Section 5, from the study of reduction, given in Section 6. The results of Sections 5 and 6 are new and cannot be properly done using algebraic theories and iteration.

(4) In section 9 we have given some details for the extension of the calculus for deterministic flowchart schemes announced in [10, Section 7.a]. The results appear here for the first time but a weaker variant directly follows from the results in the nondeterministic case in [11]. The paper [11] covers the result of Section 11, too.

REFERENCES

1. S.L. Bloom, "Calvin C. Elgot, Selected Papers", Springer-Verlag, 1982.
2. S.L. Bloom and Z. Esik, Axiomatizing schemes and their behaviours, J. Comput. System Sci. 31 (1985), 375-393.
3. C.C. Elgot, Structured programming with and without GOTO statements, IEEE Trans. Software Eng. SE-2, 41-53.
4. C.C. Elgot and J.C. Shepherdson, An Equational Axiomatization of the Algebra of Reducible Flowchart Schemes, IBM Research Report, RC-8221 (April, 1980).
5. V.E. Căzănescu and Ş. Grama, On the Definition of M-Flowcharts, INCREST Preprint Series in Mathematics, No.56/1984.
6. V.E. Căzănescu and Gh. Ştefănescu, A Formal Representation of Flowchart Schemes, INCREST Preprint Series in Mathematics, No. 22/1987.
7. V.E. Căzănescu and Gh. Ştefănescu, A calculus for flowchart schemes, Abstracts 8th International Congress of Logic, Methodology and Philosophy of Science LMPS'87, Vol.1, pag. 124-127, Nauka 1987.

8. V.E. Căzănescu and C. Ungureanu, "Again on Advice on Structuring Compilers and Proving them Correct", INCREST Preprint Series in Mathematics, No. 75/1982.

9. S. MacLane, Categories for the Working Mathematician, Springer-Verlag, 1971.

10. Gh. Ştefănescu, On flowchart theories. Part I. The determinisite case, J. Comput. System Sci. 35 (1987), 163-191.

11. Gh. Ştefănescu, On flowchart theories. Part 2. The nondeterminisite case, Theoret. Comput. Sci. 52 (1987), 307-340.

12. Gh. Ştefănescu, An algebraic theory of flowchart schemes (extended abstract), Proceedings CAAP'86, LNCS 214, Springer-Verlag 1986, pp. 60-73.

13. Gh. Ştefănescu, Feedback Theories (A Calculus for Isomorphism Classes of Flowchart Schemes), INCREST Preprint Series in Mathematics, No. 24/1986.

14. ADJ (J.W. Thatcher, E.G. Wagner and J.B. Wright), Notes on algebraic fundamentals for theoretical computer science, in "Foundations of computer science III, Part 2: Language, logic, semantics" (J.W. de Bakker and J. van Leeuwen, Eds.), pp.83-164, Mathematical Centre Tracts 109, Amsterdam, 1979.

In this appendix we give an abstract of our work regarding the axiomatizing of the various classes of finite relations used before: bijections, injections etc. It has been written for the East European Category Seminar (EWCS'88, Prodelia, Bulgaria, February 28 - March 5, 1988), hence it is written in a categorical language and the notation are different from those used in the rest of the paper. ($\underline{\underline{\mathbf{Rel}}}_S$, as used here, is not $\underline{\underline{\mathbf{Rel}}}_S$ that was introduced on page 10!) Another title for the same work is "Finite relations as initial abstract data types" - this title is more adequate for the theoretical computer scientists.

ON SOME SYMMETRIC STRICT MONOIDAL CATEGORIES

Virgil Căzănescu and Gheorghe Ştefănescu

We have found that each of some subcategories of the category of finite S-sorted lations $\mathbf{Rel}_S$ is freely generated by the set S of objects in a category of symmetric strict onoidal categories (ssmc-ies, for short) [2]* endowed with an adequate additional structure.

$\mathbf{Rel}_S$ as a category: An object of $\mathbf{Rel}_S$ is an element in the free monoid $(S^*, \square, e)$, i.e. $\square$ notes juxtaposition and e the empty string. Notation: A string $a \in S^*$ is denoted as follows $= a_1 \square a_2 \square \ldots \square a_{|a|}$, where $a_i \in S$; $[n] = \{1,2,\ldots,n\}$. A morphism $f \in \mathbf{Rel}_S(a,b)$ is a relation $\subseteq [|a|] \times [|b|]$ such that $(i,j) \in f$ implies $a_i = b_j$. The composition and the identity morphisms e the usual ones.

$\mathbf{Rel}_S$ as a ssmc $(\mathbf{Rel}_S, \square, e, \gamma)$: For $f \in \mathbf{Rel}_S(a,b)$ and $g \in \mathbf{Rel}_S(c,d)$ we define

$$f \square g \in \mathbf{Rel}_S(a \square c, b \square d) \text{ as } f \cup \{(|a| + i, |b| + j) \mid (i,j) \in g\};$$

$$\gamma_{a,b} := \{(i, |b| + i) \mid i \in [|a|]\} \cup \{(|a| + j, j) \mid j \in [|b|]\}.$$

The subcategories $P\mathbf{Rel}_S$: To define them let us consider the relations:

$$a \nabla a = \{(i,i) \mid i \in [|a|]\} \cup \{(|a| + i, i) \mid i \in [|a|]\} \in \mathbf{Rel}_S(a \square a, a), \quad \nabla_a = \emptyset \in \mathbf{Rel}_S(e,a),$$

$$a \Delta a = \{(i,i) \mid i \in [|a|]\} \cup \{(i, |a| + i) \mid i \in [|a|]\} \in \mathbf{Rel}_S(a, a \square a), \quad \perp_a = \emptyset \in \mathbf{Rel}_S(a,e),$$

l the following sets $R_\nabla = \{a \nabla a \mid a \in S^*\}$, $R_\Delta = \{a \Delta a \mid a \in S^*\}$, $R_\nabla = \{\nabla_a \mid a \in S^*\}$ and $_ = \{\perp_a \mid a \in S^*\}$. By using a parameter $P \subseteq A := \{R_\nabla, R_\Delta, R_\nabla, R_\perp\}$ the subcategory $P\mathbf{Rel}_S$ defined as the least ssmc of $\mathbf{Rel}_S$ which has the same objects as $\mathbf{Rel}_S$ and contains the rphisms $\cup \{R \mid R \in P\}$. For instance, $\emptyset\mathbf{Rel}_S$ is the category of finite S-sorted bijections, $\{R_\nabla\}\mathbf{Rel}_S$ that of surjections, $\{R_\nabla\}\mathbf{Rel}_S$ that of injections, $\{R_\nabla, R_\nabla\}\mathbf{Rel}_S$ that of ctions etc.

