Concurrency: a relational approach

Ridha Khédri[1] and Jules Desharnais[2]

[1] Department of Computing and Software, McMaster University, Hamilton, ON, L8S 4K1 Canada,
E-mail: Khedri@mcmaster.ca
[2] Département d'informatique, Université Laval, Québec, QC, G1K 7P4 Canada,
E-mail: Jules.Desharnais@ift.ulaval.ca

Abstract. We model processes by means of a mathematical entity that we call a *relational process*. This model describes a process as an open system from which the description of the process as a closed system can be easily obtained. Also, it represents not only the actions of the process but also the resources needed to accomplish its behaviour. Using this model, we first define two operators. Each of these represents an extreme perception of concurrency. One, the *interleaved parallel composition operator*, reduces concurrency to interleaving and the other, the *maximal totally synchronous parallel composition operator*, reduces concurrency to a totally synchronous behaviour. Second, by combining these operators, we define the *maximal true-concurrency composition operator*, which is an operator expressing true concurrency. When many processes interfere on the same resource in order to modify it, each in its way, the two maximal operators express this situation by letting the final value of the variable modelling this resource be indeterminate. So, they allow the detection of interferences between processes. We present some of the properties of these operators.
Keywords: Concurrency, relational process, interleaved parallel composition, synchronous parallel composition, true concurrency.

1 Introduction

Since the work of Carl Petri on Petri nets (1962 [19]), which is the first general theory about concurrency, many models of concurrency were introduced. These models vary depending on the formalism used and on the simplifications made in the description of this notion.

The best-known methods dealing with concurrency are *Petri nets* [20], the *Calculus of Communicating Systems* (CCS) [15] and the *Communicating Sequential Processes* (CSP) [11]. Petri nets emphasize the graphical representation of real concurrency [16]. But, except by using the semantics of nets, there is no method allowing the verification or the calculation of the components of concurrent systems. As regards CCS and CSP, they emphasize structure and abstraction aspects [16]. Also, true-concurrency, in the two approaches, is neglected in favour of an interleaving model with synchronised communications between processes. Actual concurrent systems are usually composed of several independent processors, each of them executing a process. In such systems, the execution of actions in different processors sometimes overlaps and sometimes interleaves. This is what is called *real concurrency* or *true concurrency*.

Methods like CSP and CCS deal with *uniform concurrency* only. This means that processes are studied, performing actions $a, b, c, \ldots$ which are not subject to further investigations. So it remains unspecified if these actions are in fact assignments to variables or the falling of dominoes or other actions [22]. Also, the resources needed to accomplish these actions remain unspecified. Now, most software problems result from erroneous descriptions

of the intended behaviour, and conflicts about resources in concurrent systems only complicate the situation. The term *uniform concurrency* was used for the first time to qualify such methods by De Bakker *et al.* in [5].

In this paper, we present a relational model for processes and we introduce various operators expressing different perceptions of concurrency: interleaved, totally synchronous, and true concurrency. In the next section, we present the relational background of this paper. Next, we present the interleaved parallel composition. After, we present the totally synchronous parallel composition followed by the true-concurrency composition. Finally, we conclude and we point to our future work.

2 Background

Relations are sets and, as such, we can apply to them the usual set operators. However, to avoid confusion between sets (other than relations) and relations, we use different operator symbols for relations and sets other than relations. For example, set union is denoted by $\cup$ while relation union is denoted by $\sqcup$. Union ($\sqcup$), meet ($\sqcap$) and complementation ($\overline{\ }$) are operators applicable to relations. The inclusion ($\sqsubseteq$) confers an order to relations. Other operations are defined on relations, such as the following ones (the symbol $\triangleq$ is equality by definition).

Definition 1. Let $R \subseteq X \times Z$ and $S \subseteq Z \times Y$ be two relations.

