Processes with Multiple Entries and Exits

J.A. Bergstra[1] and 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. 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 uparrow "$\uparrow$" (feedback [Ste86]) used in the algebra of flownomials. A study of process algebra with an iteration operation (originary Kleene's star) has already been presented in [BeBP94]. Here we combine process algebra with the feedback operation.

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 [BaB95], [BeS94] and various other results on flowchart theories [Elg75, Ste87, Ste86, CaS90, BlE93, Ste94]. Like in flowchart schemes [Ste86] feedback and alternative composition "$\oplus$" suffice to express all finite state systems. ("$\oplus$" 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 [BaB95]. 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 without 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 in the case the outputs of the first process graph do not match the inputs of the second one, as well.

– 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 —e.g., 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 using alternative and sequential composition). This ϵ-free case also generates some complications in defining bisimilarity which requires an explicit splitting operator.

In this paper we study the ϵ-free model. The main results are: (i) expressiveness (already mentioned); (ii) a correct and complete axiomatisation of process graphs modulo isomorphism; (iii) a correct and complete axiomatisation of process graphs modulo strong bisimulation.

2 Process graphs modulo isomorphism

2.1 Process graphs

Notations:

– 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 $\succ\!\prec: V \times V \to V$ is given and $A \times B = \{a \succ\!\prec b \mid a \in A,\ b \in B\}$.)

– A denotes a set of atomic actions with typical elements a, b, c, d, e and closed to cartesian product.
– "$\circ$" denotes relational composition; e.g if R, S are binary relations and T a ternary relation, then $R \circ S = \{(x,z) \mid \exists y : \ (x,y) \in R \text{ and } (y,z) \in S\}$ and $R \circ T \circ S = \{(x,y,z) \mid \exists x', z' : \ (x,x') \in R, \ \ (x',y,z') \in T \text{ and } (z,z') \in S\}$.
– 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: $P = (\partial_i, E, \ell, \partial_o) : I \xrightarrow{S} O$ 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: $\partial_i : E \to I \cup S, \quad \partial_o : E \to O \cup S, \quad \ell : E \to A$

$G(A)(I,S,O)$ 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 $G(A)$). □

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); and $\partial_i(e)$, $\partial_o(e)$ 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 $e \in E$. Sometimes we write these data as $T : I \xrightarrow{S} O$.

We study two kinds of graph models: (1) with multiplicity (i.e., multiple edges with the same label between two vertices are allowed) and (2) without multiplicity.

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

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

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 $\emptyset$, $\bot^p$, $\top_p$:**

$$\emptyset = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle : \emptyset \xrightarrow{\emptyset} \emptyset, \quad \bot^p = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle : \{p\} \xrightarrow{\emptyset} \emptyset, \quad \top_q = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle : \emptyset \xrightarrow{\emptyset} \{q\}$$

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 $\uparrow^p_q$:** 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 $P \uparrow^p_q = P$.

– Otherwise, take a fresh r (not in $I \cup O \cup S \cup \{p, q\}$) and define

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

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 appears 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 appears as an exit. Finally, if p and q do not appear at all, then the result of the feedback is P itself.

– **Renaming port names $\lhd$, $\rhd$:** 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:

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

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

Effect: Rename the entry (resp. exit) ports by means of 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
$\delta^I_O := \bot^{p_1} \oplus \ldots \oplus \bot^{p_m} \oplus \top_{q_1} \oplus \ldots \oplus \top_{q_n}$; δ^p_q means $\delta^{\{p\}}_{\{q\}}$.

(2) $I(1) := \delta^r_r \uparrow^r_r$, for an $r \in V$ and $I(m) = \oplus^m_{j=1} I(1)$ 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.

(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:

$$P \odot P' = [(P \triangleright \phi) \oplus (\phi \triangleleft P')] \uparrow_{\phi(O)}^{\phi(I')}$$

Effect: Sequential composition. Note that we hide all the exits of P and all the entries of P' that have no correspondent.