The additional structure (the functor G_P): Let C be the category of ssmc-ies, M that of noids and $Ob: C \to M$ the functor which forgets the morphisms. The definition of $G_P: C \to M$ obtained from the definition of G_A, given below, by restriction to components corresponding elements of P. The functor G_A is defined by:

For a ssmc $(B, \square, e, \gamma)$ the corresponding monoid $G_A(B)$ is defined as follows:

* The axioms of ssmc are B1-10 on page 15.

(i) Its elements are 5-tuples $(a,\nabla,\Delta,O,\perp)$, where $a\in Ob(B)$, $\nabla\in B(a\square a,a)$, $\Delta\in B(a,a\square a)$, $O\in B(e,a)$ and $\perp\in B(a,e)$ satisfy the following identities:

(1) $(\nabla\square 1_a)\nabla=(1_a\square\nabla)\nabla$ (1°) $\Delta(\Delta\square 1_a)=\Delta(1_a\square\Delta)$

(2) $\nabla_{a,a}\nabla=\nabla$ (2°) $\Delta\nabla_{a,a}=\Delta$

(3) $(1_a\square O)\nabla=1_a$ (3°) $\Delta(1_a\square\perp)=1_a$

(4) $O\nabla=O\square O$ (4°) $\Delta\perp=\perp\square\perp$

(5) $O\perp=1_e$

(6) $\Delta\nabla=1_a$

(7) $\nabla\Delta=(\Delta\square\Delta)(1_a\square\nabla_{a,a}\square 1_a)(\nabla\square\nabla)$;

ii) Its operation is defined by $(a,\nabla,\Delta,O,\perp)\cdot(b,\nabla',\Delta',O',\perp')=(a\square b,\ (1_a\square\nabla_{b,a}\square 1_a)(\nabla\square\nabla'),\ \Delta\square\Delta')(1_a\square\nabla_{a,b}\square 1_b),\ O\square O',\ \perp\square\perp')$;

● For a morphism $H\in C(B,B')$ the corresponding morphism $G_A(H)$ maps $(a,\nabla,\Delta,O,\perp)$ into $H(a)$, $H(\nabla)$, $H(\Delta)$, $H(O)$, $H(\perp)$.

The categories C_P: The objects are pairs (B,F), which consist of an object B of Ċ and of a monoid) morphism $F: Ob(B)\to G_P(B)$ such that $FU^P_B=1_{Ob(B)}$, where $U^P: G_P\to Ob$ is the natural ransformation which forgets the additional struture, i.e., $U^P(a,\dots)=a$. The morphisms of $_P((B,F),(B',F'))$ are those morphisms $H\in C(B,B')$ that fulfil $F\ G_P(H)=Ob(H)\ F'$.

$PRel_S$ as an object in C_P, namely $(PRel_S,F_P)$: The definition of F_P is obtained from the efinition of F_A, given below, by restriction to components corresponding to elements of P. he monoid morphism $F_A: Ob(ARel_S)=S^*\to G_A(Rel_S)$ is defined by $F_A(a)=(a,a\nabla a,a\Delta a,O_a,\perp_a)$.

THEOREM. If (B,F) is an object in C_P, then every monoid morphism $H: S^*\to Ob(B)$ has a ique extension to a morphism in C_P from $(PRel_S,F_P)$ to (B,F).

COROLLARY. The category of finite S-sorted bijections $\emptyset Rel_S$ forms the ssmc freely enerated by the set S of objects.

Our interest in finite relations comes from theoretical computer science. In the theory of owchart schemes the finite relations play a similar role as the numbers in classical algebra [1].

[1] Căzănescu V.E., Ştefănescu Gh., Towards a new algebraic foundation of flowchart scheme theory, INCREST Preprint Series in Mathematics No.43/1987.

[2] MacLane S., Categories for the working mathematician, Springer-Verlag, 1971.

Virgil Emil Căzănescu
Faculty of Mathematics
University of Bucharest
Str. Academiei 14
70109 Bucharest, Romania.

Gheorghe Ştefănescu
Department of Mathematics
I N C R E S T
Bdul Pacii 220
76922 Bucharest, Romania.

