Processes with Multiple Entries and Exits Modulo Isomorphism and Modulo Bisimulation

J.A. Bergstra[1] Gh. Ştefănescu[2]

[1] *University of Amsterdam, Programming Research Group*
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
and
Utrecht University, Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

[2] *Institute of Mathematics of the Romanian Academy*
P.O. Box 1-764, 70700 Bucharest, Romania

E-mail: `janb@fwi.uva.nl` - `ghstef@imar.ro`

Abstract

This paper is an attempt to integrate the algebra of communicating processes (ACP) and the algebra of flownomials (AF). Basically, this means to combine axiomatized parallel and looping operators. To this end we introduce a model of process graphs with multiple entries and exits. In this model the usual operations of both algebras are defined, e.g. alternative composition (this covers both the sum of ACP and the disjoint sum of AF), sequential composition, feedback, parallel composition, left merge, communication merge, encapsulation, etc. [1] The main results consist of correct and complete axiomatisations of process graphs modulo isomorphism and modulo bisimulation.

Key words & Phrases: process algebra, feedback, flowchart theories.

1 Introduction

This paper is an attempt to integrate the algebra of communicating processes (ACP) and the algebra of flownomials (AF). Basically, this means to combine axiomatized parallel and looping operators.

There are three axiomatized looping operations that may be combined with the existing process algebra (ACP): Kleene's star "$*$" (repetition [Kle56]) used in regular algebra, Elgot's dagger "$\dagger$" (iteration [Elg75]) used in iteration theories, or up_arrow "$\uparrow$" (feedback [Ste90]) used in the algebra of flownomials. A study of process algebra with an iteration operation (originary Kleene's star) has already been presented in [BBP94]. Here we combine process algebra with the feedback operation.

[1] However, this fails to be a full model for algebra of flownomials due to the lack of the identities (= empty transitions). The study of the model with empty transitions is postpone for a later paper.

Our goal is to define a process algebra on which all operators of ACP [BeK84] are present as well as feedback, the key iteration construct of flowchart theories. To this end we combine the results of [BaB94], [BeS93] and various other results on flowchart theories [Gog74, Elg75, Ste87, Ste90, CaS90, BlE93]. Like in flowchart schemes [Ste90] feedback and alternative composition " $\diamond\!\!\!\!+$ " suffice to express all finite state systems. (" $\diamond\!\!\!\!+$ " is a mixture of disjoint sum "$\oplus$", left-composition with converses of functions and right–composition with functions.) In fact alternative composition and feedback allow the construction of normal forms modulo isomorphism of transition systems. (A similar result on undirected networks is presented in [Par93], an approach that simplifies the earlier algebra of flow graphs of Milner [Mil79].)

Technically we depart from the graph isomorphism model for the operators of ACP that was outlined in [BaB94]. This model is adapted to allow for process graphs with multiple entries and exits. Then the feedback operator is introduced. For technical reasons renaming operators for entries and exits are needed. We notice that entries and exits are just a particular kind of states.

In the process of the design of this model we have taken several decisions which we want to make clear from the very beginning.

- Our transition systems allow for multiple entries and multiple exits. This feature drastically increases the expressive power of the algebra. All finite state processes are represented by closed terms built up from atomic actions and some constants by using two operations only, i.e., alternative composition and feedback. Moreover, almost all process graphs are represented. To be precise: (1) in the case witout ϵ (i.e., without empty transitions) all process graphs with no incoming edges into the entries, no outgoing edges from the exits and such that no entry is an exit are represented; (2) if ϵ is allowed, then all process graphs are represented.

- We have decided to make the operations total by providing a default system of working in the case the types do not agree. E.g., seqential composition is defined even the outputs of the first process graph do not match the inputs of the second one.

- There are two options regarding the using of port names: (1) to use an arbitrary set of port names and renamings or (2) to order the port names —hence, to use the first n natural numbers as names for a process with n ports— and an explicit algebra to model the renamings. We have decided to use here the first variant which gives more freedom (i.e., it is more abstract) and perhaps easier to understand. The second version may be more suited for implementations.

- Finally, we had to decide whether we allow or not for ϵ steps. We start here with the model without ϵ steps which seems easier. The loss of expressivity is small in the case of cyclic processes (as we already mention), but important in the case of acyclic ones (not all acyclic processes may be represented by using alternative and sequential composition). This ϵ-free case also generates some complications in defining bisimilarity which require an explicit splitting operator.

In this paper we study the ϵ-free model. The main results are:

- Expresiveness (already mentioned);
- Correct and complete axiomatisation of process graphs modulo isomorphism;
- Correct and complete axiomatisation of process graphs modulo bisimulation.

2 Process graphs modulo isomorphism

2.1 Process graphs

Notations:

– A denotes a set of atomic actions with typical elements a, b, c, d, e.

– V denotes a set of port names with typical elements p, q, r, s, t, u, v. We assume V is closed to cartesian product. (That is, we assume an injective coding $\times\!\!\!\!<: V \times V \to V$ is given and $A \times B = \{a \times\!\!\!\!< b \mid a \in A,\ b \in B\}$.)

– "$\circ$" denotes relational composition of two binary relations (i.e., $R \circ T = \{(x, z) \mid \exists y : x\ R\ y \text{ and } y\ T\ z\}$.

– Id_A denotes the identity relation on a set A (i.e., $Id_A = \{(x, x) \mid x \in A\}$).

– $\phi|_A$ denotes the restriction of a function ϕ to a subset A of the definition domain.

– $[n] = \{1, \ldots, n\}$

– We write $f \ \cup\ g$ for the union of two functions; if the definition domains are disjoint, the union is a function, too.

Definition 1 (process graphs with multiple entries and multiple exits) A *process graph* P is given by the following data: $P = (\partial_i, E, l, \partial_o) : I \xrightarrow{S} O$ where:

- I, S, O are finite subsets of V and $S \cap (I \cup O) = \emptyset$
- E is a finite set
- $\partial_i, \partial_o, \ell$ are functions: $\quad \partial_i : E \to I \cup S, \quad \partial_o : E \to O \cup S, \quad \ell : E \to A$

$G(A)(I, S, O)$ denotes the process graphs with entries in I, exits in O, internal vertices in S and labels in A. The class of process graphs with I and O given and arbitrary S (resp. with arbitrary $I,\ S,\ O$) is denoted by $G(A)(I, O)$ (resp. by $G(A)$). $\square$

The meaning of these data is the following:

- I specifies the start vertices, S the internal ones and O the end ones.

