Fundamenta Informaticae 27 (1996) 37–56
IOS Press

PROCESSES WITH MULTIPLE ENTRIES AND EXITS MODULO ISOMORPHISM AND MODULO BISIMULATION*

J.A. BERGSTRA
Programming Research Group, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands; Department of Philosophy, Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, The Netherlands. E-mail: janb@fwi.uva.nl

Gh. ŞTEFĂNESCU
Department of the Fundamentals of Computer Science, University of Bucharest, Str. Academiei 14, 70109 Bucharest, Romania. E-mail: ghstef@imar.ro

Abstract. This paper proposes a framework for the integration of the algebra of communicating processes (ACP) and the algebra of flownomials (AF). Basically, this means to combine axiomatisations of parallel and looping operators. To this end a model of process graphs with multiple entries and exits is introduced. In this model the usual operations of both algebras are defined, e.g. alternative composition, sequential composition, feedback, parallel composition, left merge, communication merge, encapsulation, etc. The main results consist of correct and complete axiomatisations for process graphs modulo isomorphism and modulo bisimulation.

1 Introduction

This paper proposes a framework to integrate the algebra of communicating processes (ACP) and the algebra of flownomials (AF). Basically, this means to combine axiomatisations of parallel and looping operators.

There are three axiomatized looping operations that may be combined with the existing process algebra: Kleene's star "$*$" (repetition [17]) used in regular algebra, Elgot's dagger "$\dagger$" (iteration [14]) used in iteration theories, or uparrow "$\uparrow$" (feedback [23]) used in the algebra of flownomials. A study of process algebra with an iteration operation (originally Kleene's binary star operator) has already been presented in [5]. Here we combine process algebra with the feedback operation.

Our goal is to define a process algebra in which all operators of ACP [6] are present as well as feedback, the key iteration construct of flowchart theories. To this end we combine the results of [2], [8] and various other results on flowchart theories [11, 13, 14, 16, 22, 23]. Like in flowchart schemes [23] feedback and alternative composition " $\oplus$ " suffice to express all finite state systems. (Actually, " $\oplus$ " 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 for transition systems modulo isomorphism. A similar result on undirected networks is presented in [21], an approach that simplifies the earlier algebra of flow graphs of Milner [18].

*An extended abstract of this paper has been presented at FCT'95 conference in Dresden and has been published in its proceedings [9].

Technically we depart from the graph isomorphism model for the operators of ACP that was outlined in [2]. This model is adapted to allow for process graphs with multiple entry states and multiple exit states. Then the feedback operator is introduced. For technical reasons renaming operators for entry and exit states are needed. As the name indicates, entries and exits are just a particular kind of states. A port is either an entry state or an exit state. We will need a naming mechanism for ports.

The main features of this model are:

1) The process graphs have multiple entry states and multiple exit states. 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: (i) in the case witout ϵ (i.e., without empty transitions) all process graphs with no incoming edges into the entry states, no outgoing edges from the exit states and such that no entry state is an exit state are represented; (ii) if ϵ is allowed, then all process graphs are represented.

2) In this model the operations are totally defined by providing a default system of working in the case the types do not agree. For instance, seqential composition is defined in the case the outputs of the first process graph do not match the inputs of the second one.

3) There are two options regarding the naming mechanism for ports: (i) to use an arbitrary set of port names and renamings on them or (ii) to order the entry (resp. exit) ports and to use inplicitely the first natural numbers as names for them; then one may use an explicit algebra for the represention of the finite relations and it may model the renamings by composition with appropriate relations. In the presentation below we will use the first variant which gives more freedom (because it is more abstract) and which is perhaps easier to understand. The second version may be more suited for implementations, however.

4) The main results presented here are for the model without ϵ steps. The loss of expressivity is small in the case of cyclic processes, as we already mention. However, in the acyclic case not all acyclic processes may be represented by using alternative and sequential composition, only. This ϵ-free case also generates some complications in the definition of bisimulation which requires an explicit splitting operator.

Besides the value of the model itself, the main results of the paper are: certain expresiveness results (already mentioned), a correct and complete axiomatisation of process graphs modulo isomorphism, and a correct and complete axiomatisation of process graphs modulo bisimulation.

2 Process graphs modulo isomorphism

In this section the model of process graphs with multiple entry states and multiple exit states is introduced. Various operations on such process graphs are defined. Some expressivity results and an axiomatisation theorem for such process graphs modulo graph isomorphism are provided.

2.1 *Process graphs*

This subsection is devoted to the introduction of basic process graphs definitions. Before giving them, a few notational conventions are fixed:

- V denotes a set of states names with typical elements p, q, r, s, t, u, v. We need that V is closed under taking ordered pairs and we assume an injective coding $\rightarrow\!\!<$: $V \times V \rightarrow V$ is

given. The inverse (partially defined) function is denoted by $\asymp^{-1}: V \to V \times V$.

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

– "$\circ$" is a polymorphic notaion for a relational composition of relations (an instance of relational join operator). That is, if $R \subseteq A_1 \times \ldots \times A_m$ is an m-ary relation and $S \subseteq B_1 \times \ldots \times B_n$ is an n-ary relation for some $m, n \geq 2$, then $R \circ S \subseteq A_1 \times \ldots \times A_{m-1} \times B_2 \times \ldots \times B_n$ is the $m + n - 2$ relation defined by

$$R \circ S = \{(x_1, \ldots, x_{m-1}, y_2, \ldots, y_n) \mid \exists z \in A_m \cap B_1 : (x_1, \ldots, x_{m-1}, z) \in R \text{ and } (z, y_2, \ldots, y_n) \in S\}$$

Notice that in the case of binary relations the present definition is equivalent to the usual definition of composition, but it is written in the diagramatic order and it is slightly extended (the requirement that $A_m = B_1$ is dropped).

