
INSTITUTUL DE MATEMATICA
AL ACADEMIEI ROMANE

PREPRINT SERIES OF THE INSTITUTE OF MATHEMATICS

OF THE ROMANIAN ACADEMY

ISSN 0250 3638

# TRANSLATIONS BETWEEN FLOWCHART SCHEMES AND PROCESS GRAPHS

by

J.A. BERGSTRA AND GH. STEFANESCU

PREPRINT Nr.9/1993

# TRANSLATIONS BETWEEN FLOWCHART SCHEMES AND PROCESS GRAPHS

by

J.A. Bergstra and Gh. Stefanescu

June, 1993

Institute of Mathematics of the Romanian Academy

P.O. Box 1-764, RO-70700, Bucharest, Romania.

# Translations Between Flowchart Schemes and Process Graphs*

J.A. Bergstra and Gh. Ştefănescu

Programming Research Group, University of Amsterdam
P.O. Box 41882, 1009 DB Amsterdam
and
Institute of Mathematics, Romanian Academy
P.O. Box 1-764, RO-70700 Bucharest

**Abstract.** In a flowchart scheme an atomic action is modelled as a vertex (box), while in a process graph an atomic action is modelled as an edge. We define translations between these two graphical representations. By using these translations, we show that the classical bisimulation equivalence on process graphs coincides with the natural extension of the classical step-by-step flowchart equivalence to the nondeterministic case. This result allows us to translate axiomatisation results from flowcharts to processes and viceversa.

## 1 Introduction

The main difference between a flowchart scheme and a process graph (or a finite automaton) is the way of modelling the atomic actions: in a flowchart scheme an atomic action is modelled as a vertex (box), while in a process graph (or in an automaton) an atomic action is modelled as an edge.

Both models use the "one-step-further" description. For process graphs this is "state — action ⟶ new state", while for a flowchart scheme (built up with boxes of actions) this is "action — state ⟶ new action". This shows that by a "half step" we may pass from one formalism to the other.

Using this idea, we define translations between these two graphical representations. (The translation from flowcharts to processes may be defined *only* in the particular case where the flowcharts are built up with one-input/one-output atoms.) Then, we show that the classical bisimulation equivalence on process graphs coincides with the natural extension of the classical step-by-step flowchart equivalence to the nondeterministic case and apply this result for translating some axiomatisation results.

For deterministic schemes, this step-by-step equivalence was introduced by Elgot in [9]. Technically, it may be defined as the equivalence relation generated by simulation via functions cf. [11, 15], a definition that has a straightforward extension to the nondeterministic case.

* This research was completed while the second author was visited the Programming Research Group of the University of Amsterdam.

To appear in: Proceedings FCT'93

## 2 Flowchart Schemes

We briefly describe an algebraic formalism for flowchart schemes and their behaviour, known as *the calculus of flownomials*; see [6, 8, 15, 16], for example.

Let $X = \{X(m,n)\}_{m,n\geq 0}$ be a family of doubly-ranked sets. An element $x \in X(m,n)$ represents an atomic scheme with $m$ inputs and $n$ outputs. In the examples given in Fig. 1, $s \in X(1,1)$ and $p, q \in X(1,2)$.

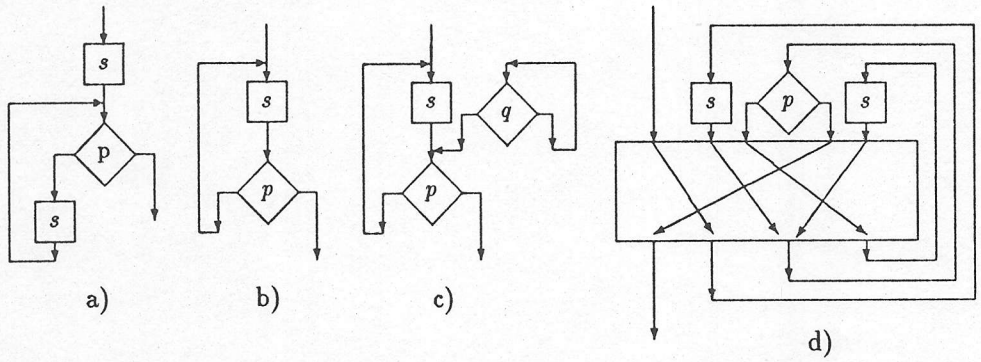



**Fig. 1.** Three flowcharts a), b) and c). In d) is given a normal form of a).

A flowchart picture may be redrawn in a normal form as it is shown in Fig.1.d for the scheme in Fig.1.a. This shows that a flowchart $F$ may be represented by a pair $(w, f)$, where $w$ is a sequence of all the atoms of $F$ and $f$ is a relation which represents the connections of $F$. For example, the scheme in Fig.1.a may be represented by the pair $(s \oplus p \oplus s, f)$, where $f \in \mathbb{R}\mathrm{el}(5,4)$ is the function that maps $1,2,3,4,5$ into $2,3,4,1,3$, respectively, or by the *normal form flownomial expression* $((I \oplus s \oplus p \oplus s) \cdot f)\uparrow^3$, with $f$ as above.