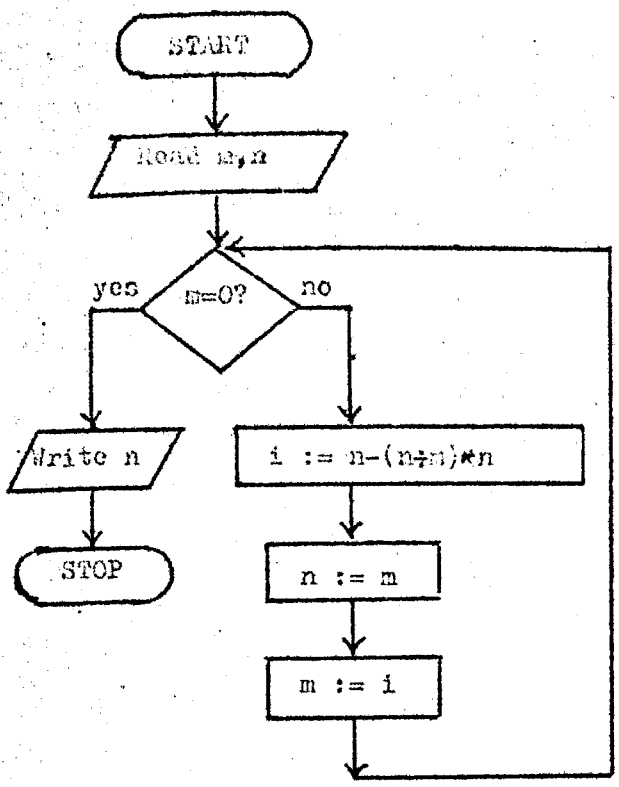


Figure 1. A usual, concrete flowchart

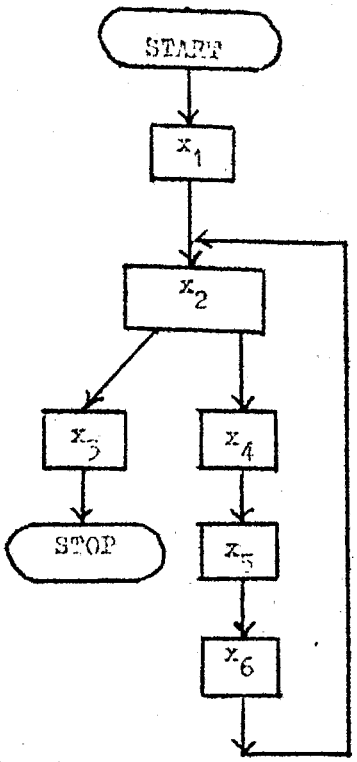


Figure 2. A usual, abstract flowchart scheme

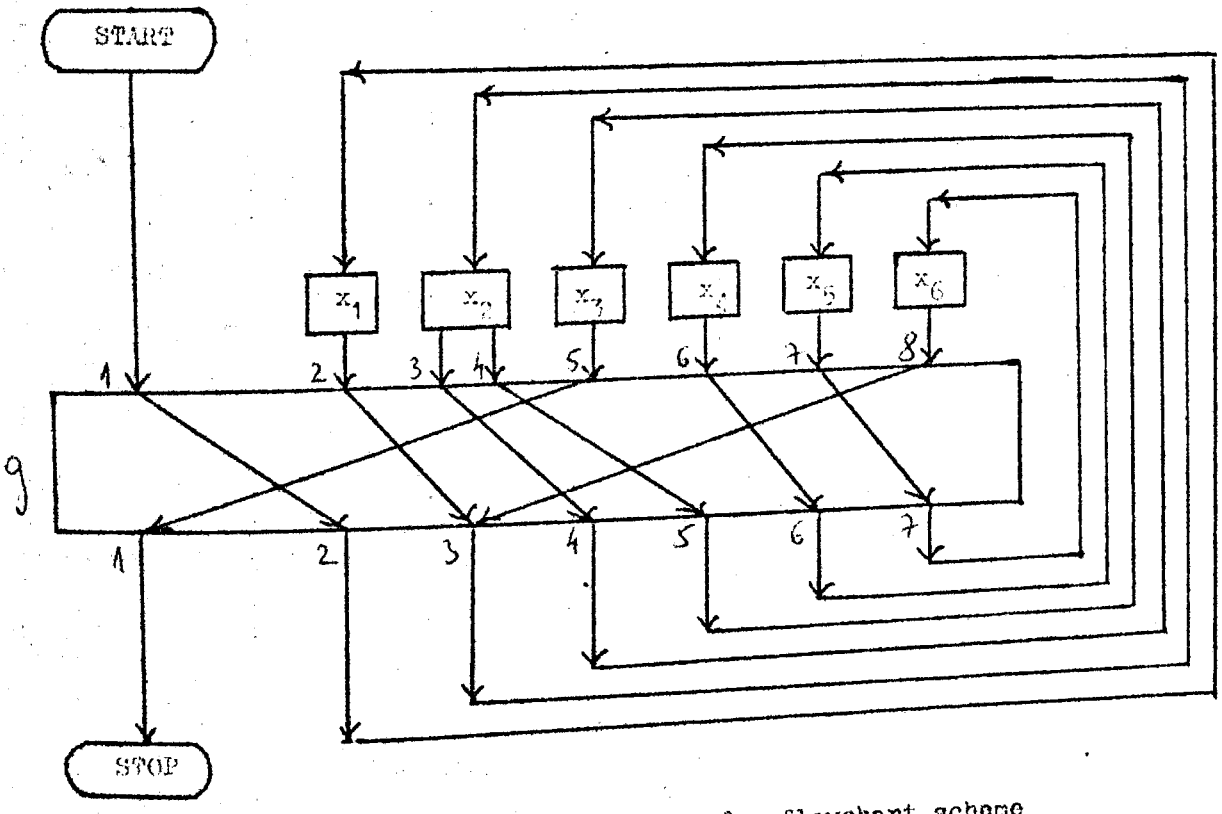


Figure 3. The normal form of a flowchart scheme

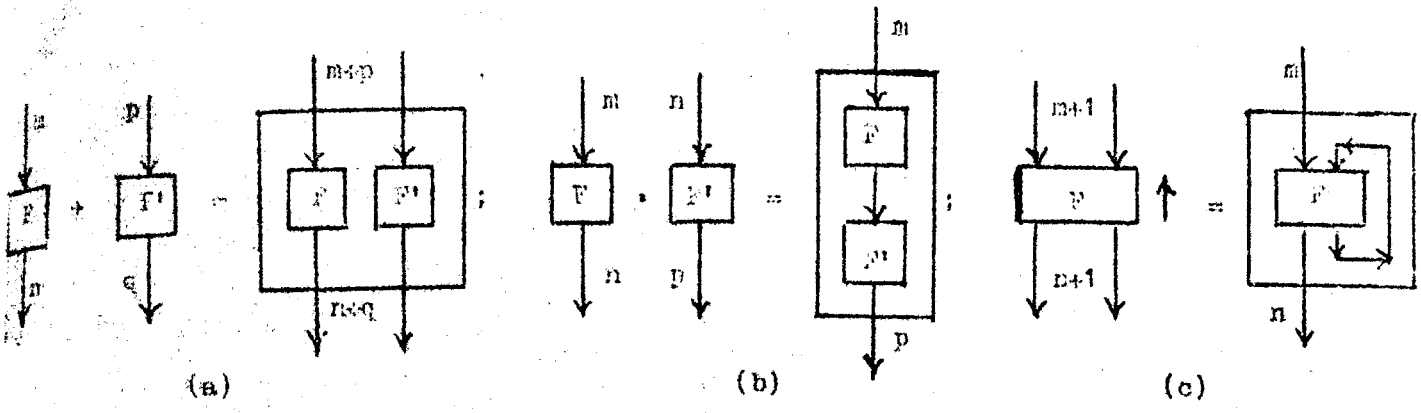


Figure 4. The basic operations on flowchart schemes: (a) "+" denotes sum; (b) "." denotes composition; (c) "↑" denotes feedback.