- E specifies the edges (transitions). $\partial_i(e)$, $\partial_o(e)$ and $\ell(e)$ give the source vertex, the target vertex and the label of an edge e, respectively.

Note that $G(A)$ includes all process graphs obeying the following restriction: "a start vertex has no incoming edges, an end vertex has no outgoing edges and a start vertex is not an end vertex".

A process graph $P = (\partial_i, E, l, \partial_o) : I \xrightarrow{S} O$ may also be specified by using the transition relation $T \subseteq (I \cup S) \times A \times (O \cup S)$ consisting of all the transitions $\partial_i(e) \xrightarrow{\ell(e)} \partial_o(e)$, for $e \in E$.

There are two possibilities here: (1) to take multiplicity into account, i.e., to allow for multiple edges with the same label between two vertices and to work with multisets or (2) to avoid such a possibility and to work in the classical set theory. Below we start with the first case[2] .

Sometimes we write these data as $T : I \xrightarrow{S} O$ or as a $(I \cup S) \times (O \cup S)$-matrix $\begin{pmatrix} f_{io} & f_{is} \\ f_{so} & f_{ss} \end{pmatrix}$ with entries multisets of elements in A.

Definition 2 Let $P : I \xrightarrow{S} O$ and $P' : I \xrightarrow{S'} O$ be two process graphs and T and T' their transition relations. A bijection $\phi : S \to S'$ is called an *isomorphism* if

$$(Id_I \cup \phi) \circ T' = T \circ (Id_O \cup \phi)$$

($\circ$ is relational composition) $\Box$

2.2 Operations on process graphs

The operations will be defined on process graphs modulo isomorphism.

Definition 3 (operations and constants on process graphs)

- **Constants:**
  - **Atomic actions:** a^p_q, for $a \in A,\ p, q \in V$
    $a^p_q = \langle \partial_i, \{\star\}, \ell, \partial_o \rangle : \{p\} \xrightarrow{\emptyset} \{q\}$, where $\partial_i(\star) = p,\ \ell(\star) = a,\ \partial_o(\star) = q$.
  - **Empty processes:** $\emptyset$, $\perp^p$, $\top_p$
    $\emptyset = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle : \emptyset \xrightarrow{\emptyset} \emptyset, \quad \perp^p = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle : \{p\} \xrightarrow{\emptyset} \emptyset,$
    $\top_q = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle : \emptyset \xrightarrow{\emptyset} \{q\}$

[2] Formally, the relationship between T and $\partial_i, \partial_o, \ell$ may be stated as follows: We work with matrices having multisets of elements in A as entries. Take the $E \times E$-matrix L with all entries equal to the empty set, except for the diagonal elements that are $\{\ell(e)\}$, for each $e \in E$. Then $T = \partial_i^{-1} \circ L \circ \partial_o$.

By using the operations on process graphs of this paper T may also be defined as follows: Take a proper port name r_e for each edge $e \in E$. Denote by ∂'_i the renaming corresponding to ∂_i, i.e. $\partial'_i(r_e) = \partial_i(e)$ for e in E, otherwise identity. Similarly for ∂'_o. Now T corresponds to
$\partial'_i \triangleleft [(\Diamond_{p \in I'} \perp^p) \Diamond (\Diamond_{q \in O'} \top_q) \Diamond (\Diamond_{r \in S'} \delta^r_r) \Diamond (\Diamond_{e \in E} \ell(e)^{r_e}_{r_e})] \triangleright \partial'_o$,
where I', O' and S' are the vertices without both incoming or outgoing vertices.

- **Operations:**
  - **Alternative composition:** $\oplus$

    Let $P = \langle \partial_i, E, \ell, \partial_o \rangle : I \xrightarrow{S} O$ and $P' = \langle \partial_i', E', \ell', \partial_o' \rangle : I' \xrightarrow{S'} O'$ be given. Assume $S \cap (I' \cup S' \cup O') = \emptyset$, $S' \cap (I \cup S \cup O) = \emptyset$ and $E \cap E' = \emptyset$. Then

    $$P \oplus Q = \langle \partial_i \cup \partial_i',\ E \cup E',\ \ell \cup \ell',\ \partial_o \cup \partial_o' \rangle : I \cup I' \xrightarrow{S \cup S'} O \cup O'$$

    Effect: Disjoint union, but the entries (resp. the exits) with the same port name are identified.
  - **Feedback:** $\uparrow_q^p$

    Let $P = \langle \partial_i, E, \ell, \partial_o \rangle : I \xrightarrow{S} O$ be given.
    * If $p \notin I$ and $q \notin O$, then $P \uparrow_q^p = P$.
    * Otherwise, take a fresh r (not in $I \cup O \cup S \cup \{p, q\}$) and define

      $$P \uparrow_q^p = \langle \partial_i \circ (Id_{(I-\{p\}) \cup S} \cup t_{\{p\} \cap I \mapsto \{r\}}),\ E,\ \ell,\ \partial_o \circ (Id_{(O-\{q\}) \cup S} \cup t_{\{q\} \cap O \mapsto \{r\}}) \rangle$$

      $$: I - \{p\} \xrightarrow{S \cup \{r\}} O - \{q\}$$

      where $t_{A \mapsto \{r\}}$ is the unique function from A to $\{r\}$, for a set A with at most one element.