– 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 domain of definition.

– $[n] = \{1, \ldots, n\}$

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

Definition 2.1 (process graphs with multiple entries and multiple exits) Given finite subsets I, S, O of V a *process graph* of type $I \xrightarrow{S} O$ is $P = (\partial_i, E, l, \partial_o)$ where:

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

The meaning of these data is the following. Elements of I, S and O are called respectively, entry states, exit states, and internal states of P. The set E specifies the edges (transitions). The values $\partial_i(e)$, $\partial_o(e)$, and $\ell(e)$ give respectively the source state, the target state and the label of an edge e.

Notice that a process graph obeys the following restriction: "an entry state has no incoming edges, an exit state has no outgoing edges and an entry state is not an exit state (however, an entry state and an exit state may share the same name)". Conversely, all transition diagrams with distinguished entry and exit states and fullfilling the above condition are "process graphs" according to the given definition.

A process graph $P = (\partial_i, E, l, \partial_o) : I \xrightarrow{S} O$ may also be specified by using the transition (ternary) relation

$$T \subseteq (I \cup S) \times A \times (O \cup S)$$

consisting of the transitions $(\partial_i(e), \ell(e), \partial_o(e))$ (often written $\partial_i(e) \xrightarrow{\ell(e)} \partial_o(e)$), for $e \in E$. There are two possibilities here: (i) to allow for multiple edges with the same label between two vertices (in such a case the transition relations consists of multisets of transitions) or (ii) to avoid such a possibility (in that case the transition relations are simply given by sets of transitions). For the sake of simplicity we treat the later case. Hoever, the result in Theorem 2.9 on the axiomatisation of process graphs modulo isomorphism may be easily extended to the first case. Actually only axiom $A6$ of Table 1 has to be dropped and one gets an axiomatisation for the former case.

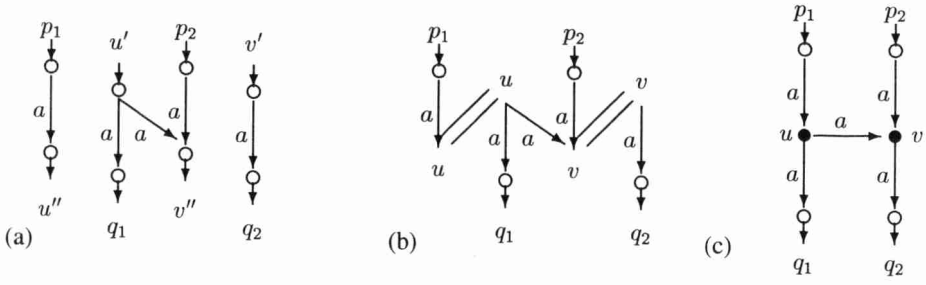


Figure 1: Process graphs

Definition 2.2 Let $P : I \xrightarrow{S} O$, $P' : I \xrightarrow{S'} O$ be two process graphs and T, T' their associated transition relations, respectively. A bijection $\phi : S \to S'$ is called an *isomorphism* if

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

Example 2.3 In the graphical representation of the processes, an internal state is represented by a filled (black) circle, an entry state by an open circle with a short incoming arrow, and an exit state by an open circle with a short outgoing arrow.

An example is given in Figure 1. The process graph corresponding to the expression $P = a^{p_1}_{u''} \oplus a^{u'}_{q_1} \oplus a^{u'}_{v''} \oplus a^{p_2}_{v''} \oplus a^{v'}_{q_2}$ is illustrated in (a). The process graph corresponding to $Q = P \uparrow^{u'}_{u''} \uparrow^{v'}_{v''}$ is illustrated in (c), with an intermediary step shown in (b). The way to associate graphical representation to process expressions will be explained in the next subsection. Q is particularly interesting since it cannot be represented without feedback, i.e. using atomic actions, constants, alternative and sequential composition, only; see Subsection 2.3 below. $\square$

2.2 *Operations on process graphs*

In this subsection the operations on process graphs are introduced. They are defined on process graphs modulo isomorphism.

1. **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 $\partial_i(\star) = p$, $\ell(\star) = a$, $\partial_o(\star) = q$.
   An illustration is given in Figure 2.

2. **Empty processes:** $\emptyset$, $\bot^p$, $\top_p$
   $\emptyset = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle : \emptyset \xrightarrow{\emptyset} \emptyset$, $\bot^p = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle : \{p\} \xrightarrow{\emptyset} \emptyset$,
   $\top_q = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle : \emptyset \xrightarrow{\emptyset} \{q\}$
   Illustrations are given in Figure 2.

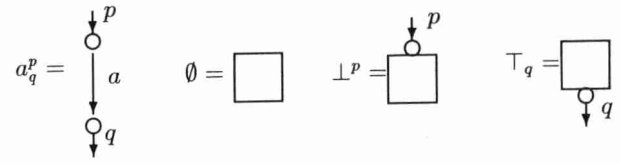


Figure 2: Atomic actions and constants

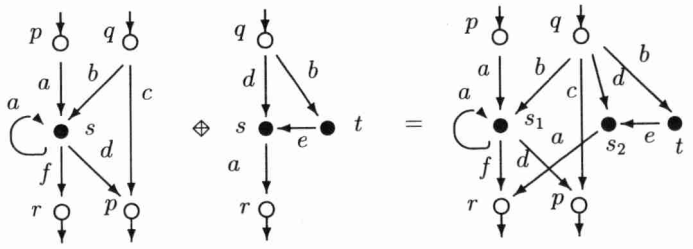


Figure 3: Alternative composition

3. **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: Union on entry states, union on exit states and disjoint union on all other components.

   Recall that the operations are defined on classes of isomorphic process graphs. Hence, we always may rename the internal states of a process in order to meet the condition in this definition. Consequently, $\oplus$ is a totally defined operation.