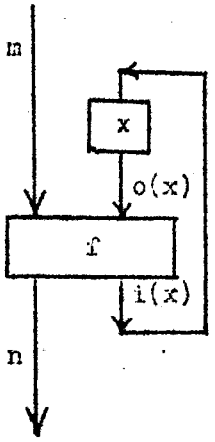


Figure 5. A concise picture of a scheme in normal form.

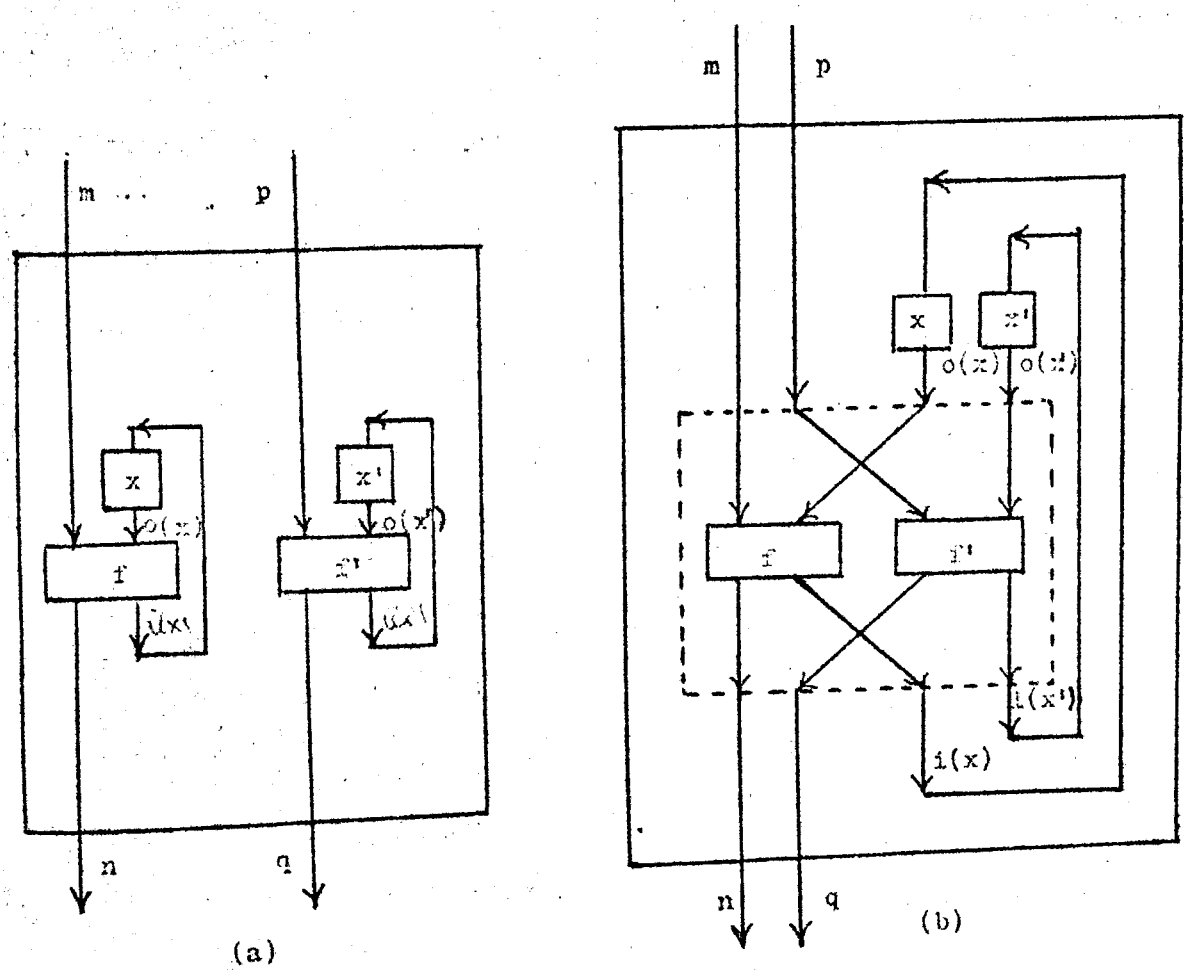


Figure 6. The normal form of the sum of two schemes in normal form.