For the deterministic flowchart schemes, a standard equivalence is that given by the unfoldment, i.e., two schemes are equivalent iff they unfold into the same tree. An example is provided by the schemes in Fig.1.a,b,c. This equivalence characterizes the *step-by-step behaviour* of the flowchart schemes cf. [5, 6, 8, 9, 12, 15] (see also [10], for a connection with automata).

In the case of nondeterministic schemes, this definition is difficult to be used for defininining the step-by-step behaviour. A nondeterministic scheme may be easily defined using the normal form representation, namely, we have to allow the connection box $f$ above to be an arbitrary relation. What about the unfoldment of such a nondeterministic scheme? Since the nondeterministic choice operator $\wedge$ is symmetric, idempotent and associative the unfolding tree cannot be defined in a unique way (we may interchange the branches in some choice points, or we may delete a branch of a choice if both branches continue with the same subtree, etc.).

There is another way to define a step-by-step behaviour for nondeterministic schemes. In [11, 15], it is shown that in the case of deterministic schemes the equivalence relation corresponding to the (input) step-by-step behaviour may be defined using a transformation of schemes, the so-called "simulation via functions."[2]

[2] The definition of this transformation may be seen as an extension of the usual definition

**Definition 1.** Two schemes represented by $F = (x_1 \oplus \ldots \oplus x_m, f)$ and $G = (y_1 \oplus \ldots \oplus y_n, g)$ are *similar* via a function $r \in \mathbb{F}\mathrm{n}(m,n)$ (notation: $F \to_r G$, or $F \to_{\mathbb{F}\mathrm{n}} G$ when $r$ has no importance) iff

(i) $(i,j) \in r \Rightarrow x_i = y_j$;

(ii) $f \cdot (I \oplus i(r)) = (I \oplus o(r)) \cdot g$.[3]

Intuitively, if $r$ is a surjection, then $F \to_r G$ means that we may identify the vertices of $F$ corresponding to $Ker(r) = \{(i,j) \mid r(i) = r(j)\}$ because they have the same label and their connections became equal after identification, and $G$ is the resulting scheme. If $r$ is an injection, $F \to_r G$ means that $F$ is a subscheme of $G$ via $r$, that is the part in $G$ corresponding to the complement of the image of $r$ is not accessible from the remaining one.

*Example 1.* The schemes in Fig.2.a,b may be represented by $F_a = (s \oplus p \oplus s, f_a)$, where $f_a$ maps $1,2,3,4,5$ into $2,3,4,1,3$, respectively and by $F_b = (s \oplus p, f_b)$, where $f_b$ maps $1,2,3,4$ to $2,3,2,1$, respectively. Then $F_a \to_r F_b$, where $r \in \mathbb{F}\mathrm{n}(3,2)$ is the function that maps $1,2,3$ to $1,2,1$, respectively. Indeed, the extension of $r$ to inputs is $r$ and to outputs is $o(r)$ = "the function that maps $1,2,3,4$ to $1,2,3,1$, respectively". Now, one may easily check that conditions (i) and (ii) hold.

On the other hand, if $f_c$ is the function that maps $1,2,3,4,5,6$ to $2,3,2,1,3,4$, respectively and $F_c = (s \oplus p \oplus q, f_c)$, then $F_b \to_r F_c$, where $r \in \mathbb{F}\mathrm{n}(2,3)$ is the injective function that maps $1,2$ to $1,2$, respectively.

Simulation via functions may be defined whenever the class of connections contains functions, in particular in the case of nondeterministic flowchart schemes (in which case this class is $\mathbb{R}\mathrm{el}$).

*Notation:* We denote the equivalence relation generated by simulation via functions by $\leftrightarrow_{\mathbb{F}\mathrm{n}}$. (In the case of nondeterministic schemes this equivalence may be interpreted as a formalization of the step-by-step behaviour, too.)

## 3 Process Graphs

Let $A$ be a set of atomic actions. The set $G(A)$ of process graphs built up with atoms in $A$ consists of rooted finite directed graphs in which each edge has a label in $A$ and in which some nodes are distinguished as terminal. An example is given in Fig.4.a. (This model was developed in [1, 2, 3], for example.)