(a) The *converse* of relation R, noted $R^\smile$, is defined as $R^\smile \triangleq \{(x,z) \mid (z,x) \in R\}$.
(b) The *relational composition* of R and S, noted $R;S$, is defined as

$$R;S \triangleq \{(x,y) \mid \vee\,(z \;:\; z \in Z \;:\; (x,z) \in R \;\wedge\; (z,y) \in S)\}.$$

■

We remark that $R;S \subseteq X \times Y$. The precedence of the relational operators, from highest to lowest, is: $\overline{\ }, ^\smile, ;, \sqcap, \sqcup$ and the precedence of the set operators is $\overline{\ }, \cap, \cup$. The *empty relation*, the *universal relation* and the *identity relation* are denoted by $\bot\!\!\!\bot, \top\!\!\!\top$ and $\mathbb{I}$, respectively.

Some of the usual rules of the calculus of relations follow (see [4] [21]).

Theorem 2. *Let P, Q and R be relations. Then*

1. $P;(Q \sqcup R) = P;Q \sqcup P;R$,
2. $(P \sqcup Q);R = P;R \sqcup Q;R$,
3. $P \sqcap (Q \sqcup R) = P \sqcap Q \sqcup P \sqcap R$,
4. $Q \sqsubseteq R \Rightarrow P;Q \sqsubseteq P;R$,
5. $P \sqsubseteq Q \Rightarrow P;R \sqsubseteq Q;R$,
6. $Q \sqsubseteq R \iff Q^\smile \sqsubseteq R^\smile$.

For all the remainder of the paper, we fix a set of indexes $\mathcal{U}$ and a family of domains $(i \;:\; i \in \mathcal{U} \;:\; D_i)$. The set $\mathcal{U}$ is considered as a universal set in a given context.

Definition 3. Let $I \subseteq \mathcal{U}$, $D_I \triangleq \prod(i \;:\; i \in I \;:\; D_i)$. ■

The set D_I can be seen as the set of all functions

$$f : I \longrightarrow \cup(i \ : \ i \in I \ : \ D_i \,)$$

such that $\wedge\,(i \ : \ i \in I \ : \ f(i) \in D_i \,)$. This can be written more formally as

4. $f \in D_I \iff (f : I \longrightarrow \cup(i \ : \ i \in I \ : \ D_i \,)) \ \wedge \ (\ \wedge\,(i \ : \ i \in I \ : \ f(i) \in D_i \,))$, ■

where $I \subseteq \mathcal{U}$ and f is a function.

Definition 5. Let $f \in D_I$. The *restriction of the function* f to a subset $K \subseteq I$ is the function element of D_K given by $\mathcal{I}_K ; f$, where $\mathcal{I}_K$ is the identity over K, *i.e.*
$\mathcal{I}_K = \{(i,j) \mid i,j \in K \ \wedge \ i = j \,\}$. ■

We use different fonts to distinguish between the identity over a subset of indexes K, noted $\mathcal{I}_K$, and the identity over D_K, noted $\mathbb{I}_K$. For the latter, we simply write $\mathbb{I}$ when the context allows to know K. It is easy to see that, for I and J subsets of $\mathcal{U}$, we have $\mathcal{I}_\emptyset = \perp\!\!\!\perp$, $\mathcal{I}_I \sqcup \mathcal{I}_J = \mathcal{I}_{I \cup J}$, and $\mathcal{I}_I ; \mathcal{I}_J = \mathcal{I}_I \sqcap \mathcal{I}_J = \mathcal{I}_{I \cap J}$. The following definition introduces projections.

Definition 6. Let $I \subseteq \mathcal{U}$. The *projection* from $D_\mathcal{U}$ onto D_I, noted Π_I, is defined as:

$$\Pi_I \triangleq \{(f,g) \mid f \in D_\mathcal{U} \ \wedge \ g = \mathcal{I}_I ; f \,\}.$$

■

From this definition, it ensues that, for $I \subseteq \mathcal{U}$, $\Pi_\mathcal{U} = \mathbb{I}$, $\breve{\Pi_I} ; \Pi_I = \mathbb{I}_I$, and Π_I is total.