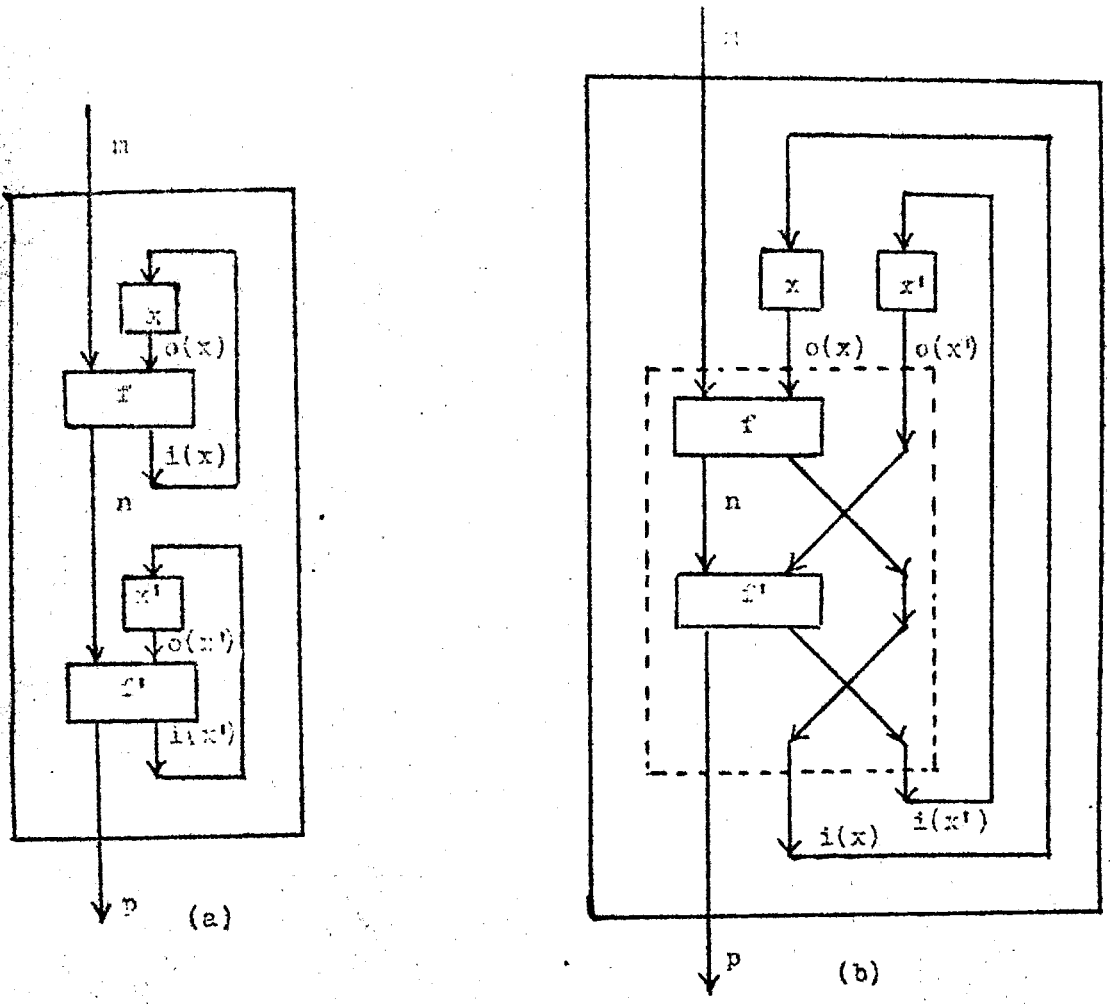


Figure 7. The normal form of the composite of two schemes in normal form.

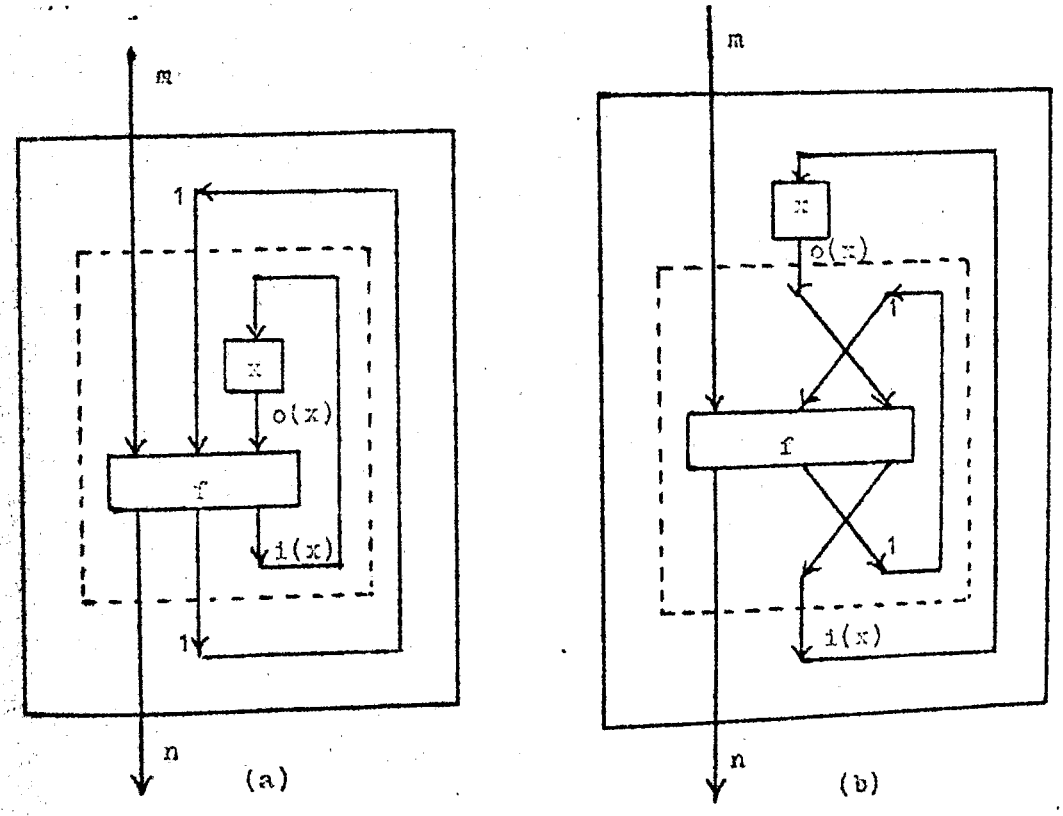


Figure 8. The normal form of the feedback of a scheme in normal form.