   An example is shown in Figure 3. Both graphs use q as a name of an entry state. Hence, there is one state q of the resulting graph which collects the outgoing transitions from both components. Similarly with the exit states named by r. On the other hand, s is a name for internal states in both graphs, but the corresponding internal states are not identified in the resulting graph.

4. **Feedback:** $\uparrow_q^p$

   Let $P = \langle \partial_i, E, \ell, \partial_o \rangle : I \xrightarrow{S} O$ be given.

   - Case $p \notin I$ and $q \notin O$: Then $P \uparrow_q^p = P$.
   - Case $p \in I$ and $q \notin O$: Then $P \uparrow_q^p = \langle \partial_i, E, \ell, \partial_o \rangle \quad : I - \{p\} \xrightarrow{S \cup \{p\}} O$

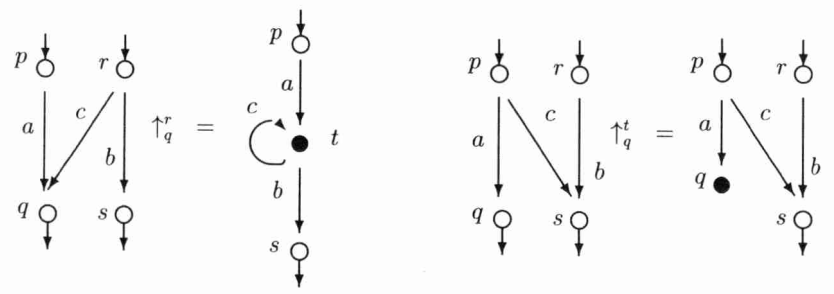


Figure 4: Feedback

- The case $p \notin I$ and $q \in O$ is similar.
- Case $p \in I$ and $q \in O$: Take a fresh r (not in $I \cup O \cup S$) and define

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

Effect: If there are an entry state p and an exit state q, then they are identified as a unique new internal state r. If q does not appear as an exit state and p appears as an entry state, then the effect is that p becomes an internal state. Similarly if p does not appear as an entry state and q appears as an exit state. Finally, if both p and q do not appear at all, then the result of the feedback is P itself.

Some examples are shown in Figure 4.

5. **Renaming port names:** $\triangleleft$, $\triangleright$

Let $\phi : V \to V$ be a renaming function and $P = \langle \partial_i, E, \ell, \partial_o\rangle : I \xrightarrow{S} O$. Taking an isomorphic representation for P we may assume 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) states by using ϕ and identify the entry (resp. the exit) states that becomes equal.

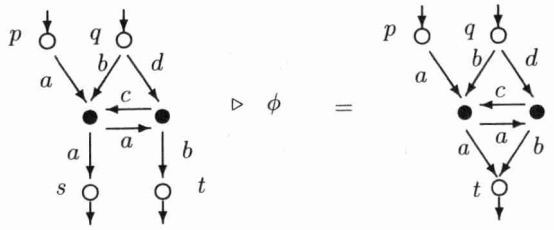
An example of a renaming for the exit states is given in Figure 5.

Notations: In order to simplify the definitions that follows, we still introduce a few additional notational conventions.

– For $I = \{p_1, \ldots, p_m\}$ and $O = \{q_1, \ldots, q_n\}$ we define

$$\bot^I := \bot^{p_1} \oplus \ldots \oplus \bot^{p_m},\quad \top_O := \top_{q_1} \oplus \ldots \oplus \top_{q_n},\quad \delta_O^I := \bot^I \oplus \top_O$$

Moreover, δ_q^p abbreviates $\delta_{\{q\}}^{\{p\}}$.



where $\phi(s) = \phi(t) = t$

Figure 5: Renaming

– It is meaningful (up to isomorphism) to introduce the constants

$$I(1) := \delta_r^r \uparrow_r^r, \quad r \in V \quad \text{and} \quad I(m) = \oplus_{j=1}^{m} I(1), \quad m \in \mathbb{N}$$

Up to isomorphism $I(m)$ is $\langle \emptyset, \emptyset, \emptyset, \emptyset \rangle : \emptyset \xrightarrow{S} \emptyset$, for an S with m elements.

– 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

$$P \uparrow_B^A := P \uparrow_{p_1}^{p_1} \cdots \uparrow_{p_k}^{p_k} \uparrow_{q_1'}^{q_1} \cdots \uparrow_{q_m'}^{q_m} \uparrow_{r_1}^{r_1'} \cdots \uparrow_{r_n}^{r_n'}$$

where $q_1', \ldots, q_m', r_1', \ldots, r_n'$ are fresh variables. (Notice that this convention works on graphs modulo isomorphism.)

– Let $P = \langle \partial_i, E, \ell, \partial_o \rangle : I \xrightarrow{S} O$ and $E' \subseteq E$ be given. The restriction of P to E is

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

Now we may continue the presentation of the definitions of the process graph operators.

6. **Sequential composition:** $\odot$

   Let $P : I \xrightarrow{S} O$ and $P' : I' \xrightarrow{S'} O'$ 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. Notice that all the exit states of P (resp. all the entry states of Q) that have no correspondent entry state of Q (resp. exit state of P) are hidden.

   An example is given in Figure 6.

7. **Parallel composition:** $||$

   Assume $\gamma : A \times A \to A$ is a (partial) communication function; $dom(\gamma)$ denotes its domain of definition.

   1) First, we define $||$ for processes $P : I \to O$ with $I \cap O = \emptyset$.

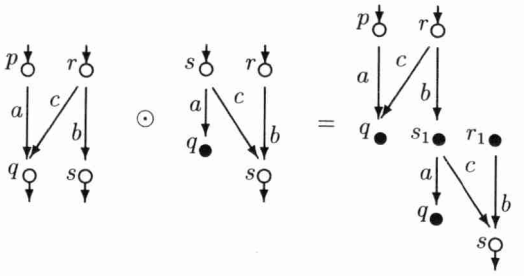


Figure 6: Sequential composition

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. Denote $\overline{E} = \{(e, e') \in E \times E' \mid (l(e), l'(e')) \in dom(\gamma)\}$.