Definition 7. Let $I \subseteq \mathcal{U}$, $\delta_I \triangleq \Pi_I ; \breve{\Pi_I}$. The relation δ_I is said to be an I-cylindrification relation. ■

Example 8. Let $\mathcal{U} = \{x, y, z\}$ and $D_{\{x\}} = D_{\{y\}} = D_{\{z\}} = \mathbb{N}$ (the natural numbers). Then

$$\Pi_{\{x\}} = \{((x,y,z), x') \mid x' = x\} \text{ and } \delta_{\{x\}} = \{((x,y,z),(x',y',z')) \mid x' = x\},$$

where we use the well-known correspondence between the tuple notation and the function notation (*e.g.*, using the ordering x, y, z, the tuple $(3,4,5)$ corresponds to the function $\{(x,3),(y,4),(z,5)\}$).

Projecting from $D_\mathcal{U}$ onto D_I followed by the inverse operation comes down to preserve uniquely the components given by the family of indexes I. This is what the relation δ_I expresses. The effect of this relation is similar to what is expressed in cylindric algebras [1] [9] [10] by some unary operators called *cylindrification operators*. It is mentioned in [9, page 1] that this terminology is borrowed from analytic geometry. We note that if we were to stay close to the cylindric algebras terminology, we should call δ_I an $\overline{I}$-cylindrification relation instead of an I-cylindrification relation; however, for convenience, we adopt the term I-cylindrification. Composed to a relation R, the I-cylindrification relation introduces an existential quantification. We say that we accomplish a *right I-cylindrification* on a relation R in the case of $R ; \delta_I$ and a *left I-cylindrification* on R in the case of $\delta_I ; R$. Either in the literature [6] [10] or in the course of our study of concurrent systems, we saw that the cylindrification relations introduced by definition 7 have very interesting properties that could simplify the notation and the proofs. Some of these properties follow; their proof can be found in [13].

Theorem 9. *Let P, Q, R be relations and let I and J be sets of indexes. Then,*

1. $\delta_I ; \delta_J = \delta_J ; \delta_I = \delta_{I \cap J}$,
2. $\delta_\emptyset = \top$,
3. $\delta_{\mathcal{U}} = \mathbb{I}$,
4. $\Pi^\smile_I ; \delta_I = \Pi^\smile_I$,
5. $\delta_I ; \delta_I = \delta_I$,
6. $\delta^\smile_I = \delta_I$,
7. $\delta_I \sqcap \delta_{\overline{I}} = \mathbb{I}$,
8. $\mathbb{I} \sqsubseteq \delta_I$ *i.e.* δ_I *is reflexive,*
9. δ_I *is difunctional,*
10. δ_I *is an equivalence relation,*
11. $I \subseteq J \Rightarrow (Q ; \delta_I \sqcap R) ; \delta_J = Q ; \delta_I \sqcap R ; \delta_J$,
12. $I \subseteq J \Rightarrow \delta_J ; (\delta_I ; Q \sqcap R) = \delta_I ; Q \sqcap \delta_J ; R$,
13. δ_I *is total and surjective.*

■

2.1 (J, K)-determined relations

Definition 10. Let R be a relation and let I and J be two subsets of $\mathcal{U}$. The relation R is said to be *(J, K)-determined* iff $R = \delta_J ; R ; \delta_K$. ■

In other words, given two subsets of $\mathcal{U}$, J and K, a relation R is said to be (J, K)-determined if and only if the left J-cylindrification with the right K-cylindrification of R do not lose any of the information carried by R. If we see R as an input-output relation, this means that all the information carried by R concerns input components that are elements of J and output components that are elements of K.

By the following proposition, we give some of the properties of (J, K)-determined relations.

Proposition 11. *Let P and Q be two relations such that $P = \delta_J ; P ; \delta_K$ and $Q = \delta_L ; Q ; \delta_M$. We have*

(a) $P \sqcup Q = \delta_{J \cup L} ; (P \sqcup Q) ; \delta_{K \cup M}$,
(b) $P \sqcap Q = \delta_{J \cup L} ; (P \sqcap Q) ; \delta_{K \cup M}$,
(c) $P ; \delta_{\overline{K \cap L}} = P ; \delta_{K \cap \overline{L}} = P ; \delta_{\overline{L}}$.

Proof.