A basic equivalence on communicating processes was introduced by Milner using the observable behaviour, see [13, 14]. The idea is that the equivalent processes not only perform the same sequence of actions but also their branching structure is identical.

Technically, this idea is modelled by bisimulation.

---

of "graph morphism" to the case of flowchart schemes, that are considered "hypergraphs" built up with many-input/many-output atoms.

[3] Here $i(r)$ represents the "block" extension of $r$ to inputs, i.e., if $r$ relates two variables $x_i$ and $y_j$, then $i(r)$ relates the first input of $x_i$ to the first input of $y_j$, the second to the second, etc.; similarly $o(r)$ for outputs. And for a relation $R$, we have denoted by $I \oplus R$ the relation $\{(1,1)\} \cup \{(i+1, j+1) \mid (i,j) \in R\}$.

**Definition 2.** Let $p$ and $q$ be two process graphs and $R$ a relation between the nodes of $p$ and the nodes of $q$. We say $R$ is a *bisimulation* between $p$ and $q$ (notation: $p \leftrightarrows_R q$) iff the following four conditions hold.

1. The roots of $p$ and $q$ are related via $R$;

2. If $s \xrightarrow{a} s'$ is an edge in $p$ and $(s,t) \in R$, then there exists an edge $t \xrightarrow{a} t'$ in $q$ such that $(s',t') \in R$;

3. Viceversa, if $t \xrightarrow{a} t'$ is an edge in $q$ and $(s,t) \in R$, then there exists an edge $s \xrightarrow{a} s'$ such that $(s',t') \in R$;

4. If $(s,t) \in R$ then $s$ is terminal iff $t$ is terminal.

For example, the process graphs in Figs.2 and 4.a are bisimilar. Let us notice that the simulation relation is not an equivalence (it is not symmetrical), whilst the bisimulation relation is.

## 4 Translations

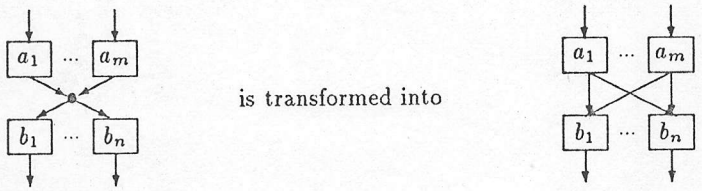
Suppose $A$ is a set of atomic one-input/one-output actions. Denote the set of flowchart schemes with one input and one output built up with atoms in $A$ and connections in IRel by $\mathbb{F}\mathrm{l}_{A,\mathbb{I}\mathrm{Rel}}(1,1)$ and the process graphs built up with atoms in $A$ by $G(A)$. We define here two transformations $\mathtt{pg} : \mathbb{F}\mathrm{l}_{A,\mathbb{I}\mathrm{Rel}}(1,1) \rightarrow G(A)$ and $\mathtt{fs} : G(A) \rightarrow \mathbb{F}\mathrm{l}_{A,\mathbb{I}\mathrm{Rel}}(1,1)$.

### 4.1 From Processes to Flowcharts

**Definition 3.** (definition of fs) Suppose $g$ is a process graph in $G(A)$. Then $\mathtt{fs}(p)$ is the flowchart scheme obtained by using the following procedure:

(fs-1) Replace an edge  in $g$ by the flowchart 

(fs-2) Then delete the vertices in the resulting picture by making direct connections between the atomic flowcharts we have introduced, i.e.,

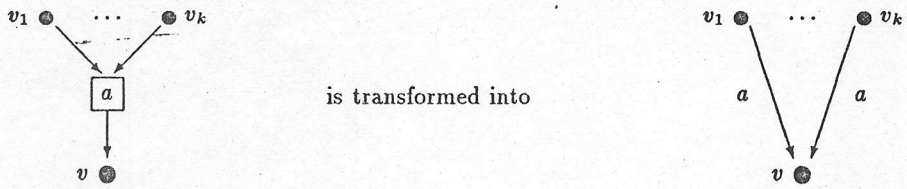



### 4.2 From Flowcharts to Processes

**Definition 4.** (definition of pg) Suppose $F$ is a nondeterministic flowchart scheme in $\mathbb{F}\mathrm{l}_{A,\mathbb{I}\mathrm{Rel}}(1,1)$. Then $\mathtt{pg}(F)$ is the process graph obtained by using the following procedure:

(pg-1) Put vertices on the input arrow of the flowchart and on each output arrow of the atoms (but before the possible choice points of the continuation);

(pg-2) Replace the atomic boxes by edges using the transformation




(In the case $k = 0$, this means that the atomic box $a$ gives zero edges.)