– **Parallel composition** $||$:

$\star$ 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

$$P \,||\, P' = \quad \langle \partial_i^1, E^1, \ell^1, \partial_o^1 \rangle \quad : I \times I' \xrightarrow{(I \cup S \cup O) \times (I' \cup S' \cup O') - (I \times I' \cup O \times O')} O \times O'$$

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

$$\begin{aligned}
\partial_i^1 &= \partial_i \times \partial_i' \ \cup\ \partial_i \times Id_{I' \cup S' \cup O'} \ \cup\ Id_{I \cup S \cup O} \times \partial_i', \\
E^1 &= E \times E' \ \cup\ E \times (I' \cup S' \cup O') \ \cup\ (I \cup S \cup O) \times E', \\
\ell^1 &= \ell \times \ell' \ \cup\ pr_1 \circ \ell \ \cup\ pr_2 \circ \ell', \\
\partial_o^1 &= \partial_o \times \partial_o' \ \cup\ \partial_o \times Id_{I' \cup S' \cup O'} \ \cup\ Id_{I \cup S \cup O} \times \partial_o'
\end{aligned}$$

$\star$ 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

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

Effect: Parallel composition, i.e. $P \,||\, P'$ 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** $\lfloor\!\lfloor$ and **communication merge** $|$ are defined in a standard way using the definition of $||$.

– **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 $\partial_H(P) = P|_{A-H}$.

Effect: All the transitions in H are deleted. □

Example 1. An example is given in Figure 1. $P = a_{u''}^{p_1} \oplus a_{q_1}^{u'} \oplus a_{v''}^{u''} \oplus a_{v''}^{p_2} \oplus a_{q_2}^{v'}$ is illustrated in figure (a). The process $Q = P \uparrow_{u''}^{u'} \uparrow_{v''}^{v'}$ is illustrated in figure (c) with an intermediary step shown in (b). Q is particularly interesting since it cannot be represented using atomic actions, constants, alternative and sequential composition, only. □

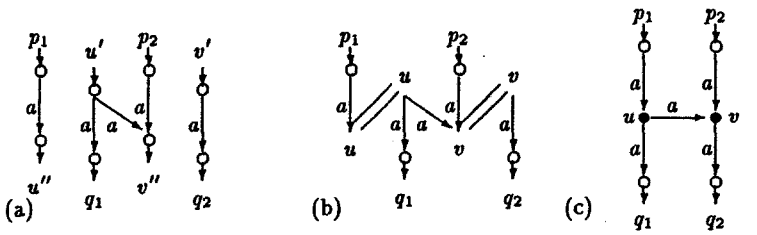

Fig. 1. Process graphs

2.3 Expressiveness

Theorem 4. *1) All graphs in $G(A)$ may be constructed starting with atomic actions or constants $\emptyset$, $\bot$, $\top$ 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.

2.4 Axioms, correctness, completeness

Let $Exp(Act, Const; \oplus, \uparrow, \triangleleft, \triangleright)$ 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.

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

$$(f_1 \oplus \ldots \oplus f_n) \uparrow_{q_1}^{p_1} \ldots \uparrow_{q_k}^{p_k} \quad \text{with } f_i \text{ of type } \bot^p,\ \top_p \text{ or } a_q^p, \text{ for all } i$$

Every expression may be brought to a prenormal form using the axioms in Table 1.

Denote $\bot^I := \oplus_{p \in I} \bot^p$, $\top_O := \oplus_{q \in O} \top_q$. We say an expression E is in a *normal form (nf)* if it is of the following type

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

where

(1) $\alpha = \bot^{I'} \oplus \top_{O'} \oplus I(m)$, for some $I' \subseteq i(E)$, $O' \subseteq o(E)$ and $m \leq s(E)$;
(2) $\beta = \oplus_{i=1}^{n} a_{s_i}^{r_i}$;
(3) $i(\alpha) \cap i(\beta) = \emptyset$, $o(\alpha) \cap o(\beta) = \emptyset$;
(4) there are no unused feedbacks, i.e, (i) $p_1, \ldots, p_k$ are all distinct and (ii) 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.