(a) It follows from Theorem 2(1, 2), and Proposition 9(1).
(b) It results from Proposition 9(1), Proposition 9(11), and Proposition 9(12).
(c) It follows from Proposition 9(1), the distributivity of $\cap$ over $\cup$, and Proposition 9(5).

The following proposition gives a connection between cylindrification relations, (I, J)-determined relations and partial identities.

Proposition 12. *Let P and Q be relations such that $P = \delta_I ; P ; \delta_I \sqcap \mathbb{I}$ et $Q = \delta_J ; Q ; \delta_J \sqcap \mathbb{I}$. We have*

$$P \sqcap Q = \delta_{I \cup J} ; (P \sqcap Q) ; \delta_{I \cup J} \sqcap \mathbb{I}.$$

Proof. It follows from Proposition 11(b) .

3 Relational processes

Sequential systems do not interfere with other systems which could block them an acces to a resource or could modify the result of a job that they are doing. All the accesses to the resources that they use is done either before they start or after they finish. Consequently, when modeling sequential programs (which are instances of sequential systems), neither the atomicity of program actions nor the distinction between read and modified variables or between controlled and non-controlled variables is needed. A system which interacts interactively with its environment (with the environment possibly acting on the system before it stops) is said to be a *reactive system*. A model of a reactive system should make explicit what the resources used by each action are. Also, the modeling should refer to the atomicity of actions [3]. When modeling a given process, we group the resources of the process and its environment in three categories: the resources from where the process draws the needed raw material, the resources that it could modify and the resources which it does not use. Certainly a model that takes into consideration the atomicity of actions and the resources used contains some useless information for the study of sequential systems. However, this information becomes useful when the system acts concurrently with other systems and its environment.

In this section, we propose a relational model of processes. This model can be used either to represent stand-alone systems without any kind of interaction, or to represent systems acting with the cooperation of other systems to accomplish a work. Our mathematical entity is called a *relational process* and is introduced by the following definition.

Definition 13. A *relational process* $\mathcal{P}$ is a 5-tuple

$$((J,K \ : \ J,K \in \mathcal{U} \ : \ P_{_{JK}}), \alpha_P, E_P, \omega_P, F_P)$$

satisfying

(a) $P_{_{JK}} = \delta_{_J} ; P_{_{JK}} ; \delta_{_K}$
(b) $\alpha_P = \delta_{_{E_P}} ; \alpha_P ; \delta_{_{E_P}} \sqcap \mathbb{I}$
(c) $\omega_P = \delta_{_{F_P}} ; \omega_P ; \delta_{_{F_P}} \sqcap \mathbb{I}$

■

The 5-tuple defining a relational process is formed by

- a family of (J, K)-determined relations. Each of these relations describes the behaviour of the part of the system which reads its values from the variables given by the set of indexes J and which writes the result of its processing on the variables indicated by the set K.
- an input relation α_P which is included in the identity and is also (E_P, E_P)-determined. This relation characterizes the states from which the process can be activated. To sum up, α_P characterizes a subset of input states.
- a set E_P of indexes; these indexes are those over which the input relation may impose constraints (by clause (b) of the definition).

- the output relation ω_P which is included in the identity and is also (F_P, F_P)-determined. This relation characterizes the states in which the process may stop. Thus, ω_P characterizes a subset of output states.
- a set F_P of indexes; these indexes are those over which the output relation may impose constraints (by clause (c) of the definition).

In the remainder of this paper, a relational process will be indicated by calligraphic capital letters, such as $\mathcal{P}$ in Definition 13. The relations in the first component of the 5-tuple (P_{JK} in Definition 13) are denoted by standard font capital letters and they are indexed by the set of the read resources, followed by the set of the (possibly) modified resources. The other four components of the relational process ($\alpha_P, E_P, \omega_P, F_P$ in Definition 13) are indexed by the same capital letter which denotes the process, but in standard font. From now on, to lighten the notation, we will write by $(J, K :: P_{JK})$ instead of $(J, K : J, K \in \mathcal{U} : P_{JK})$.