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

$$\begin{array}{rcl} \partial_i^1 & = & ((\partial_i \times \partial'_i)|_{\overline{E}} \cup \partial_i \times Id_{I' \cup S' \cup O'} \cup Id_{I \cup S \cup O} \times \partial'_i) \circ \asymp \\ E^1 & = & \overline{E} \cup E \times (I' \cup S' \cup O') \cup (I \cup S \cup O) \times E' \\ \ell^1 & = & (\ell \times \ell')|_{\overline{E}} \circ \gamma \cup pr_1 \circ \ell \cup pr_2 \circ \ell' \\ \partial_o^1 & = & ((\partial_o \times \partial'_o)|_{\overline{E}} \cup \partial_o \times Id_{I' \cup S' \cup O'} \cup Id_{I \cup S \cup O} \times \partial'_o) \circ \asymp \end{array}$$

and pr_1 (resp. pr_2) denotes the 1st (resp. the 2nd) projection.

2) 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

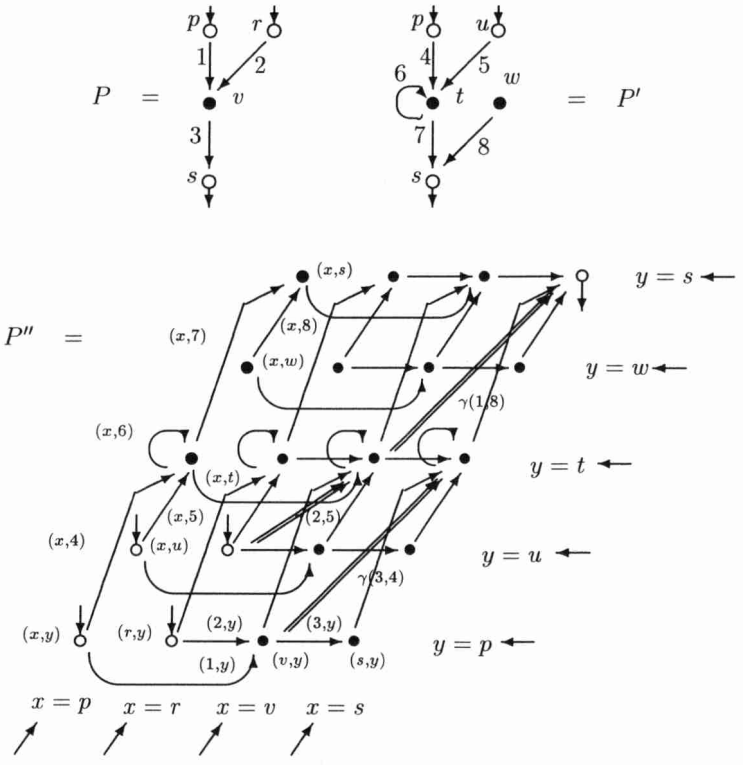
$$P \,||\, P' = [(P \triangleright \phi) \,||\, (P' \triangleright \psi)] \;\triangleright\; (\asymp^{-1} \circ (\phi^{-1} \times \psi^{-1}) \circ \asymp)$$

This complication is generated by the fact that all the states in a process are coupled to the states of the other process. This may create confusion when certain entry and exit states in a process share a common name. For this reason certain auxiliary renaming functions are necessary.

Effect: Parallel composition, i.e. $P \,||\, P'$ performs a transition from P, or a transition from P', or a communicating action corresponding to a pair of transitions (one from P and one from P'), if possible.

Notice that, in general, $P \,||\, P'$ is $\langle \partial_i^2, E^1, \ell^1, \partial_o^2 \rangle$, where E^1 and ℓ^1 are as in the first case above and ∂_i^2 (resp. ∂_o^2) is obtained by composing ∂_i^1 (resp. ∂_o^1) with appropriate bijective renamings.

An example is shown in Figure 7. In order to avoid the spelling of the labels for all the components in the resulting graph $P||P'$ we use some conventions: (a) The coding



The communication function is undefined, except for the pairs $\{(1,8),(2,5),(3,4)\}$

Figure 7: Parallel composition

function $\asymp$ is not explicitely stated. The final result is obtained by applying $\asymp$ to all the states of the graph P'' in Figure 7 which are pairs of states of P and P'. (b) The graph P'' contains one horizontal slice copy of P for each state in P'. The components of this slice are denoted by (x,y), for $y \in \{p,u,t,w,s\}$. They are explicitely mentioned in the figure only for the leftmost slice, where $x = p$. (c) P'' contains one vertical slice copy of P' for each state in P. The components of this slice are denoted by (x,y), for $x \in \{p,r,v,s\}$. They are mentioned in the figure only for the bottom slice, where $y = p$. (d) Finally, P'' contains communicating actions which are represented by double arrows. In the case of Figure 7 there are 3 such actions with labels in $\{\gamma(1,8),\gamma(2,5),\gamma(3,4)\}$.

8. **Left merge:** $\between$

Let P, P' be given and $P \parallel P' = \langle \partial_i^2, E^1, \ell^1, \partial_o^2 \rangle$ be constructed as above. Denote $E_l = \{(e,e') \in \overline{E} \mid \partial_i(e) \in I \text{ and } \partial_i'(e') \in I'\} \cup \{(r,e') \in I \times E' \mid \partial_i'(e') \in I'\}$. Then

$$P \between P' = (P \parallel P')|_{E^1 \setminus E_l}$$

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

9. **Communication merge:** $|$

   Take $E_c = \{(e, r') \in E \times I' \mid \partial_i(e) \in I\} \cup \{(r, e') \in I \times E' \mid \partial'_i(e') \in I'\}$. where the notations in the definition of parallel composition are used. Then

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

   Effect: As in $P \parallel P'$, but the outgoing transitions from an entry state are communicating actions only.