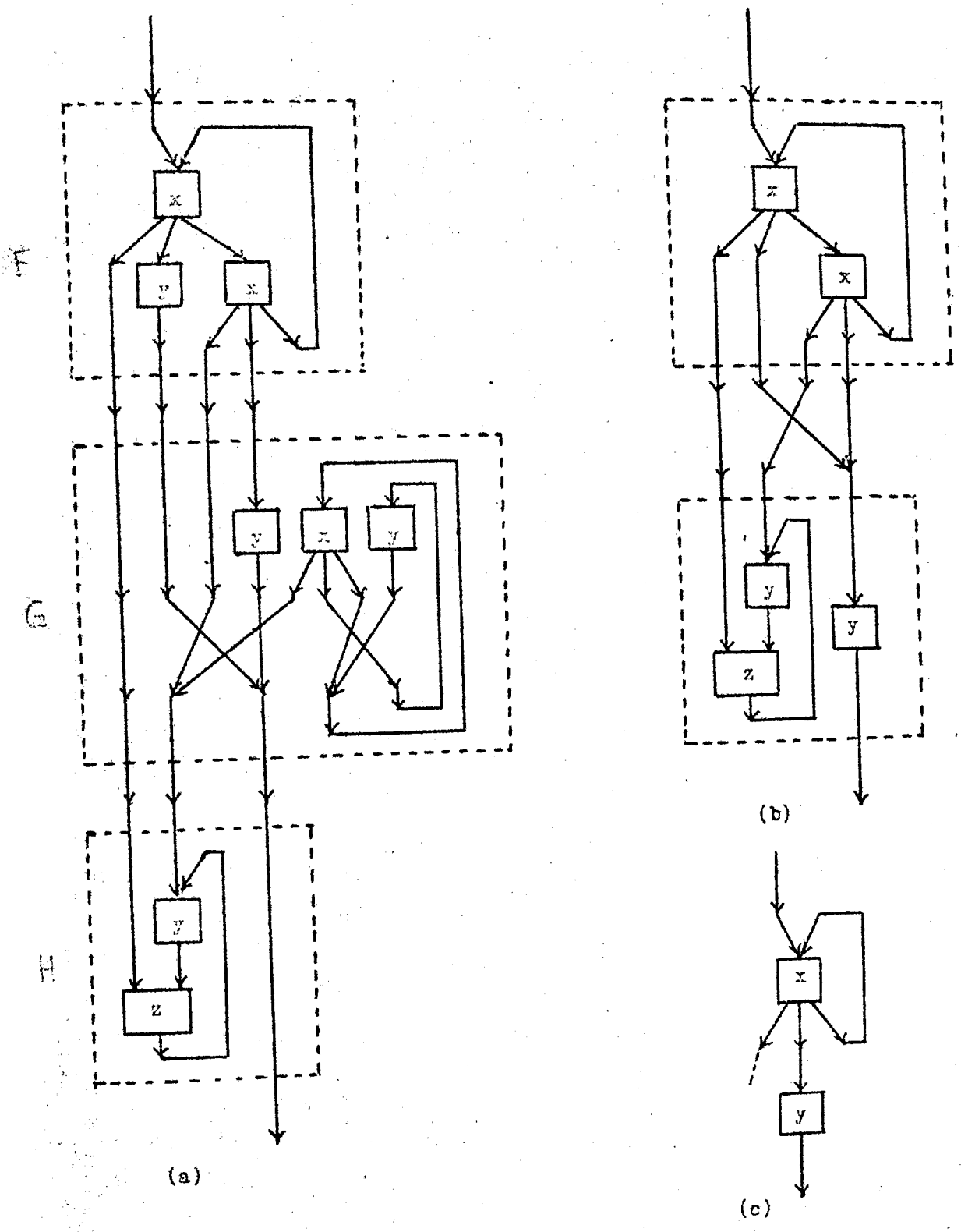
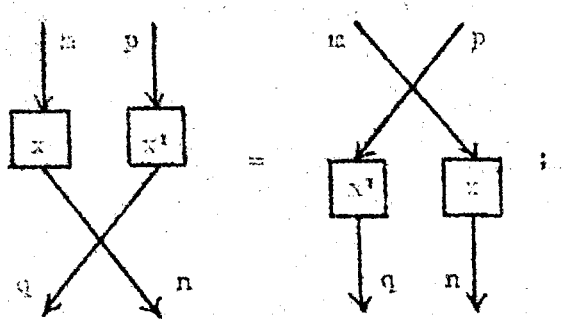
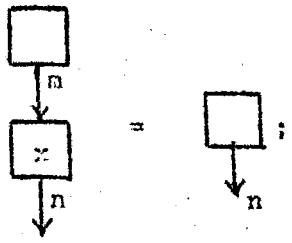
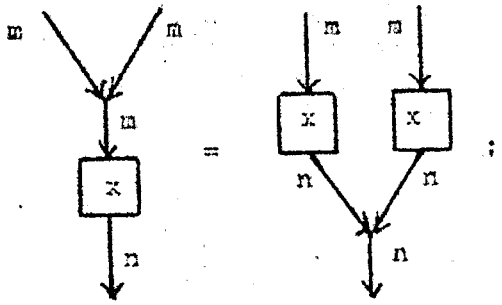


Figure 9. A schema (a); its complete minimization (b); and its deterministic minimization (c).







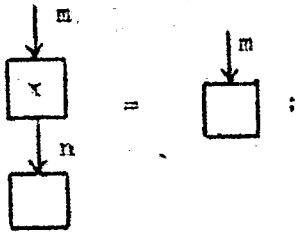
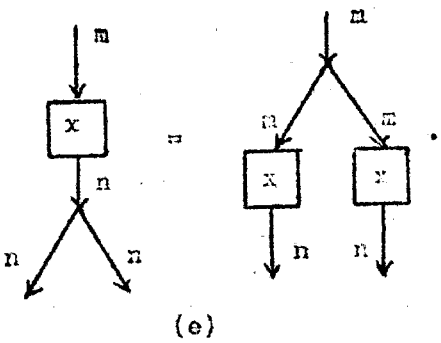


Figure 10. Simple generators for simulation: (a)-(e).