    Effect: If there are an entry with label p and an exit with label q, then they are identified as a unique internal vertex r. If q does not appear as an exit and p appers as an entry, then the effect is that we hide enter p that becomes an internal vertex. Similarly if p does not appear as an entry and q appear as an exit. Finally, if p and q does not apper at all, then the result of the feedback is P itself.
  - **Renaming port names:** $\triangleleft$, $\triangleright$

    Let $\phi : V \to V$ be a renaming function. Assume $P = \langle \partial_i, E, \ell, \partial_o \rangle : I \xrightarrow{S} O$ is such that $S \cap (\phi(I) \cup \phi(O)) = \emptyset$. Then:

    $$\phi \triangleleft P = \langle \partial_i \circ (\phi|_I \cup Id_S),\ E,\ \ell,\ \partial_o \rangle : \phi(I) \xrightarrow{S} O$$

    and

    $$P \triangleright \phi = \langle \partial_i,\ E,\ \ell,\ \partial_o \circ (\phi|_O \cup Id_S) \rangle : I \xrightarrow{S} \phi(O)$$

    Effect: Rename the entry (resp. exit) ports by using ϕ and identify the entries (resp. the exits) that get equal renamings.

Notation:

(1) For $I = \{p_1, \ldots, p_m\}$ and $O = \{q_1, \ldots, q_n\}$ we define
$\delta_O^I := \perp^{p_1} \oplus \ldots \oplus \perp^{p_m} \oplus \top_{q_1} \oplus \ldots \oplus \top_{q_n}$; δ_q^p means $\delta_{\{q\}}^{\{p\}}$.

(2) It is meaningful (up to an isomorphism) to introduce the constant $I(1) := \delta_r^r \uparrow_r^r$, for an $r \in V$ and $I(m) = \oplus_{j=1}^{m} I(1)$ for an $m \in \mathbb{N}$. Up to an isomorphism $I(m)$ is $\langle \emptyset, \emptyset, \emptyset, \emptyset \rangle : \emptyset \xrightarrow{S} \emptyset$, for an S with m elements.

(3) Let $A, B \subset V$ be such that $A \cap B = \{p_1, \ldots, p_k\}$, $A - B = \{q_1, \ldots, q_m\}$ and $B - A = \{r_1, \ldots, r_n\}$. Then $P \uparrow^A_B$ means $P \uparrow^{p_1}_{p_1} \ldots \uparrow^{p_k}_{p_k} \uparrow^{q_1}_{q'_1} \ldots \uparrow^{q_m}_{q'_m} \uparrow^{r'_1}_{r_1} \ldots \uparrow^{r'_n}_{r_n}$ where $q'_1, \ldots, q'_m, r'_1, \ldots, r'_n$ are fresh variables. (Note that this convention works on graphs modulo isomorphism. Apparently the definition of $\uparrow^A_B$ depends on the order of feedbacks or on the names of the fresh variables, but this freedom do not change the isomorphism class.)

(4) Let $P = \langle \partial_i, E, \ell, \partial_o \rangle : I \xrightarrow{S} O$ and $E' \subseteq E$ be given. We denote by $P|_{E'}$ the restriction

$$P|_{E'} = \langle \partial_i|_{E'},\ E',\ \ell|_{E'},\ \partial_o|_{E'} \rangle : I \xrightarrow{S} O$$

- **Operations (continuation):**
  - **Sequential composition:** $\odot$

    Let $P : I \to O$ and $P' : I' \to O'$ be given. Take a bijective renaming ϕ such that $\phi(O \cup I')$ contains fresh names only. Then define:

    $$P \odot P' = [(P \triangleright \phi) \ \text{\textdiamond}\ (\phi \triangleleft P')] \uparrow^{\phi(I')}_{\phi(O)}$$

    Effect: Sequential composition. Note that we hide all the exits of P and all the entries of Q that have no correspondent.
  - **Parallel composition:** $||$
    - First, we define $||$ for processes $P : I \to O$ with $I \cap O = \emptyset$. Let $P = \langle \partial_i, E, \ell, \partial_o \rangle : I \xrightarrow{S} O$ and $P' = \langle \partial'_i, E', \ell', \partial'_o \rangle : I' \xrightarrow{S'} O'$ be two such processes. Define

      $$P \mid\mid P' = \quad \langle \partial^1_i, E^1, \ell^1, \partial^1_o \rangle \quad : I \times I' \overset{(I \cup S \cup O) \times (I' \cup S' \cup O') - (I \times I' \cup O \times O')}{\longrightarrow} O \times O'$$

      where if pr_1 and pr_2 denote the 1st and the 2nd projection, respectively, then

      $$\partial^1_i = \partial_i \times \partial'_i \ \cup\ \partial_i \times Id_{I' \cup S' \cup O'} \ \cup\ Id_{I \cup S \cup O} \times \partial'_i,$$

      $$E^1 = E \times E' \ \cup\ E \times (I' \cup S' \cup O') \ \cup\ (I \cup S \cup O) \times E',$$

      $$\ell^1 = \ell \times \ell' \ \cup\ pr_1 \circ \ell \ \cup\ pr_2 \circ \ell',$$

      $$\partial^1_o = \partial_o \times \partial'_o \ \cup\ \partial_o \times Id_{I' \cup S' \cup O'} \ \cup\ Id_{I \cup S \cup O} \times \partial'_o$$
    - In general, if $P : I \xrightarrow{S} O$ and $P' : I' \xrightarrow{S'} O'$ are two arbitrary graphs, then take two bijective renaming functions ϕ and ψ such that $\phi(O) \cap (I \cup S) = \emptyset$ and $\psi(O') \cap (I' \cup S') = \emptyset$. Define

      $$P \mid\mid P' = [(P \triangleright \phi) \mid\mid (P' \triangleright \psi)] \triangleright (\phi^{-1} \times \psi^{-1})$$

    Note: In general, $P \mid\mid P'$ is $\langle \partial^2_i, E^1, \ell^1, \partial^2_o \rangle$, where E^1 and ℓ^1 as above and ∂^2_i (resp. ∂^2_o) obtained by composing ∂^1_i (resp. ∂^1_o) with appropriate bijective renamings.

    Effect: Each time $P \mid\mid P'$ performs a transition from P, or a transition from P', or a pair of transitions (one from P and one from P'), if possible.