10. **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 $E_{A-H} = \{e \in E \mid \ell(e) \notin H\}$. Then

$$\partial_H(P) = P|_{E_{A-H}}$$

    Effect: All the transitions with label in H are deleted.

2.3 *Expressivity of the process graph operators*

A few expressivity results regarding the process graph operations are presented here for the parallel-free syntax.

Theorem 2.4 *1) All finite process graphs with multiple entry states and multiple exit states may be obtained up to isomorphism from atomic actions and constants in* $\emptyset$, $\bot$, $\top$ *by alternative composition and feedback only.*

2) There are finite acyclic process graphs that isomorphis copies obtained from atomic actions and constants in $\emptyset$, $\bot$, $\top$ *by alternative and sequential composition, only.*

Proof: 1) Suppose $P : I \xrightarrow{S} O$ is given. Let $(a_1)^{p_1}_{q_1}, \ldots, (a_k)^{p_k}_{q_k}$ be all the edges of P with the corresponding source and target state name. Let $I' \subseteq I$, $O' \subseteq O$ and $S' \subseteq S$ be the sets of all the states in P without both incoming or outgoing edges. Finally, assume $S = \{r_1, \ldots, r_n\}$. Then

$$(*) \quad P = [(\Phi_{p \in I'} \bot^p) \Phi (\Phi_{q \in O'} \top_q) \Phi (\Phi_{r \in S'} \delta^r_r) \Phi (a_1)^{p_1}_{q_1} \Phi \ldots \Phi (a_k)^{p_k}_{q_k}] \uparrow^{r_1}_{r_1} \ldots \uparrow^{r_n}_{r_n}$$

2) Let $H : \{p_1, p_2\} \xrightarrow{\{u,v\}} \{q_1, q_2\}$ be the acyclic process graph given by the transitions $p_1 \xrightarrow{a} u \xrightarrow{a} q_1$, $p_2 \xrightarrow{a} v \xrightarrow{a} q_2$, $u \xrightarrow{a} v$ Actually the corresponding graph has been drawn in Figure 1.

H cannot be represented from atomic actions and constants using alternative and sequential composition only. First, it is clear that H cannot be written as a sum $P \Phi Q$ with both P and Q nonempty. Next, suppose H is written as a composite $P \odot Q$. Then u cannot be an internal state in P. (If so, then q_1 would also be a state of P and could not be an exit state 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 it follows that H cannot be written as a composite of two process graphs either and the theorem is proved. □

2.4 *Axiomatising process graphs modulo isomorphism*

In this section we provide an axiomatisation for process graphs modulo graph isomorphism. One may state two versions of the result, according to whether the transitions form a set or a multiset. For simplicity we present the proof for the former case. In the case of multisets of transitions the proof is similar and an axiomatisation may be given using the axioms in Table 1 with axiom A6 removed.

Let $Exp(Act, Const;\ \oplus, \uparrow, \triangleleft, \triangleright)$ be the set of expressions constructed from 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 respectively, the set of entry states, the set of exit states and the number of the internal states. Formally they are defined as follows:

$$i(a_q^p) = \{p\},\ o(a_q^p) = \{q\},\ s(a_q^p) = 0;$$

$$i(\bot^p) = \{p\},\ o(\bot^p) = \emptyset,\ s(\bot^p) = 0;$$

$$i(\top_p) = \emptyset,\ o(\top_p) = \{p\},\ s(\top_p) = 0;$$

$$i(\emptyset) = o(\emptyset) = \emptyset,\ s(\emptyset) = 0;$$

$$i(P \oplus Q) = i(P) \cup i(Q),\ o(P \oplus Q) = o(P) \cup o(Q),\ s(P \oplus Q) = s(P) \cup s(Q);$$

$$i(P \uparrow_q^p) = i(P) - \{p\},\ o(P \uparrow_q^p) = o(P) - \{q\},$$
$$s(P \uparrow_q^p) = \text{“if } p \notin i(P) \text{ and } q \notin o(P) \text{ then } s(P) \text{ else } s(P) + 1\text{”};$$

$$i(P \triangleright \phi) = i(P),\ o(P \triangleright \phi) = \phi(o(P)),\ s(P \triangleright \phi) = s(P);$$

$$i(\phi \triangleleft P) = \phi(i(P)),\ o(\phi \triangleleft P) = o(P),\ s(\phi \triangleleft P) = s(P).$$

The axiomatisation is given in Table 1. The following notations are used in some axioms:

$$[q/p](x) = \text{“if } x = p \text{ then } q \text{ else } x\text{”}$$

$$\phi_{|A|}(x) = \text{“if } x \in A \text{ then } x \text{ else } \phi(x)\text{”}$$

The proof of the correctness of the axiomatisation in Table 1 may be done by means of straightforward verifications. As usual, for the completeness part a normal form should be detected such that every expression may be brought to the normal form via the axioms. In addition, the equivalent normal forms should be connected via the axioms, too.

A first step in the completeness proof is the introduction of an intermediary form for expressions. We say an expression is in a *prenormal form* if it is of the following form:

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

Lemma 2.5 *Every expression may be brought to a prenormal form via the axioms in Table 1.*

Proof: The constants are in prenormal form. For operations, it suffices to show that the result of an operation may be brought to a prenormal form via the given axioms whenever the arguments are prenormal form expressions.

1) For $\uparrow$ it is clear: if E is in a prenormal form, then $E \uparrow_q^p$ is in a prenormal form, too.

2) For $\triangleright$: We prove by induction on k that if E is a prenormal form with k feedbacks, then $E \triangleright \phi$ may be brought to a prenormal form. For $k = 0$ it is obvious.