In the case without multiplicity the following condition is added:

(5) β does not contains two equal terms.

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

$R1_o.$	$a_q^p \triangleright \phi = a_{\phi(q)}^p$	$R1^i.$	$\phi \triangleleft a_q^p = a_q^{\phi(p)}$
$R2_o.$	$\bot^p \triangleright \phi = \bot^p$	$R2^i.$	$\phi \triangleleft \bot^p = \bot^{\phi(p)}$
$R3_o.$	$\top_q \triangleright \phi = \top_{\phi(q)}$	$R3^i.$	$\phi \triangleleft \top_q = \top_q$
$R4_o.$	$\emptyset \triangleright \phi = \emptyset$	$R4^i.$	$\phi \triangleleft \emptyset = \emptyset$
$RA_o.$	$(f \oplus g) \triangleright \phi = (f \triangleright \phi) \oplus (g \triangleright \phi)$	$RA^i.$	$\phi \triangleleft (f \oplus g) = (\phi \triangleleft f) \oplus (\phi \triangleleft g)$
$RF1_o.$	$(f \uparrow_q^p) \triangleright \phi = (f \triangleright \phi_{\|\{q\}\|}) \uparrow_q^p$ if $q \notin \phi(o(f) - \{q\})$	$RF1^i.$	$\phi \triangleleft (f \uparrow_q^p) = (\phi_{\|\{p\}\|} \triangleleft f) \uparrow_q^p$ if $p \notin \phi(i(f) - \{p\})$
$RF2_o.$	$f \uparrow_q^p = (f \triangleright [s/q]) \uparrow_s^p$ if $s \notin o(f)$	$RF2^i.$	$f \uparrow_q^p = ([r/p] \triangleleft f) \uparrow_q^r$ if $r \notin i(f)$

$$R'_o.\ \phi \triangleleft (f \triangleright \psi) = (\phi \triangleleft f) \triangleright \psi$$

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

Table 1. Axioms for graph isomorphism

Every expression may be brought to a normal form by using the axioms in Table 1. In addition, the normal form 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 (and identification of equal terms, in the case without multiplicity), permutations of feedbacks and bijective renamings of feedback ports. This may be used to show that two normal forms that represent the same graph may be proved equivalent via the axioms. Hence we get a first axiomatisation result.

Theorem 5. *The axioms in Table 1 (resp. the axioms in Table 1 except for A(**)) are correct and complete for graph isomorphism without (rest. with) 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. [Ste87, Ste94]). Bisimulation is an equivalence on transition systems introduced by Park [Pa81] in connection with Milner's work on concurrency [Mil80]. They may be defined as follows.

Definition 6. (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 $\phi : S \to S'$ if

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

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

$$\begin{array}{rcl} (Id_I \cup \rho) \circ T' & \subseteq & T \circ (Id_O \cup \rho) \\ (Id_I \cup \rho^{-1}) \circ T & \subseteq & T' \circ (Id_O \cup \rho^{-1}) \end{array} \quad \square$$

Theorem 7. *[BeS94] Strong bisimulation is the equivalence relation generated by simulation via functions.*

3.2 Axioms, correctness, completeness

Theorem 7 suggests that in order to axiomatize strongly 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 $\phi : S \to S'$

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

But which is the meaning of the composition here? Clearly, $T \circ (Id_O \cup \phi)$ may be replaced by $T \triangleright (Id_O \cup \phi)$. Our model is ϵ-free here, hence we have to define a new operation

$$\psi \triangleright P$$

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

The intuitive meaning of this operation is to model multiple incoming ϵ-edges to an entry by splitting 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

$$\phi \rhd P = \langle \bigcup_{i=1}^{n} pr_2^i \cup \partial_i|_{E'},\ \bigcup_{i=1}^{n} E_i \times I_i \cup E',\ \bigcup_{i=1}^{n} pr_1^i \circ \ell \cup \ell|_{E'},\ \bigcup_{i=1}^{n} pr_1^i \circ \partial_o \cup \partial_o|_{E'} \rangle : \phi^{-1}(I) \xrightarrow{S} O$$

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 $\phi \rhd P$ 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 ϕ, $\psi : V \to V$ 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 $\phi \circ \psi^{-1} = \psi'^{-1} \circ \phi'$.