*Note 5.* Another way to define the transformation pg is by using the normal form representation of flowcharts and the recursive specification of processes. That is, if

$$F = [(\mathrm{I} \oplus a_1 \oplus \ldots \oplus a_k) \cdot f] \uparrow^k$$

is a normal form flownomial expression representing a flowchart $F$, then $\mathrm{pg}(F)$ is the first component of the unique solution in $G(A)$ of the system

$$X_i = \sum_{j \in f(i) - \{1\}} a_{j-1} X_j + d_i, \quad \text{for} \quad i = 1, \ldots, k+1$$

where $f(i)$ denotes $\{j \mid (i,j) \in f\}$ and $d_i = \epsilon$ if $1 \in f(i)$, otherwise $d_i = \delta$.

*Example 2.* An example for the action of `fs` is given in Fig.2. Clearly, the number of the labelled edges in $p$ is equal to the number of the labelled boxes in $\mathtt{fs}(p)$.

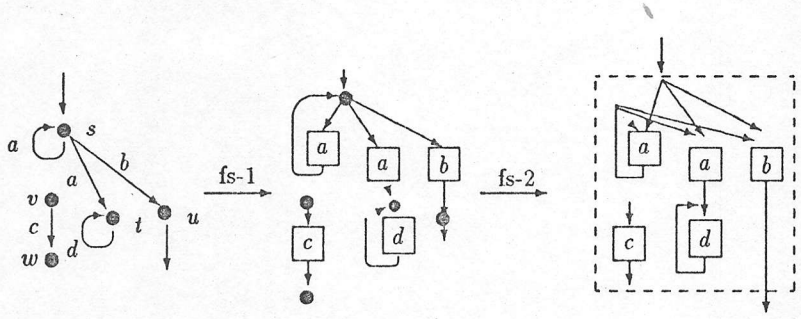



**Fig. 2.** From processes to flowcharts

*Example 3.* An example for the action of pg is given in Fig.3. We see that the number of the edges in $\mathrm{pg}(F)$ introduced by a box is equal to the number of the incoming arrows into that box. In this example $a$ produces one edge, $b$ three and $c$ zero.

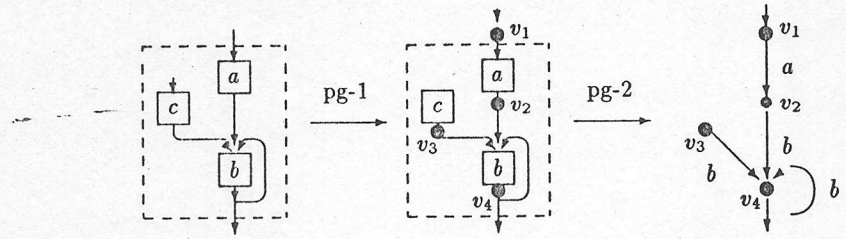



**Fig. 3.** From flowcharts to processes

## 5 Relating process and flowchart equivalences

The informal presentation of the bisimulation equivalence is closed to the intuitive meaning of the equivalence given by the unfolding of the nondeterministic flowchart schemes. The following proposition makes this connection precise.

**Proposition 6.** *The above transformations* pg : $\mathbb{F}\mathrm{l}_{A,\mathbb{R}\mathrm{el}}(1,1) \to G(A)$ *and* fs : $G(A) \to \mathbb{F}\mathrm{l}_{A,\mathbb{R}\mathrm{el}}(1,1)$ *have the following properties:*

*(A) If* $F \in \mathbb{F}\mathrm{l}_{A,\mathbb{R}\mathrm{el}}(1,1)$, *then* $\mathtt{fs}(\mathrm{pg}(F)) \leftrightarrow_{\mathbb{F}\mathrm{n}} F$.
*If* $p \in G(A)$, *then* $\mathrm{pg}(\mathtt{fs}(p)) \leftrightarrows p$.

*(B) If* $F, G \in \mathbb{F}\mathrm{l}_{A,\mathbb{R}\mathrm{el}}(1,1)$ *and* $F \leftrightarrow_{\mathbb{F}\mathrm{n}} G$, *then* $\mathrm{pg}(F) \leftrightarrows \mathrm{pg}(G)$.
*If* $p, q \in G(A)$ *and* $p \leftrightarrows q$, *then* $\mathtt{fs}(p) \leftrightarrow_{\mathbb{F}\mathrm{n}} \mathtt{fs}(q)$.