Table 1: Axioms for graph isomorphism

$$
\begin{array}{lrcll}
A1. & f \oplus (g \oplus h) & = & (f \oplus g) \oplus h & \\
A2. & f \oplus g & = & g \oplus f & \\
A3. & f \oplus \emptyset & = & f & \\
A4. & f \oplus \bot^p & = & f & \text{if } p \in i(f) \\
A5. & f \oplus \top_q & = & f & \text{if } q \in o(f) \\
A6 & f \oplus f & = & f &
\end{array}
$$

$$
\begin{array}{lrcll}
F1. & f \uparrow_q^p \uparrow_s^r & = & f \uparrow_s^r \uparrow_q^p & \text{if } p \neq r \text{ and } q \neq s \\
F2. & f \uparrow_q^p & = & f & \text{if } p \notin i(f) \text{ and } q \notin o(f) \\
FA1. & f \oplus (g \uparrow_q^p) & = & (f \oplus g) \uparrow_q^p & \text{if } p \notin i(f) \text{ and } q \notin o(f) \\
FA2. & (f \oplus \bot^p) \uparrow_q^p & = & f \uparrow_q^p & \text{if } q \in o(f) \\
FA3. & (f \oplus \top_q) \uparrow_q^p & = & f \uparrow_q^p & \text{if } p \in i(f)
\end{array}
$$

$$
R_o^i. \quad \phi \triangleleft (f \triangleright \psi) = (\phi \triangleleft f) \triangleright \psi
$$

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

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

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

The statement follows from the inductive hypothesis.

3) Similarly for $\triangleleft$.

Consequently, all renamings may be shifted to act on atomic actions and constants and then they may be eliminated using the axioms $R1_o$-$R4_o$ and $R1^i$-$R4^i$

4) For $\oplus$: Assume E and E' are in prenormal form, say $E = (f_1 \oplus \ldots \oplus f_n) \uparrow_{q_1}^{p_1} \ldots \uparrow_{q_k}^{p_k}$ and $E' = (f_1' \oplus \ldots \oplus f_{n'}') \uparrow_{q_1'}^{p_1'} \ldots \uparrow_{q_{k'}'}^{p_{k'}'}$. The axioms of type $RF2$ and the properties proved in 2) and 3) above show 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 $A2$

$$
E \oplus E' = (E \oplus f_1' \oplus \ldots \oplus f_{n'}') \uparrow_{q_1'}^{p_1'} \ldots \uparrow_{q_{k'}'}^{p_{k'}'}
$$

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

Hence every expression may be brought to a prenormal form via the axioms in Table 1. □

Next we pass to normal form expressions.

Definition 2.6 We say an expression E is in a *normal form* if it is of the following type

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

where

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

(2) $\beta = \oplus_{i=1}^{n} a^{r_i}_{s_i}$;

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

(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 $a^{p_i}_q$ or a term $a^r_{p_i}$ in β

(5) β does not contains two equal terms □

Lemma 2.7 *Every expression E may be brought to a normal form via the axioms in Table 1.*

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

a) Eliminate a term $f_i = \perp^p$ if either

- there exists another term $f_j = \perp^p$ $(j \neq i)$ or $f_j = a^p_q$ (use axiom $A4$) or
- there exists a term $f_j = a^r_s$ or $f_j = \top_s$ in the sum and a feedback $\uparrow^p_s$ (use axiom $AF2$)

b) Similarly for $\top$.

Repeat these reductions as long as possible. Finally we get a prenormal form such that:

- for each term $\perp^p$ (resp. $\top_q$) of the sum there is no other term t in the sum such that $p \in i(t)$ (resp. $q \in o(t)$);
- for each term $\perp^p$ (resp. $\top_q$) that is in the scope of a feedback $\uparrow^p_q$ there is no term t in the sum such that $q \in o(t)$ (resp. $p \in i(t)$).

The next step is to move all the feedbacks that apply to $\perp^p$ or $\top_p$ terms near that terms and then to replace the resulting terms by $(\delta^p_p) \uparrow^p_p (= I(1))$, for esthetical reasons.

Finally a normal form is obtained by eliminating the unused feedbacks (cf. axiom $F2$) and replacing the feedbacks $\uparrow^p_q$ by $\uparrow^r_r$ for certain fresh names r (this transformation is allowed by $RF2$ and $F1$). □

Lemma 2.8 *(uniqueness of the normal form)*

(1) 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, permutations of feedbacks and bijective renamings of feedback ports.

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

Proof: All the components of a normal form are uniquely determined by the corresponding process graph as follows:

- I' of $\perp^{I'}$ is the set of the entry states without outgoing edges;
- O' of $\top_{O'}$ is the set of the exit states without incoming edges;
- m of $I(m)$ is the number of the internal states without both incoming or outgoing edges;
- k is the number of the internal states having at least an incoming or an outgoing edge;
- the terms in β are in a bijective correspondence to the edges of the graph.

By axioms $A2$, $F1$ and $RF2$ one may pass from a normal form to another one that represents the same graph. □

Theorem 2.9 *The axioms in Table 1 are correct and complete for finite process graphs modulo graph isomorphism.*

Proof: The correctness is obvious; – it may be checked by means of straightforward verifications. The completeness part follows by the above lemmas. □

3 Process graphs modulo bisimulation

In this section we show that the algebra of (strongly) bisimilar processes may be easily introduced and studied on top of the algebra of isomorphic process graphs.

3.1 *Simulation and bisimulation*

In this subsection we provide two different formal ways to define the bisimulation equivalence on process graphs. One definition is a relational algebra formalisation of the usual bisimulation definition. The second equivalent definition is an adaptation of the definition of the standard flowchart scheme equivalence is terms ðf simulation via functions.