– **Left merge:** $⌊⌊$
Let P, P' and $P \mid\mid P' = \langle \partial_i^2, E^1, \ell^1, \partial_o^2 \rangle$ be as above and $E_l = \{e \in E_i^1 \mid \partial_i^2(e) \in I \times I'$ and $pr_2(\partial_i^2(e)) \neq pr_2(\partial_o^2(e))\}$. Then

$$P ⌊⌊ P' = (P \mid\mid P')|_{E^1 - E_l}$$

Effect: As in $P \mid\mid P'$, but the outgoing transitions from an initial state are only from P.

– **Communication merge:** $\mid$
As above, but take $E_c = \{e \in E^1 \mid \partial_i^2(e) \in I \times I'$ and $[pr_1(\partial_i^2(e)) = pr_1(\partial_o^2(e))$ or $pr_2(\partial_i^2(e)) = pr_2(\partial_o^2(e))]\}$. Then

$$P \mid P' = (P \mid\mid P')|_{E^1 - E_c}$$

Effect: As in $P \mid\mid P'$, but the outgoing transitions from an initial state are pairs of transitions (one from P and one from P'.

– **Encapsulation:** ∂_H
Let a subset $H \subseteq A$ and a process $P = \langle \partial_i, E, \ell, \partial_o \rangle : I \xrightarrow{S} O$ be given. Let $E_{A-H} = \{e \in E | \ell(e) \notin A\}$. Then

$$\partial_H(P) = \langle \partial_i|_{E_{A-H}},\ E_{A-H},\ \ell|_{E_{A-H}},\ \partial_o|_{E_{A-H}} \rangle : I \xrightarrow{S} O$$

Effect: All the transitions in H are deleted.

□

2.3 Expressiveness

Theorem 4 (expressiveness)

1) All graphs in $G(A)$ may be constructed starting with atomic actions or constants $\emptyset$, $\perp$, $\top$ and using alternative composition and feedback only.

2) There are acyclic processes that cannot be constructed starting with atomic actions and constants and using alternative and sequential composition only.

Proof: 1) Let $P : I \xrightarrow{S} O$ be given. Let $(a_1)_{q_1}^{p_1}, \ldots, (a_k)_{q_k}^{p_k}$ be all the edges of P with the corresponding input and output vertex name. Finally, let $I' \subseteq I$, $O' \subseteq O$ and $S' \subseteq S$ be the sets of all the vertices in P without both incoming or outgoing edges. Assume $S = \{r_1, \ldots, r_n\}$. Then

$$(*) \quad P = [(\ \oplus_{p \in I'} \perp^p) \oplus (\ \oplus_{q \in O'} \top_q) \oplus (\ \oplus_{r \in S'} \delta_r^r) \oplus (a_1)_{q_1}^{p_1} \oplus \ \ldots \ \oplus (a_k)_{q_k}^{p_k}] \uparrow_{r_1}^{r_1} \cdots \uparrow_{r_n}^{r_n}$$

2) The acyclic process "H" defined below cannot be represented with atomic actions and constants and using alternative and sequential composition only. "H" : $\{p_1, p_2\} \xrightarrow{\{u,v\}} \{q_1, q_2\}$ is the process graph given by five transitions $p_1 \xrightarrow{a} u$, $u \xrightarrow{a} q_1$, $p_2 \xrightarrow{a} v$, $v \xrightarrow{a} q_2$ and

$u \xrightarrow{a} v$. First, it is clear that "H" cannot be written as a sum $P \oplus Q$ with both P and Q nonempty. Suppose "H" is written as a composite $P \odot Q$. Then u cannot be an internal vertex in P. (If so, then q_1 would also be a vertex of P and could not be an exit of the composite $P \odot Q$; contradiction.) Hence $u \in o(P)$. It follows that transition $u \xrightarrow{a} v$ cannot be in P. Similarly, transition $u \xrightarrow{a} v$ cannot be in Q. From this impossibility we infer that "H" cannot be written as a composite of two process graphs and consequently cannot be represented in this context. □

2.4 Axioms, correctness, completeness

Let $Exp(Act, Const;\ \oplus, \uparrow, \lhd, \rhd)$ be the set of expressions constructed from the atomic actions and constants using alternative composition, feedback and renamings. Define three functions $i, o : Exp \to V$ and $s : Exp \to \mathbb{N}$ giving the sets of port names for the entries and for the exits and the number of the internal vertices, respectively. Formally they are defined as follows:

- $i(a^p_q) = \{p\},\ o(a^p_q) = \{q\},\ s(a^p_q) = 0;$
- $i(\perp^p) = \{p\},\ o(\perp^p) = \emptyset,\ s(\perp^p) = 0;$
- $i(\top_p) = \emptyset,\ o(\top_p) = \{p\},\ s(\top_p) = 0;$
- $i(\emptyset) = o(\emptyset) = \emptyset,\ s(\emptyset) = 0;$
- $i(P \oplus Q) = i(P) \cup i(Q),\ o(P \oplus Q) = o(P) \cup o(Q),\ s(P \oplus Q) = s(P) \cup s(Q);$
- $i(P \uparrow^p_q) = i(P) - \{p\},\ o(P \uparrow^p_q) = o(P) - \{q\},\ s(P \uparrow^p_q) =$ "if $p \notin i(P)$ and $q \notin o(P)$ then $s(P)$ else $s(P) + 1$";
- $i(P \rhd \phi) = i(P),\ o(P \rhd \phi) = \phi(o(P)),\ s(P \rhd \phi) = s(P);$
- $i(\phi \lhd P) = \phi(i(P)),\ o(\phi \lhd P) = o(P),\ s(\phi \lhd P) = s(P).$

TABLE 1. Axioms for graph isomorphism

Notation: $[q/p](x) =$ "if $x = p$ then q else x";
$\phi_{|A|}(x) =$ "if $x \in A$ then x else $\phi(x)$"