Regarding to property (A), we first notice that due to transformation (pg-2) we should not expect that these transformations are inverses one to the other. Nevertheless their composites behave quite nicely:

- The effect of applying, first fs, then pg to a process graph, is the transformation in which each vertex in $p$ is split into a number of copies, one for each incoming arrow. In the example given in Fig.4.a, $s$ is split into $(s, \downarrow)$ and $(s, \downarrow a)$, $t$ into $(t, \downarrow a)$ and $(t, \downarrow d)$, $u$ into $(u, \downarrow b)$, $v$ into nothing, and $w$ into $(w, \downarrow c)$. So, we get a graph bisimilar to the original one.

- The effect of applying, first pg, then fs to a flowchart scheme, is the transformation in which each box is moved backward on its incoming arrows, making one copy for each such an arrow. In the example given in Fig.4.b, $b$ is split 3 times, $a$ one time, and $c$ zero times. So, we get a scheme similar to the original one.

In order to prove the first half of property (B) we have shown that $F, G \in \mathbb{F}\mathrm{l}_{A,\mathbb{R}\mathrm{el}}(1,1)$ and $F \to_r G$ imply $\mathrm{pg}(F) \leftrightarrows \mathrm{pg}(G)$. In this proof, we effectivelly used the fact that the morphism for simulation $r$ is a function and not an arbitrary relation !

For the second implication in property (B), we have to show that a bisimulation between two process graphs may be modelled by a chain of simulations and inverses of simulations. For this, it might be helpful to have a characterization of bisimulation in terms of bisimulation via functions. The result given below is based on a construction of a common refinement of two bisimilar processes.

**Lemma 7.** *(interpolation property) If* $p, q \in G(A)$ *and* $p \leftrightarrows_{f^{-1} \cdot g} q$, *with* $f, g$ *functions, then there exists an* $r \in G(A)$ *such that* $p \leftrightarrows_{f^{-1}} r$ *and* $r \leftrightarrows_g q$.

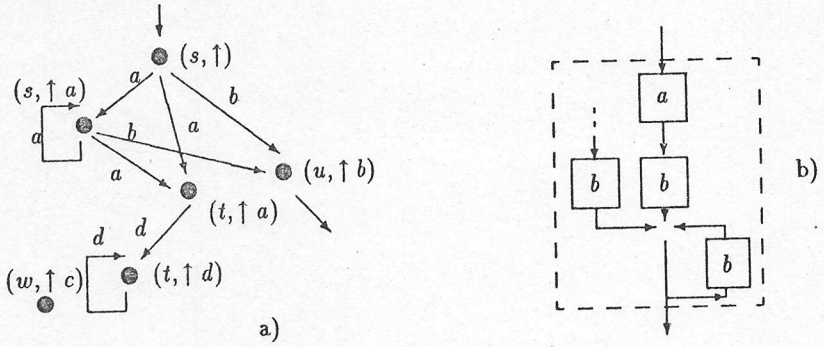



**Fig. 4.** a) The result of the application of $\mathtt{fs} \cdot \mathtt{pg}$ to the process in Fig.2. b) The result of the application of $\mathtt{pg} \cdot \mathtt{fs}$ to the flowchart in Fig.3.

Now, the other half of property (B) follows from this result and the fact that

if $p, q \in G(A)$ and $p \rightleftharpoons_r q$, with $r$ function, then $\mathtt{fs}(p) \leftrightarrow_{\mathbb{F}n} \mathtt{fs}(q)$.

All the above show that:

**Theorem 8.** *The translations* pg *and* fs *give a bijection between the classes of bisimilar process graphs in $G(A)$ and the classes of flowchart schemes in $\mathbb{F}\mathrm{l}_{A,\mathbb{R}\mathrm{el}}(1,1)$ which are equivalent using simulations via functions.*

## 6 Axiomatising Flowchart / Process Behaviours

As an application of the translations we have defined and of the previous result, we show here how an axiomatisation for the classes of bisimilar process graphs may be obtained. By the above theorem, such an axiomatisation may be obtained by translating an axiomatisation for the classes of $\leftrightarrow_{\mathbb{F}n}$-equivalent schemes in $\mathbb{F}\mathrm{l}_{A,\mathbb{R}\mathrm{el}}(1,1)$.

First we recall some results known in the axiomatisation of abstract flowchart schemes, see [15, 16, 17]. These abstract schemes are obtained connecting the atoms by elements in an abstract algebraic structure, not neccessarily one of relations included in $\mathbb{R}$el.