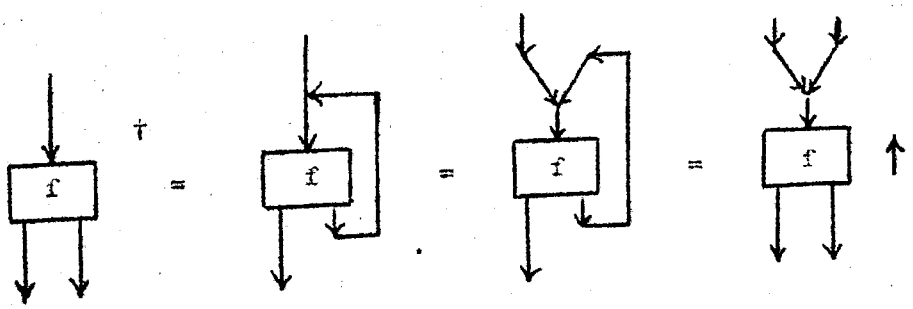


Figure 10bis. Iteration ⇒ Feedback + Tupling.

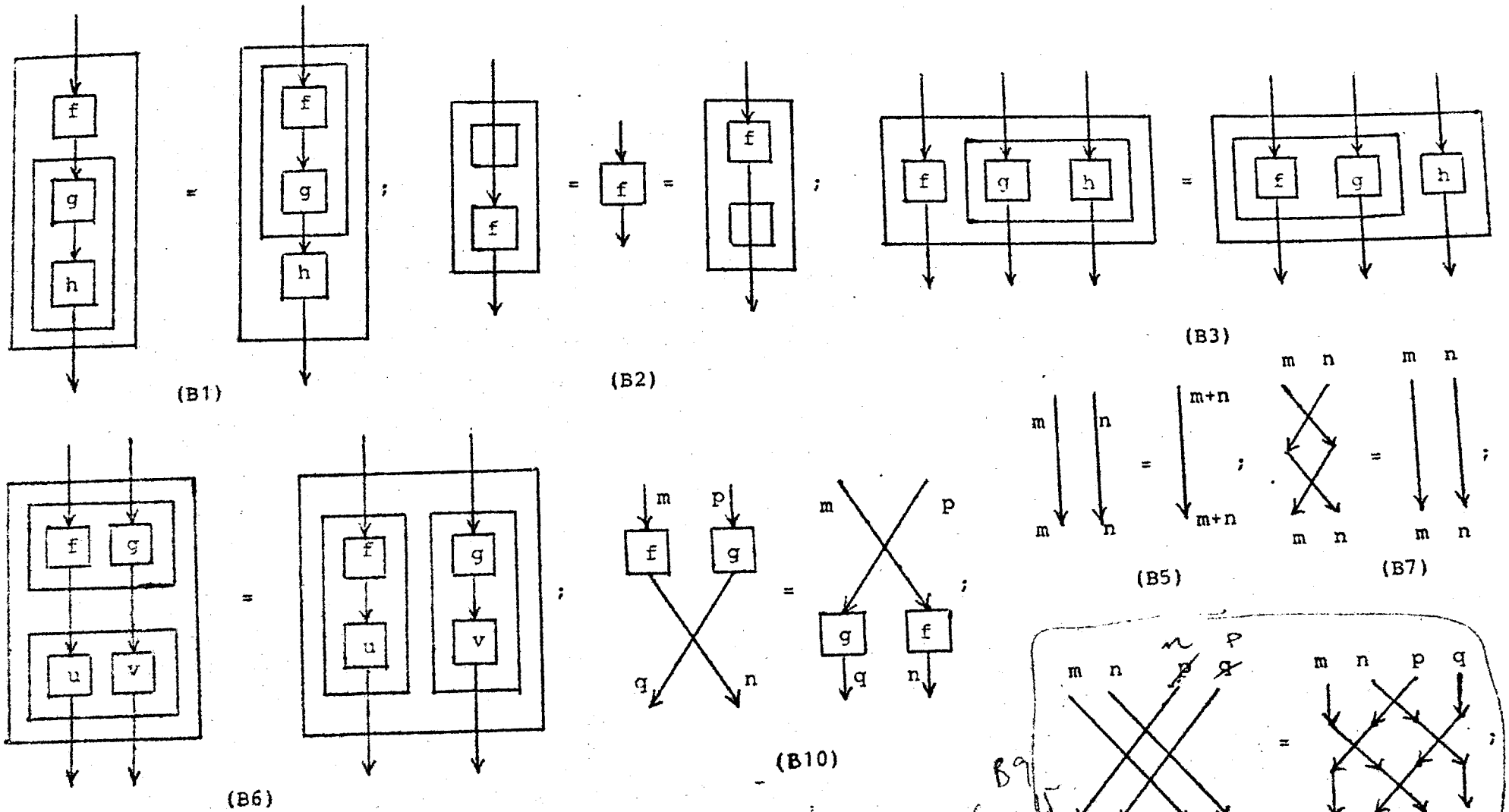
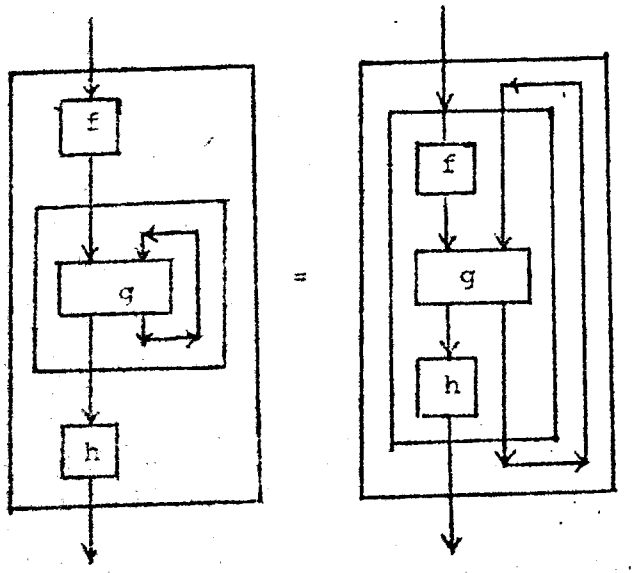


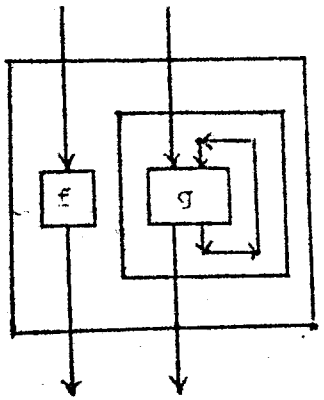
Figure 1. The axioms that define a biflow.