Simulation is a standard notion of graph homomorphism that has been used in the study of flow diagram programs (see, e.g. [15, 16, 22]). Bisimulation is an equivalence on transition systems introduced by Park [20] in connection with Milner's work on concurrency [19]. In [8] it is shown that bisimulation is the equivalence relation generated by simulation via functions. The proof in [8] uses a translation between flowchart schemes and process graphs. Another direct and much simpler proof of this fact is given in [10] using transition systems and simple rules of the calculus of relations, only.

Definition 3.1 (simulation and bisimulation) Assume $P : I \xrightarrow{S} O$, $P' : I \xrightarrow{S'} O$ are two process graphs and T, T' are the associated transition relations, respectively.

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

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

Notice that simulation $\to$ is a transitive, but not a symmetric relation, whilst bisimulation $\leftrightarrow$ is an equivalence relation. This may be seen as an advantage of the direct bisimulation definition. On the other hand, in the formal version of the definitions above, the simulation definition uses an equality relation, while the bisimulation is defined by two inequalities. This may be seen as an advantage of the simulation-based definition of bisimulation leading to equational logic reasoning.

Theorem 3.2 *(cf. [10]) Bisimulation is the equivalence relation generated by simulation via functions. More precise:* $\leftrightarrow \; = \; \leftarrow \circ \rightarrow$, *where* $\leftarrow$ *denotes the converse of* $\rightarrow$. $\square$

There are two main steps in the proof of this theorem. The first observation is that simulation via a function coincides with bisimulation via a function. Then, in order to pass from bisimilar process graphs to chains of similar ones it is necessary to have a characterization of bisimulation via arbitrary relations in terms of bisimulation via functions. An interpolation result solves the problem by the construction of a common refinement of two bisimilar process graphs which is functionally bisimilar to each of them.

Another possibility may be to use the minimal process graphs. Actually, simulation is an useful tool to speak about minimization. A *direct* simulation via a surjective functions models identification of states with the same behaviour and the *converse* of a simulation via an injective function models the deletion of nonaccessible parts.

3.2 *Axiomatising strongly bisimilar process graphs*

In this subsection we provide an axiomatisation of strongly bisimilar process graphs. The axiomatisation is based on the graph-isomorphism axiomatisation given in the previous section. However, it uses an additional "splitting" operation on process graphs, which is firstly introduced.

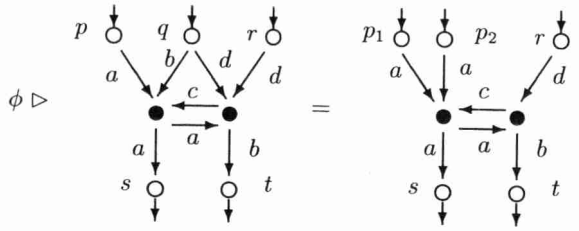
By theorem 3.2 one may obtain an axiomatisation of the classes of strongly bisimilar processes adding a new axiom corresponding to simulation via functions. As a starting form for this intended axiom one may consider the following: for a function $\phi : S \to S'$

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

Which is the meaning of the composition here? One term $T \circ (Id_O \cup \phi)$ is correctly modelled by $T \rhd (Id_O \cup \phi)$. Since our model is ϵ-free, in order to give meaning to the other term $(Id_O \cup \phi) \circ T'$ we have to define a new operation

$$\psi \rhd P$$

which has to simulate the effect of the composition. The intuitive meaning of this operation is that we have multiple incoming ϵ-edges to an entry state. It may be modelled on the classes of



where $\phi^{-1}(p) = \{p_1, p_2\}, \quad \phi^{-1}(q) = \emptyset, \quad \phi^{-1}(r) = r$

Figure 8: Splitting

strongly bisimilar process graphs by splitting each entry together with its outgoing transitions in order to have a copy for each incoming ϵ-edge.

Operations (continuation):

11. **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

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

where for $i \in [n]$, pr_1^i (resp. pr_2^i) denotes the 1st (resp. the 2nd) projection of $E_i \times I_i$.

Effect: In the resulting graph there is a copy of an entry state 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.

An example is shown in Figure 8.

Note: This splitting operation may be easier defined using transition relations. Namely, if T is the transition relation associated to P, then the transition relation associated to $\phi \triangleright P$ is just the relational composite $(\phi|_{\phi^{-1}(I)} \cup Id_S) \circ T$.

The axioms for this new operation are given in Table 2. A comment related to axiom SR^i may be useful. 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 (resp. pr_2^r) the 1st (resp. the 2nd) projection of $\phi^{-1}(r) \times \psi^{-1}(r)$. Finally, take $\psi' = \bigcup_{r \in V} pr_1^r$ and $\phi' = \bigcup_{r \in V} pr_2^r$. These functions fullfill $\phi \circ \psi^{-1} = \psi'^{-1} \circ \phi'$.

After these preliminaries, the axiom corresponding to simulation has a definite phrasing and it is given in Table 3.

Table 2: Axioms for splitting

$S1.$	$\phi \rhd a_q^p$	$=$	if $\phi^{-1}(p) \neq \emptyset$ then $\Leftplus_{r \in \phi^{-1}(p)} a_q^r$ else $\top_q$
$S2.$	$\phi \rhd \bot^p$	$=$	if $\phi^{-1}(p) \neq \emptyset$ then $\Leftplus_{r \in \phi^{-1}(p)} \bot^r$ else $\emptyset$
$S3.$	$\phi \rhd \emptyset$	$=$	$\emptyset$
$S4.$	$\phi \rhd \top_q$	$=$	$\top_q$
$SA.$	$\phi \rhd (f \Leftplus g)$	$=$	$(\phi \rhd f) \Leftplus (\phi \rhd g)$
$SF.$	$\phi \rhd (f \uparrow_q^p)$	$=$	$(\phi_{\mid \{p\} \cup \phi^{-1}(p)\mid} \rhd f) \uparrow_q^p$
$SR^i.$	$\phi \rhd (\psi \lhd f)$	$=$	$\psi' \lhd (\phi' \rhd f)$ 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 3: Axiom for simulation