The core of the axioms given in Table 1 is

3.1 $a\alpha$-flow = B1-B10 + R1-R3 + F1-F2

giving a presentation of abstract flowchart schemes (i.e., labelled hypergraphs), cf. [6, 17]. For axiomatizing different types of behaviours, we use subsets of axioms in Table 1 defined using the notation $xy$, with $x \in \{a, b, c, d\}$ and $y \in \{\alpha, \beta, \gamma, \delta\}$. A few of them are recalled here:

1.1 $a\delta$-ssmc = B1-B10 + A1-A4 + A12-A15;
1.2 $b\delta$-ssmc = $a\delta$-ssmc + A9 + A16-A17;
1.3 $d\delta$-ssmc = B1-B10 + A1-A19;

B1 $f \oplus (g \oplus h) = (f \oplus g) \oplus h$
B2 $\mathsf{I}_0 \oplus f = f = f \oplus \mathsf{I}_0$
B3 $f \cdot (g \cdot h) = (f \cdot g) \cdot h$
B4 $\mathsf{I}_a \cdot f = f = f \cdot \mathsf{I}_b$
B5 $(f \oplus f') \cdot (g \oplus g') = f \cdot g \oplus f' \cdot g'$
B6 $\mathsf{I}_a \oplus \mathsf{I}_b = \mathsf{I}_{a+b}$
B7 ${}^a\mathsf{X}^b \cdot {}^b\mathsf{X}^a = \mathsf{I}_{a+b}$
B8 ${}^a\mathsf{X}^0 = \mathsf{I}_a$
B9 ${}^a\mathsf{X}^{b+c} = ({}^a\mathsf{X}^b \oplus \mathsf{I}_c) \cdot (\mathsf{I}_b \oplus {}^a\mathsf{X}^c)$
B10 $(f \oplus g) \cdot {}^c\mathsf{X}^d = {}^a\mathsf{X}^b \cdot (g \oplus f)$ for $f : a \to c, g : b \to d$

**I. Axioms for ssmc-ies (symmetric strict monoidal categories)**

A1 $(\vee_a \oplus \mathsf{I}_a) \cdot \vee_a = (\mathsf{I}_a \oplus \vee_a) \cdot \vee_a$
A2 ${}^a\mathsf{X}^a \cdot \vee_a = \vee_a$
A3 $(\top_a \oplus \mathsf{I}_a) \cdot \vee_a = \mathsf{I}_a$
A4 $\vee_a \cdot \perp^a = \perp^a \oplus \perp^a$
A5 $\wedge^a \cdot (\wedge^a \oplus \mathsf{I}_a) = \wedge^a \cdot (\mathsf{I}_a \oplus \wedge^a)$
A6 $\wedge^a \cdot {}^a\mathsf{X}^a = \wedge^a$
A7 $\wedge^a \cdot (\perp^a \oplus \mathsf{I}_a) = \mathsf{I}_a$
A8 $\top_a \cdot \wedge^a = \top_a \oplus \top_a$

A9 $\top_a \cdot \perp^a = \mathsf{I}_0$
A10 $\vee_a \cdot \wedge^a = (\wedge^a \oplus \wedge^a) \cdot (\mathsf{I}_a \oplus {}^a\mathsf{X}^a \oplus \mathsf{I}_a) \cdot (\vee_a \oplus \vee_a)$
A11 $\wedge^a \cdot \vee_a = \mathsf{I}_a$

A12 $\top_0 = \mathsf{I}_0$
A13 $\top_{a+b} = \top_a \oplus \top_b$
A14 $\vee_0 = \mathsf{I}_0$
A15 $\vee_{a+b} = (\mathsf{I}_a \oplus {}^b\mathsf{X}^a \oplus \mathsf{I}_b) \cdot (\vee_a \oplus \vee_b)$
A16 $\perp^0 = \mathsf{I}_0$
A17 $\perp^{a+b} = \perp^a \oplus \perp^b$
A18 $\wedge^0 = \mathsf{I}_0$
A19 $\wedge^{a+b} = (\wedge^a \oplus \wedge^b) \cdot (\mathsf{I}_a \oplus {}^a\mathsf{X}^b \oplus \mathsf{I}_b)$

**II. Axioms for the additional constants $\top, \perp, \vee, \wedge$ (without feedback)**

R1 $f \cdot (g\uparrow^c) \cdot h = ((f \oplus \mathsf{I}_c) \cdot g \cdot (h \oplus \mathsf{I}_c))\uparrow^c$ (relating "$\uparrow$" and "$\cdot$")
R2 $f \oplus g\uparrow^c = (f \oplus g)\uparrow^c$ (relating "$\uparrow$" and "$\oplus$")
R3 $(f \cdot (\mathsf{I}_b \oplus g))\uparrow^c = ((\mathsf{I}_a \oplus g) \cdot f)\uparrow^d$ (shifting blocks on feedback)
for $f : a + c \to b + d, g : d \to c$