$A1.$	$f \oplus (g \oplus h) = (f \oplus g) \oplus h$	
$A2.$	$f \oplus g = g \oplus f$	
$A3.$	$f \oplus \emptyset = f$	
$A4.$	$f \oplus \perp^p = f$	if $p \in i(f)$
$A5.$	$f \oplus \top_q = f$	if $q \in o(f)$
$A(**)$	$f \oplus f = f$	
$F1.$	$f \upharpoonright^p_q \upharpoonright^r_s = f \upharpoonright^r_s \upharpoonright^p_q$	if $p \neq r$ and $q \neq s$
$F2.$	$f \upharpoonright^p_q = f$	if $p \notin i(f)$ and $q \notin o(f)$
$FA1.$	$f \oplus (g \upharpoonright^p_q) = (f \oplus g) \upharpoonright^p_q$	if $p \notin i(f)$ and $q \notin o(f)$
$FA2.$	$(f \oplus \perp^p) \upharpoonright^p_q = f \upharpoonright^p_q$	if $q \in o(f)$
$FA3.$	$(f \oplus \top_q) \upharpoonright^p_q = f \upharpoonright^p_q$	if $p \in i(f)$

$R1_o.$	$a^p_q \triangleright \phi = a^p_{\phi(q)}$	$R1^i.$	$\phi \triangleleft a^p_q = a^{\phi(p)}_q$
$R2_o.$	$\perp^p \triangleright \phi = \perp^p$	$R2^i.$	$\phi \triangleleft \perp^p = \perp^{\phi(p)}$
$R3_o.$	$\top_q \triangleright \phi = \top_{\phi(q)}$	$R3^i.$	$\phi \triangleleft \top_q = \top_q$
$R4_o.$	$\emptyset \triangleright \phi = \emptyset$	$R4^i.$	$\phi \triangleleft \emptyset = \emptyset$
$RA_o.$	$(f \oplus g) \triangleright \phi = (f \triangleright \phi) \oplus (g \triangleright \phi)$	$RA^i.$	$\phi \triangleleft (f \oplus g) = (\phi \triangleleft f) \oplus (\phi \triangleleft g)$
$RF1_o.$	$(f \upharpoonright^p_q) \triangleright \phi = (f \triangleright \phi_{\lvert\{q\}\rvert}) \upharpoonright^p_q$	$RF1^i.$	$\phi \triangleleft (f \upharpoonright^p_q) = (\phi_{\lvert\{p\}\rvert} \triangleleft f) \upharpoonright^p_q$
	if $q \notin \phi(o(f) - \{q\})$		if $p \notin \phi(i(f) - \{p\})$
$RF2_o.$	$f \upharpoonright^p_q = (f \triangleright [s/q]) \upharpoonright^p_s$ if $s \notin o(f)$	$RF2^i.$	$f \upharpoonright^p_q = ([r/p] \triangleleft f) \upharpoonright^r_q$ if $r \notin i(f)$

$R^i_o.$	$\phi \triangleleft (f \triangleright \psi) = (\phi \triangleleft f) \triangleright \psi$

We say an expression is in a *prenormal form (pnf)* if it is of the following type

$$(f_1 \oplus \ldots \oplus f_n) \uparrow^{p_1}_{q_1} \ldots \uparrow^{p_k}_{q_k} \quad \text{with } f_i \text{ of type } \perp^p,\ \top_p \text{ or } a^p_q, \text{ for all } i$$

Lemma 5 *Every expression may be brought to a pnf by using the axioms in Table 1.*

Proof: The constants are in pnf. For operations, it is enough to show that the result of an operation is a pnf expression whenever the arguments are pnf expressions.

- For $\uparrow$ it is clear: if E is in a pnf, then $E \uparrow^p_q$ is in a pnf, too.
- For $\triangleright$:

  We prove on induction on k that if E is a pnf with k feedbacks, then $E \triangleright \phi$ may be brought to a pnf.

  For $k = 0$ it is obvious.

  Suppose it holds for all $j < k$. Assume $E = (f_1 \oplus \ldots \oplus f_n) \uparrow^{p_1}_{q_1} \ldots \uparrow^{p_k}_{q_k}$ is a given pnf with k feedbacks. Take a fresh q'_k (not in $\phi(o(E) \cup \{q_1, \ldots, q_{k-1}\}) \cup o(E) \cup \{q_1, \ldots, q_{k-1}\}$). Then by $RF2_o$ and $RF1_o$,

  $$\begin{aligned} E \triangleright \phi &= (f_1 \oplus \ldots \oplus f_n) \uparrow^{p_1}_{q_1} \ldots \uparrow^{p_k}_{q_k} \triangleright \phi \\ &= (((f_1 \oplus \ldots \oplus f_n) \uparrow^{p_1}_{q_1} \ldots \uparrow^{p_{k-1}}_{q_{k-1}}) \triangleright [q'_k/q_k]) \uparrow^{p_k}_{q'_k} \triangleright \phi \\ &= ((f_1 \oplus \ldots \oplus f_n) \uparrow^{p_1}_{q_1} \ldots \uparrow^{p_{k-1}}_{q_{k-1}} \triangleright [q'_k/q_k] \triangleright \phi_{|\{q'_k\}|}) \uparrow^{p_k}_{q'_k} \end{aligned}$$

  and the statement follows by applying the inductive hypothesis.
- Similarly for $\triangleleft$.

Consequently, all renamings may be eliminated by shifting them to act on constants and atomic actions.

- For $\oplus$:

  Assume E and E' are in pnf, say $E = (f_1 \oplus \ldots \oplus f_n) \uparrow^{p_1}_{q_1} \ldots \uparrow^{p_k}_{q_k}$ and $E' = (f'_1 \oplus \ldots \oplus f'_{n'}) \uparrow^{p'_1}_{q'_1} \ldots \uparrow^{p'_{k'}}_{q'_{k'}}$. By using axiom RF2 we may suppose that $p'_1, \ldots, p'_{k'} \notin i(E)$, $q'_1, \ldots, q'_{k'} \notin o(E)$ and $p_1, \ldots, p_k \notin i(E') \cup \{p'_1, \ldots, p'_{k'}\}$, $q_1, \ldots, q_k \notin o(E') \cup \{q'_1, \ldots, q'_{k'}\}$. Then by FA1 and A2

  $$\begin{aligned} E \oplus E' &= (E \oplus f'_1 \oplus \ldots \oplus f'_{n'}) \uparrow^{p'_1}_{q'_1} \ldots \uparrow^{p'_{k'}}_{q'_{k'}} \\ &= (f_1 \oplus \ldots \oplus f_n \oplus f'_1 \oplus \ldots \oplus f'_{n'}) \uparrow^{p_1}_{q_1} \ldots \uparrow^{p_k}_{q_k} \uparrow^{p'_1}_{q'_1} \ldots \uparrow^{p'_{k'}}_{q'_{k'}} \end{aligned}$$