$S1.$	$\phi \rhd a_q^p =$ if $\phi^{-1}(p) \neq \emptyset$ then $\Plus_{r \in \phi^{-1}(p)} a_q^r$ else $\top_q$
$S2.$	$\phi \rhd \bot^p =$ if $\phi^{-1}(p) \neq \emptyset$ then $\Plus_{r \in \phi^{-1}(p)} \bot^r$ else $\emptyset$
$S3.$	$\phi \rhd \emptyset = \emptyset$
$S4.$	$\phi \rhd \top_q = \top_q$
$SA.$	$\phi \rhd (f \Plus g) = (\phi \rhd f) \Plus (\phi \rhd g)$
$SF.$	$\phi \rhd (f \restriction_q^p) = (\phi_{\mid \{p\} \cup \phi^{-1}(p) \mid} \rhd f) \restriction_q^p$
$SR^i.$	$\phi \rhd (\psi \lhd f) = \psi' \lhd (\phi' \rhd f)$ where ϕ', ψ' are functions such that $\phi \circ \psi^{-1} = \psi'^{-1} \circ \phi'$
$SR_o.$	$\phi \rhd (f \rhd \psi) = (\phi \rhd f) \rhd \psi$

Table 2. Axioms for splitting

Now the axiom corresponding to simulation has a definite phrasing:

$$SIM.\ f \rhd (Id_O \cup \phi) = (Id_I \cup \phi) \rhd f' \quad \Rightarrow \quad f \restriction_A^A = f' \restriction_B^B$$

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')

Theorem 8. *The axioms of graph isomorphism, those for splitting (in Table 2) and SIM are correct and complete for process graphs modulo bisimulation.*

An axiom scheme analogous to SIM, —sometimes known as a "functoriality axiom"—, was widely used in various axiomatizations related to flowchart schemes (see [CaS90, Ste94], 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 and was used to get an axiomatisation of synchronization trees, cf. [BlE93].

4 Conclusions

We have presented a model of process graphs with multiple entries and exits. This model allows for a smooth integration of looping and parallel operators. Axiomatisations for such processes under graph isomorphism and strong bisimulation equivalences are given.

An extension of this model is given in [BaBS95]. There more operations and results may be found, including axiomatisation results for parallel operators.

References

[BaB95] J.C.M. Baeten and J.A. Bergstra. Graph isomorphism models for noninterleaving process algebra. In: Proceedings of ACP94 (Eds. A. Ponse, C. Verhoef and S.F.M. van Vlijmen), 299–318. Workshops in Computing, Springer–Verlag, 1995.

[BaBS95] J.C.M. Baeten, J.A. Bergstra and Gh. Stefanescu. Process algebra with feedback. In: Proceedings of the workshop: Three Days of Bisimulation (Eds. A. Ponse, M. de Rijke and Y. Venema). To appear as: CSLI volume, Stanford, 1995.

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

[BeBP94] J.A. Bergstra, I. Bethke and A. Ponse. Process algebra with iteration and nesting. *The Computer Journal*, 37:243–258, 1994.

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

[BeS94] J.A. Bergstra and Gh. Stefănescu. Bisimulation is two-way simulation. *Information Processing Letters*, 52:285–287, 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.

[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.

[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.

[Pa81] 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.

[Ste86] Gh. Stefănescu. Feedback theories (a calculus for isomorphism classes of flowchart schemes). INCREST Preprint No. 24, Bucharest, April 1986. Also in: *Revue Roumaine de Mathematiques Pures et Applique*, 35:73–79, 1990.

[Ste94] Gh. Stefănescu. *Algebra of Flownomials. Part 1: Binary flownomials; Basic Theory.* Report TUM-I9437, Technical University Munich, 1994.