**III. Axioms for feedback**

F1 $\mathsf{I}_a\uparrow^a = \mathsf{I}_0$
F2 ${}^a\mathsf{X}^a\uparrow^a = \mathsf{I}_a$
F3 $\vee_a\uparrow^a = \perp^a$
F4 $\wedge^a\uparrow^a = \top_a$
F5 $[(\mathsf{I}_a \oplus \wedge^a) \cdot ({}^a\mathsf{X}^a \oplus \mathsf{I}_a) \cdot (\mathsf{I}_a \oplus \vee_a)]\uparrow^a = \mathsf{I}_a$

**IV. Axioms for the action of feedback on constants**

S1 $\top_a \cdot f = \top_b$
S2 $\vee_a \cdot f = (f \oplus f) \cdot \vee_b$
S3 $f \cdot \perp^b = \perp^a$
S4 $f \cdot \wedge^b = \wedge^a \cdot (f \oplus f)$

**V. The strong axioms ($f : a \to b$)**

FUNC$_E$: $f \cdot (\mathsf{I}_b \oplus y) = (\mathsf{I}_a \oplus y) \cdot g$ implies $f\uparrow^c = g\uparrow^d$,

where $E$ is a class of abstract relations (i.e., of terms written with $\oplus, \cdot, \mathsf{I}, X$ and some constants in $\top, \perp, \vee, \wedge$), $y : c \to d$ is in $E$ and $f : a + c \to b + c, g : a + d \to b + d$ are arbitrary

**VI. The functoriality rule**

Table 1. Algebra for flownomials

2.1 $a\delta$-ssmc with feedback = $b\delta$-ssmc + R1-R3 + F1-F3;
2.2 $b\delta$-ssmc with feedback = $a\delta$-ssmc with feedback;
2.3 $d\delta$-ssmc with feedback = B1-B10 + A1-A19 + R1-R3 + F1-F5;

3.2 $a\delta$-flow[4] = $a\delta$-ssmc with feedback + S1-S2 + $\mathrm{FUNC}_{\text{abstract functions}}$;
3.3 $b\delta$-flow = $b\delta$-ssmc with feedback + S1-S3 + $\mathrm{FUNC}_{\text{abstract partial functions}}$;
3.4 $d\delta$-flow[5] = $d\delta$-ssmc with feedback + S1-S4 + $\mathrm{FUNC}_{\text{abstract relations}}$.

**Theorem 9.** *(axiomatisation of finite relations, cf. [7, 17])*
*1.* $\mathbb{F}\mathrm{n}$ *is the free* $a\delta$*-ssmc;*
*2.* $\mathbb{P}\mathrm{fn}$ *is the free* $b\delta$*-ssmc; with feedback, it is the free* $b\delta$*-ssms with feedback;*
*3.* $\mathbb{R}\mathrm{el}$ *is the free* $d\delta$*-ssmc; with feedback, it is the free* $d\delta$*-ssmc with feedback.*

**Theorem 10.** *(the meaning of the equivalence* $\leftrightarrow_E$ *generated by simulation via a class of relations* $E$*, cf. [17, 15])*
*1. Two deterministic flowchart schemes (i.e., over* $\mathbb{P}\mathrm{fn}$*) are* $\leftrightarrow_{\mathbb{F}\mathrm{n}}$*-equivalent iff they have the same step-by-step computation sequences.*
*2. Two deterministic flowchart schemes are* $\leftrightarrow_{\mathbb{P}\mathrm{fn}}$*-equivalent iff they have the same input-output step-by-step computation sequences.*
*3. Two nondeterministic flowchart schemes (i.e., over* $\mathbb{R}\mathrm{el}$*) are* $\leftrightarrow_{\mathbb{R}\mathrm{el}}$*-equivalent iff they have the same input-output step-by-step computation sequences.*

**Theorem 11.** *(axiomatising flowchart behaviours) Let* $X$ *be a family of doubly-ranked variables. Let* $\leftrightarrow_{a\delta}$*,* $\leftrightarrow_{b\delta}$ *and* $\leftrightarrow_{d\delta}$ *denote the equivalences generated by simulation via abstract functions, abstract partial functions and abstract relations, respectively.*
*1. If* $T$ *is an* $a\delta$*-flow, then* $Fl_{X,T}/\leftrightarrow_{a\delta}$ *is the* $a\delta$*-flow freely generated by adding* $X$ *to* $T$ *(cf. [15]).*
*2. If* $T$ *is an* $b\delta$*-flow fulfilling a technical condition IP*[6]*, then* $Fl_{X,T}/\leftrightarrow_{b\delta}$ *is the* $b\delta$*-flow freely generated by adding* $X$ *to* $T$ *(see [17]).*
*3. If* $T$ *is a zero-sum-free* $d\delta$*-flow "with intersection" (cf. [16]), then* $Fl_{X,T}/\leftrightarrow_{d\delta}$ *is the* $d\delta$*-flow freely generated by adding* $X$ *to* $T$ *(see [16]).*