Hence every expression may be brought to a pnf. $\Box$

Next we define normal form expressions, but first a few notation.

$$\perp^I := \ \Diamond_{p\in I}\perp^p, \quad \top_O := \ \Diamond_{q\in O}\top_q, \quad I(m) := \ \Diamond_{j\in[m]}(\delta_r^r)\uparrow_r^r$$

We say an expression E is in a *normal form (nf)* if it is of the following type

$$(\alpha \Diamond \beta)\uparrow_{p_1}^{p_1}\ldots\uparrow_{p_k}^{p_k}$$

where

(1) $\alpha = \perp^{I'} \Diamond \top_{O'} \Diamond I(m)$, for some $I' \subseteq i(E),\ O' \subseteq o(E)$ and $m \leq s(E)$;

(2) $\beta = \ \Diamond_{i=1}^{n} a_{s_i}^{r_i}$;

(3) $i(\alpha)\ \cap\ i(\beta) = \emptyset, \quad o(\alpha)\ \cap\ o(\beta) = \emptyset$;

(4) there are no unused feedbacks, i.e.

- $p_1, \ldots, p_k$ are all distinct
- for each feedback $\uparrow_{p_i}^{p_i},\ i \in [n]$, there exists either a term $a_q^{p_i}$ or a term $a_{p_i}^r$ in β

Lemma 6 *Every expression E may be brought to a nf by using the axioms in Table 1.*

Proof: By the above lemma we may suppose E is in a pnf. The sum may contain several terms $\perp^p$ that may be eliminated by using the following rule:

> Eliminate a term $f_i = \perp^p$ if:
>
> (1) there exists another term $f_j = \perp^p$ $(j \neq i)$ or $f_j = a_q^p$ (this elimination may be done by axiom A4)
>
> or
>
> (2) there exists a term $f_j = a_s^r$ or $f_j = \top_s$ in the sum and a fedback $\uparrow_s^p$ (this elimination may be done by axiom AF2)

Similarly for $\top$. Repeat these reductions as long as possible. Finally we get a pnf such that:

- each term $\perp^p$ (resp. $\top_q$) that is not under a feedback has its own name;
- a term $\perp^p$ (resp. $\top_q$) that is under a feedback $\uparrow_q^p$ is such that q (resp. p) is fictitious.

The next step is to move all the feedback that apply to a $\perp^p$ or to a $\top_p$ term near these terms and then to replace the resulting terms by $(\delta_p^p)\uparrow_p^p$ — i.e., by $I(1)$ — (for esthetical reasons).

Finally we get a nf by eliminating the unused feedbacks (making use of axiom F2) and replacing each feedback $\uparrow_q^p$ by a $\uparrow_r^r$ for a fresh name r (making use of RF2 and F1). □

Lemma 7 (uniqueness of the nf)

(1) The nf associated to a process graph has a unique empty part α, a unique transition part β and a unique feedback part up to commutations of terms, permutations of feedbacks and bijective renamings of feedback ports.

(2) Two nf that represent the same graph may be proved equivalent via the axioms.

Proof: All the components are uniquely determinated by the process graph as follows:

- $\perp^{I'}$ uniquely corresponds to the entries without outgoing edges;
- $\top_{O'}$ corresponds to the exits without incoming edges;
- m of $I(m)$ is the number of the internal vertices without both incoming or outgoing edges;
- k is the number of the internal vertices having at least an incoming or an outgoing edge;
- β corresponds to the edges of the graph.

By axioms A2, F1 and RF2 we may pass from a nf to another arbitrary one that represent the same graph. □

Theorem 8 *The axioms in Table 1, except for A(**), are correct and complete for graph isomorphism with multiplicity.*

Proof: Correctness: Obvious. Completeness: By the above lemmas. □

In the case without multiplicity by definition a nf is one which also satisfies the condition:

(5) β does not contains two equal terms

Now, it is clear that

Theorem 9 *The axioms in Table 1 are correct and complete for graph isomorphism without multiplicity.* □

3 Process graphs modulo bisimulation

3.1 Simulation and bisimulation

Simulation is a standard notion of graph homomorphism that has been used in the study of flow diagram programs (see, e.g. [Gog74, Elg77, Ste87]). Bisimulation is an equivalence on transition systems introduced by Park [Pa80] in connection with Milner's work on concurrency [Mil80]. In [BeS93] we have shown that bisimulation is the equivalence relation generated by simulation via functions by using a translation between flowchart schemes and process graphs. In [BeS94] it is given a simple proof of this fact using transition systems and simple rules of the calculus with relations, only.

Definition 10 (simulation and bisimulation) Assume two process graphs $P : I \xrightarrow{S} O$ and $P' : I \xrightarrow{S'} O$ are given. Let T and T' be the associated transition relations.

We say P and P' are *similar* via a function $\phi : S \to S'$ and write $P \to_\phi P'$ if

- $(Id_I \cup \phi) \circ T' = T \circ (Id_O \cup \phi)$

We say P and P' are *bisimilar* via a relation $\rho \subseteq S \times S'$ and write $P \leftrightarrow_\rho P'$ if

- $(Id_I \cup \rho) \circ T' \subseteq T \circ (Id_O \cup \rho)$
- $(Id_I \cup \rho^{-1}) \circ T \subseteq T' \circ (Id_O \cup \rho^{-1})$

□

In [BeS94] we have proved that:

Theorem 11 *Bisimulation is the equivalence relation generated by simulation via functions.*
□

Simulation is an useful tool to speak about minimization. Simulation via surjective functions models identification of the states with the same behaviour and the *converse* of the simulation via injective functions models the deletion of the nonaccessible parts.