$SIM.$	$f \rhd (Id_O \cup \phi) = (Id_I \cup \phi) \rhd f' \Rightarrow f \uparrow_A^A = f' \uparrow_B^B$
	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 $I \cup O$; B disjoint of $I' \cup O'$)

Theorem 3.3 *The axioms of graph isomorphism (in Table 1), those for splitting (in Table 2) and SIM are correct and complete for process graphs modulo strong bisimulation.*

Proof: *1) Completeness:* The axioms in Table 2 are sufficient to eliminate every splitting operation. Hence, every expression constructed with atomic actions, constants and operations of alternative composition, feedback, renaming entry states, renaming exit states, and splitting input ports may be brought to a normal form. By Theorem 3.2 two normal form processes are strongly bisimilar iff they may be connected by simulations via functions. Simulation is axiomatized by the SIM rule, hence the axioms are complete.

2) Correctness: It is easy to see that the axioms for splitting are correct in the model of process graphs modulo strong bisimulation. We show in detail that SIM is a sound axiom, too.

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

$$P \rhd (Id_O \cup \phi) \leftrightarrow_\rho (Id_I \cup \phi) \rhd P'$$

for a relation $\rho \subseteq S \times S'$.

Suppose T and T' are the transition relations corresponding to P and P', respectively. By the above note, the transition relation corresponding to the right part is the relational composite $(Id_I \cup \phi \cup Id_{S'}) \circ T'$. From the definition of bisimulation we get

(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 $P \uparrow_A^A \leftrightarrow_{\phi \cup \rho} P' \uparrow_B^B$.

Up to an isomorphism $P \uparrow_A^A$ and $P' \uparrow_B^B$ are the process graphs given by the transition relations $T \subseteq [I \cup (A \cup S)] \times [O \cup (A \cup S)]$ and $T' \subseteq [I \cup (B \cup S')] \times [O \cup (B \cup S')]$. (Notice

that the same transition relation occurs as for P (resp. P') but it used in a different context, i.e. the states in A (resp. B) are considered internal states.)

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

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

Since ϕ is a function, it obeys $Id_A \subseteq \phi \circ \phi^{-1}$ and $\phi^{-1} \circ \phi \subseteq Id_B$, hence the proof is finished. □

A general axiom of this type was introduced in [1] under the name of "functoriality axiom". Arbitrary "morphisms" has been considered there, not functions only. Functoriality axiom for particular classes of finite relations was widely used in various axiomatizations related to flowchart schemes (see [13], for example). The present SIM axiom is related to the functoriality of finite functions. A weaker equational instance of the functoriality of functions is the key axiom of iteration theories [11] and it has been used to get an axiomatisation of synchronization trees in [12].

4 Final Remarks

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 [3] where more operations and results may be found, for instance axiomatisations results for parallel operators.

What we have achieved? The present model is a natural extension of the usual model of single-entry/single-exit process graphs. While preserving the properties of the original model, the present extension allows to model a natural looping operator (feedback) and adds more expressivity.

On the other hand, this model fails to be a full model for the algebra of flownomials due to the lack of identities (= empty transitions). We have adopted this restriction in view of the subtle behaviour of the empty transitions when combining with parallel operators. Actually, in the algebra of flownomials there is much more freedom in the sense that arbitrary semantical theories may be taken as models for the connections between the atomic elements. A general goal may be to get a similar freedom here, i.e. to get a unified theory which is parametrized by the theory of connecting morphisms. In this view, the present paper contains the study of process graphs in the simplest case, i.e. the connections are given by the empty constants $\emptyset, \bot, \top$. The construction of a unified theory of the process graphs connected via empty transitions, τ (silent) steps, or more complicated processes will require more effort.

A different approach to mixing process algebra and the algebra of flownomials has been proposed in [7] in order to model synchronous and asynchronous dataflow networks. In the dataflow network setting the flownomials operators are used to describe the architecture of the network. The resulting network algebra model preserves the core of the algebra of flownomials, in particular it gives meaning to the identities via the dataflow semantics for wires. The rôle of the process algebra operators is to give a detailed description of the activity of the cells, as well as the wires.

References

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

[2] 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, 1994.

[3] J.C.M. Baeten, J.A. Bergstra and Gh. Stefanescu. Process algebra with feedback. In: *Modal logic and process algebra*, (Eds. A. Ponse, M. de Rijke and Y. Venema), 13–38. CSLI Publications no. 53, Stanford, 1995.

[4] J.C.M. Baeten and W.P. Weijland. *Process algebra*. Cambridge University Press, 1990.

[5] J.A. Bergstra, I. Bethke and A. Ponse. Process algebra with iteration and nesting. *The Computer Journal*, 37(4):243–258, 1994.

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

[7] J.A. Bergstra, C.A. Middelburg, and Gh. Ştefănescu. Network algebra for synchronous and asynchronous dataflow. Report P9508, Programming Research Group, University of Amsterdam, October 1995.

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

[9] J.A. Bergstra and Gh. Ştefănescu. Processes with multiple entries and exits. In *Proceedings FCT'95*, 136–145. LNCS 965, Springer Verlag, 1995.

[10] J.A. Bergstra and Gh. Ştefănescu. Bisimulation is two-way simulation. *Information Processing Letters*, 52:285–287, 1994.

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

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

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

[14] 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.

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

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

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

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

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

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

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

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

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

[24] Gh. Stefănescu. Algebra fo flownomials. Part I: Binary flonomials; basic theory. Report TUM-I9437, Institute for Informatics, Technical University Munich, October 1994.