Theorem 8 shows that the study of the classes of $\leftrightarrow_{\mathbb{F}\mathrm{n}}$-equivalent flowcharts over $\mathbb{R}\mathrm{el}$ is of interest at least for the case when we have only one-input/one-output atoms. From the above results (Theorem 9.3 and Theorem 11.1 for $T = \mathbb{R}\mathrm{el}$), we get the following corollary:

**Corollary 12.** *The classes of* $\leftrightarrow_{\mathbb{F}\mathrm{n}}$*-equivalent flowchart schemes over* $\mathbb{R}\mathrm{el}$ *and* $A$ *form the structure freely generated by* $A$ *in the category of the* $a\delta$*-flows which are also* $d\delta$*-ssmc-ies with feedback .*

---

[4] This $a\alpha$-flow structure is equivalent to the "strong iteration theory" structure in [15].
[5] In [16], this was called "repetition theory".
[6] That is: "if $y_2 : p \to q$ and $y_1$ are abstract surjections and $f \in T(m, n+p)$ is such that $f(y_1 \oplus y_2) = g \oplus \top_q$, then $f = f(I_n \oplus \bot^p \top_p)$". It is related with the "zero-sum-free" condition on semirings.

This means, the axiomatisation is given by all the axioms in Table 1, except for S3-S4, and using $\mathrm{Func}_{\text{abstract functions}}$. By translating this result to the case of process graphs, we get an axiomatisation for the classes of bisimilar graphs analogous to the one given in [4]. Many axiomatisation results in process algebra use axioms implying that a guarded system of equations has an unique solution. Here we have used the functoriality rule. This rule gives another way to connect two equivalent systems, namely by changing the variables, e.g., we may delete certain variables (which are not used in the construction of the solution), or identify certain variables (providing that the terms that define them became equal after identification).

## References


1. J. Baeten and W. Weijland. *Process algebra.* Cambridge University Press, 1990.
2. J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77–121, 1985.
3. J.A. Bergstra, J.W. Klop, and E.R. Olderog. Readies and failures in the algebra of communicating processes. *SIAM Journal of Computing*, 17:1134–1177, 1988.
4. S.L. Bloom, Z. Esik, and D. Taubner. Iteration theory of synchronization trees. *Information and Computation*, 103:1–55, 1993.
5. B. Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science*, 25:95–169, 1983.
6. V.E. Căzănescu and Gh. Stefănescu. Towards a new algebraic foundation of flowchart scheme theory. *Fundamenta Informaticae*, 13:171–210, 1990.
7. V.E. Căzănescu and Gh. Stefănescu. Classes of finite relations as initial abstract data types I. *Discrete Mathematics*, 90:233–265, 1991.
8. V.E. Căzănescu and Gh. Stefănescu. A general result of abstarct flowchart schemes with applications to the study of accessibility, reduction and minimization. *Theoretical Computer Science*, 99:1–63, 1992. (Fundamental Study).
9. C.C. Elgot. Manadic computation and iterative algebraic theories. In *Proceedings Logic Colloquium'73*, pages 175–230. North-Holland, 1975. Studies in Logic and the Foundations of Mathematics, Volume 80.
10. C.C. Elgot. Finite automata from the a flowchart schemes point of view. In *Proceedings MFCS'77*. Springer, 1977. Lecture Notes in Computer Science.
11. C.C. Elgot. Some geometrical categories associated with flowchart schemes. In *Proceedings FCT'77*, pages 256–259. Springer, 1977. Lecture Notes in Computer Science.
12. C.C. Elgot, S.L. Bloom, and R. Tindell. On the algebraic structure of rooted trees. *Journal of Computer and System Sciences*, 16:362–399, 1978.
13. R. Milner. *A calculus of communicating systems.* Springer, 1980.
14. R. Milner. *Communication and concurrency.* Prentice Hall International, 1989.
15. Gh. Stefănescu. On flowchart theories I: The determinisitc case. *Journal of Computer and System Sciences*, 35:163–191, 1987.
16. Gh. Stefănescu. On flowchart theories II: The nondeterministic case. *Theoretical Computer Science*, 52:307–340, 1987.
17. Gh. Stefănescu. *Determinism and nondeterminism in program scheme theory: algebraic aspects.* PhD thesis, University of Bucharest, 1991.


This article was processed using the LaTeX macro package with LLNCS style