3.2 Axioms, correctness, completeness

Theorem 11 suggest that in order to axiomatize bisimilar processes one has to add to the axioms corresponding to graph isomorphism a new axiom corresponding to simulation via functions, i.e., for a function $\phi : S \to S'$

$$T \circ (Id_O \ \cup \ \phi) = (Id_O \ \cup \ \phi) \circ T' \quad \Rightarrow \quad T \uparrow_S^S = T' \uparrow_{S'}^{S'}$$

But which is the meaning of the composition here? Clearly, $T \circ (Id_O \ \cup \ \phi)$ may be replaced by $T \triangleright (Id_O \ \cup \ \phi)$, but what does $(Id_O \ \cup \ \phi) \circ T'$ mean? Our model is ϵ-free here and such a construction has no meaning. So we have to define a new operation

$$\psi \ \triangleright P$$

which has to simulate the effect of the composition on bisimilar processes.

The intuitive meaning of this operation is that we have multiple incoming ϵ-edges to an entry and it is clear what we have to do: split the entry together with its outgoing transitions in order to have a copy for each incoming edge. This way left composition with functions may be defined on process graphs modulo bisimulation.

- **Operations (continuation):**

  Splitting input ports: $\triangleright$

  Let $\phi : V \to V$ be a function such that $\phi^{-1}(x)$ is finite for all x. Let $P = \langle \partial_i, E, \ell, \partial_o \rangle : I \xrightarrow{S} O$ be such that $S \cap \phi^{-1}(I) = \emptyset$. Suppose $I = \{p_1, \ldots, p_n\}$ and $I_i = \{\phi^{-1}(p_i)\}$, for $i \in [n]$. Let $E_i = \{e \in E | \ \partial_i(e) = p_i\}$, for $i \in [n]$ and $E' = E - \bigcup_{i \in [n]} E_i$. Then

  $$\phi \triangleright P = \langle \bigcup_{i=1}^n pr_2^i \ \cup \ \partial_i|_{E'}, \ \bigcup_{i=1}^n E_i \times I_i \ \cup \ E', \ \bigcup_{i=1}^n pr_1^i \circ \ell \ \cup \ \ell|_{E'},$$

  $$\bigcup_{i=1}^n pr_1^i \circ \partial_o \ \cup \ \partial_o|_{E'} \rangle : \phi^{-1}(I) \xrightarrow{S} O$$

  where for $i \in [n]$, pr_1^i and pr_2^i denote the 1st and the 2nd projection of $E_i \times I_i$, respectively.

  Effect: In the resulting graph there is a copy of an entry p and of its outgoing edges for each $x \in \phi^{-1}(p)$. In particular, if $\phi^{-1}(p) = \emptyset$ the effect is that p and its outgoing edges are deleted.

  Note: This splitting operation may be easier defined by matrices as follows: If T is the matrix associated to P, then the matrix associated to $\phi \triangleright P$ is just the matrix composition $(\phi|_{\phi^{-1}(I)} \ \cup \ Id_S) \circ T$.

The axioms for this new operation are given in Table 2. Before listing them we make a comment related to axiom SR^i: Let $\phi, \ \psi : V \to V$ be two functions such that $\phi^{-1}(x)$ and $\psi^{-1}(x)$ are finite for all x. For $r \in V$ denote by pr_1^r and pr_2^r the 1st and the 2nd projection of $\phi^{-1}(r) \times \psi^{-1}(r)$, respectively. Finally, take $\psi' = \bigcup_{r \in V} pr_1^r$ and $\phi' = \bigcup_{r \in V} pr_2^r$. These functions fulfil $\phi \circ \psi^{-1} = \psi'^{-1} \circ \phi'$.

TABLE 2. Axioms for splitting

$S1.$	$\phi \triangleright a_q^p$	$=$	if $\phi^{-1}(p) \neq \emptyset$ then $\Large\diamond_{r \in \phi^{-1}(p)} a_q^r$ else $\top_q$
$S2.$	$\phi \triangleright \perp^p$	$=$	if $\phi^{-1}(p) \neq \emptyset$ then $\Large\diamond_{r \in \phi^{-1}(p)} \perp^r$ else $\emptyset$
$S3.$	$\phi \triangleright \emptyset$	$=$	$\emptyset$
$S4.$	$\phi \triangleright \top_q$	$=$	$\top_q$
$SA.$	$\phi \triangleright (f \diamond g)$	$=$	$(\phi \triangleright f) \diamond (\phi \triangleright g)$
$SF.$	$\phi \triangleright (f \uparrow_q^p)$	$=$	$(\phi_{\vert\{p\} \cup \phi^{-1}(p)\vert} \triangleright f) \uparrow_q^p$
$SR^i.$	$\phi \triangleright (\psi \triangleleft f)$	$=$	$\psi' \triangleleft (\phi' \triangleright f)$ where ϕ', ψ' are functions such that $\phi \circ \psi^{-1} = \psi'^{-1} \circ \phi'$
$SR_o.$	$\phi \triangleright (f \triangleright \psi)$	$=$	$(\phi \triangleright f) \triangleright \psi$

Now the axiom corresponding to simulation has a definite phrasing, as a second order axiom:

SIM. $f \triangleright (Id_O \ \cup \ \phi) = (Id_I \ \cup \ \phi) \triangleright f' \quad \Rightarrow \quad f \uparrow^A_A = f' \uparrow^B_B$
for two processes $f : I \cup A \to O \cup A$ and $f' : I' \cup B \to O' \cup B$ and a function $\phi : A \to B$ (A disjoint of I, O; B disjoint of I', O')

A general axiom of this type, with ϕ an arbitrary "morphism", not only a function, was introduced in [AM80] under the name of "functoriality axiom". Functoriality axiom for particular classes of relations was widely used in various axiomatizations related to flowchart schemes (see [CaS90], for example). The present SIM axiom is related to the functoriality of functions. A weaker equational instance of the functoriality of functions is the key axiom of iteration theories [BlE93] and was used to get an axiomatisation of synchronization trees in [BlET93].