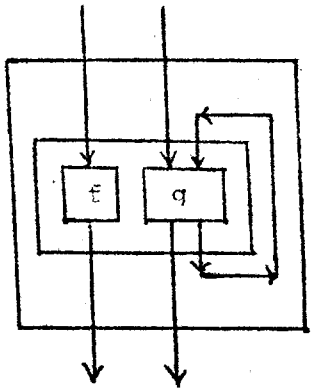
=

(B11)

;

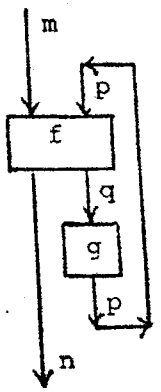


=

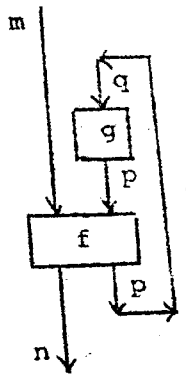


(B12)

;



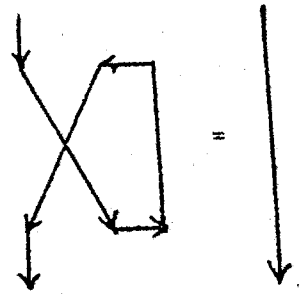
=



(B13)

;

The axioms (B4), (B8) and (B14) cannot be illustrated.

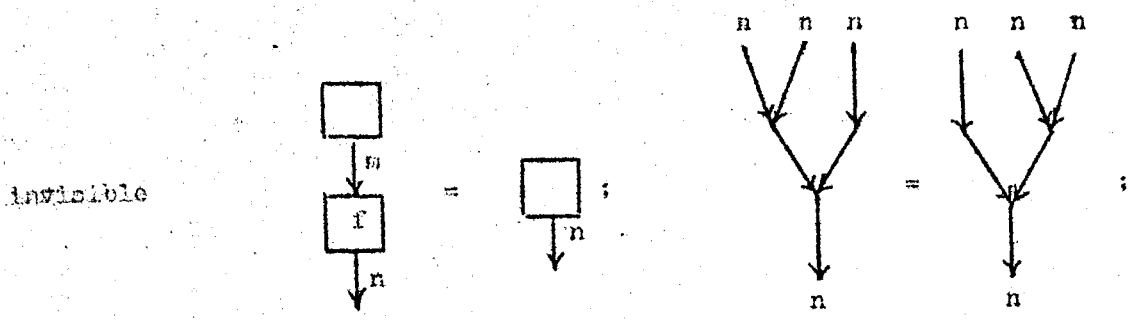


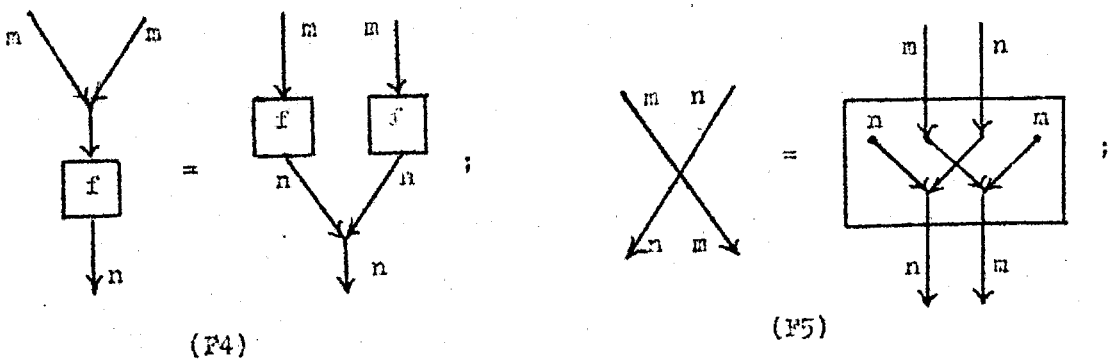
=

;

(B15)

Figure 12 (continued).





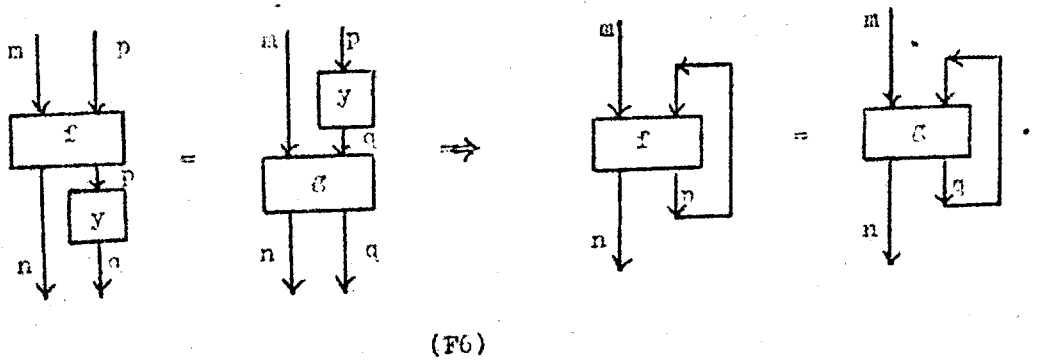


Figure 12. The axioms that are to be added to those in Figure 11 in order to define a funflow.

INSTITUTUL NAȚIONAL PENTRU
CREAȚIE ȘTIINTIFICĂ ȘI TEHNICĂ
Bd. Păcii 220
79622 București, ROMÂNIA

INSTITUTUL DE MATEMAT
Str. Academiei 14
70109 București, ROMÂNIA