Theorem 12 *The axioms of graph isomorphism, those for splitting (in Table 2) and SIM are correct and complete for process graphs modulo bisimulation.*

Proof: (i) Completeness: By using the axioms in Table 2 it follows that every splitting operation may be eliminated, so that every expression constructed with atomic actions, constants and operations of alternative composition, feedback, renaming input or output port names and splitting input ports may be brought to a normal form. By Theorem 15 two normal form processes are bisimilar iff they may be connected by simulations via functions and this step is axiomatized by rule SIM. Hence the axioms are complete for process graphs modulo bisimulation.

(ii) Correctness: Only SIM require some explanations. We show that SIM is sound in the model of process graphs modulo bisimulation.

Suppose $P : I \cup A \xrightarrow{S} O \cup A$, $P' : I \cup B \xrightarrow{S'} O \cup B$ and a function $\phi : A \to B$ are given such that

$$P \triangleright (Id_O \cup \phi) \leftrightarrow (Id_I \cup \phi) \triangleright P'$$

Suppose T and T' are the transitions matrices corresponding to P and P', respectively. By the above note, the transition matrix corresponding to the right part is $(Id_I \cup \phi \cup Id_{S'}) \circ T'$. Bisimulation shows that for a relation $\rho \subseteq S \times S'$,

(i) $(Id_I \cup Id_A \cup \rho) \circ (Id_I \cup \phi \cup Id_{S'}) \circ T' \subseteq T \circ (Id_O \cup \phi \cup Id_S) \circ (Id_O \cup Id_B \cup \rho)$ and

(ii) $(Id_I \cup Id_A \cup \rho^{-1}) \circ T \circ (Id_O \cup \phi \cup Id_S) \subseteq (Id_I \cup \phi \cup Id_{S'}) \circ T' \circ (Id_O \cup Id_B \cup \rho^{-1})$

We shall prove that $P \uparrow^A_A \ \leftrightarrow_{\phi \cup \rho} \ P' \uparrow^B_B$.

One inclusion $(Id_I \cup \phi \cup \rho) \circ T' \subseteq T \circ (Id_O \cup \phi \cup \rho)$ directly follows from (i). For the second one we use (ii). By a left composition with $Id_I \cup \phi^{-1} \cup Id_S$ and a right one with $Id_O \cup \phi^{-1} \cup Id_S$

we get

$$(Id_I \cup \phi^{-1} \cup \rho^{-1}) \circ T \circ (Id_O \cup \phi \circ \phi^{-1} \cup Id_S) \subseteq (Id_I \cup \phi^{-1} \circ \phi \cup Id_{S'}) \circ T' \circ (Id_O \cup \phi^{-1} \cup \rho^{-1})$$

and we are done by using the properties of the function ϕ, i.e., $Id_A \subseteq \phi \circ \phi^{-1}$ and $\phi^{-1} \circ \phi \subseteq Id_B$. $\Box$

References

[AM80] M.A. Arbib and E.G.Manes. Partially additive categories and flow diagram semantics. *Journal of Algebra*, 62:203–227, 1980.

[BaB94] J.C.M. Baeten and J.A. Bergstra. Graph isomorphism models for noninterleaving process algebra. Report PRG P9401, Programming Research Group, University of Amsterdam, 1994.

[BaW90] J. Baeten and W. Weijland. *Process algebra.* Cambridge University Press, 1990.

[BBP94] J.A. Bergstra, I. Bethke and A. Ponse. Process algebra with iteration and nesting. Report P9314b, Programming Research Group, University of Amsterdam, 1994. (Revised version of Report P9314, with the title "Process algebra with iteration".)

[BeK84] J.A. Bergstra and J.W. Klop. Proces algebra for synchronous communication. *Information and Control*, 60:109–137, 1984.

[BeS93] J.A. Bergstra and Gh. Stefănescu. Translations between flowchart schemes and process graphs. In *Proceedings FCT'93*, pages 153–162. LNCS 710, Springer Verlag, 1993.

[BeS94] J.A. Bergstra and Gh. Stefănescu. Bisimulation is two-way simulation. Report PRG P9402, Programming Research Group, University of Amsterdam, 1994.

[BlE93] S.L. Bloom and Z. Esik. *Iteration theories: the equational logic of iterative processes.* EATCS Monographs in Theoretical Compputer Science, Springer Verlag, 1993.

[BlET93] S.L. Bloom, Z. Esik, and D. Taubner. Iteration theory of synchronization trees. *Information and Computation*, 102:1–55, 1993.

[CaS90] V.E. Căzănescu and Gh. Stefănescu. Towards a new algebraic foundation of flowchart scheme theory. *Fundamenta Informaticae*, 13:171–210, 1990.

[Elg75] 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.

[Elg77] C.C. Elgot. Some geometrical categories associated with flowchart schemes. In *Proceedings FCT'77*, pages 256–259. LNCS 56, Springer Verlag, 1977.

[Gog74] J.A. Goguen. On homomorphism, correctness, termination, unfoldments and equivalence of flow diagram programs. *Journal of Computer and System Sciences*, 8:333–365, 1974.

[Kle56] S.C. Kleene. Representation of events in nerve nets and finite automata. In *Automata Studies*, pages 3-41. Princeton University Press, 1956.

[Mil79] R. Milner. Flowgraphs and flow algebra. *Journal of the Association for Computing Machinery*, 26:794–818, 1979.

[Mil80] R. Milner. *A calculus of communicating systems.* LNCS 92, Springer, 1980.

[Pa80] D. Park. Concurrency and automata on infinite sequences. In *Proceedings 5th GI Conference*, 167–183. LNCS 104, Springer Verlag, 1981.

[Par93] J. Parrow. Structural and behavioural equivalence of networks. *Information and Computation*, 107:58-90, 1993.

[Ste87] Gh. Stefănescu. On flowchart theories. Part I: The deterministic case. *Journal of Computer and System Sciences*, 35:163–191, 1987.

[Ste90] Gh. Stefănescu. Feedback theories (a calculus for isomorphism classes of flowchart schemes). *Revue Roumaine de Mathematiques Pures et Applique*, 35:73–79, 1990. Early distributed as *Preprint Series in Mathematics*, The National Institute for Scientific and Technical Creation, Bucharest, No. 24/April 1986.