Example 14. Let $\mathcal{P}$ and $\mathcal{Q}$ be two relational processes such that

$$\mathcal{P} = (\{P_{\{p\}\{p,x\}}, P_{\{p,x,y\}\{p,y\}}\}, \alpha_P, \{p, y\}, \perp\!\!\!\perp, \emptyset)$$

and

$$\mathcal{Q} = (\{Q_{\{c,y\}\{c,y,z\}}, Q_{\{c,t,z\}\{c,t\}}\}, \alpha_Q, \{c, y\}, \perp\!\!\!\perp, \emptyset),$$

where

$$P_{\{p\}\{p,x\}} = \{p = 1 \ \wedge \ p' = 2 \ \wedge \ x' = 1\},$$

$$P_{\{p,x,y\}\{p,y\}} = \{p = 2 \ \wedge \ p' = 1 \ \wedge \ y < N \ \wedge \ y' = y + x\}, \quad \alpha_P = \{p = 1 \ \wedge \ y = 0\} \sqcap \mathbb{I},$$

$$Q_{\{c,y\}\{c,y,z\}} = \{c = 1 \ \wedge \ c' = 2 \ \wedge \ y > 0 \ \wedge \ y' = y - 1 \ \wedge \ z' = 1\},$$

$$Q_{\{c,t,z\}\{c,t\}} = \{c = 2 \ \wedge \ c' = 1 \ \wedge \ t' = f(z, t)\}, \text{ and } \alpha_Q = \{c = 1 \ \wedge \ y = 0\} \sqcap \mathbb{I}.$$

We remark that $Q_{\{c,y\}\{c,y,z\}}$, for example, describes the part of a $\mathcal{Q}$ which reads its information from the resources indicated by $\{c, y\}$ and which modifies the resources indicated by $\{c, y, z\}$. We assume that the two relational processes are intended to run forever, so that their output relation is the empty relation. The two processes are graphically illustrated in Figure 1. Each node is labelled by a predicate characterizing a set of states. The nodes labelled by $p = 1$ and $c = 1$ are initial nodes. These are pointed to by a small arrow and labelled by a predicate constraining the values of some variables at the initiation of the system. These values are taken from the input relation. Each arrow of the transition systems is labeled with a triplet formed by the set of read variables, by a predicate relating primed and unprimed variables (except those present in the nodes), and the set of potentially modified variables. Since these transition systems are used here only to illustrate examples visually, we do not describe further the connection between them and the formal description of a relational process.

Definition 15. Let $\mathcal{P} = (\ (J, K :: P_{JK}), \alpha_P, E_P, \omega_P, F_P)$ be a relational process. The *associated relation* of the relational process $\mathcal{P}$ is the relation $P \triangleq \sqcup(J, K :: P_{JK} \sqcap \delta_{\overline{K}})$. ∎

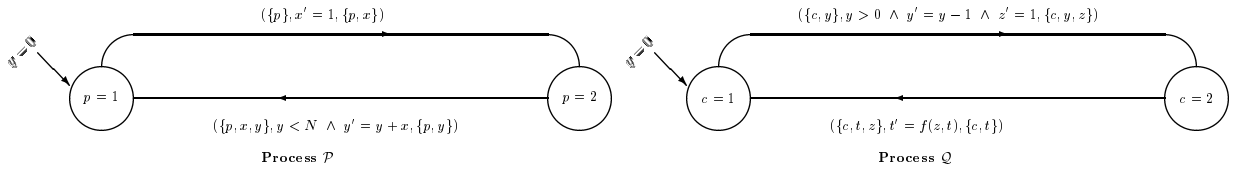


Fig. 1. A graphical illustration of $\mathcal{P}$ and $\mathcal{Q}$

The associated relation is a union of relations. Each relation is obtained from a member belonging to the family of relations given in the relational process expression, and this by imposing that the variables not intended to be modified (*i.e.* indicated by $\overline{K}$) are preserved. This associated relation is a good approximation of the specification of a system and this under the hypothesis that this system is a *closed system*. A system is said to be closed if it does not have an environment; otherwise, the system is said to be *open*. If a system is closed, a variable that is modified is necessarily modified by the system (no interference).

Example 16. Consider the relational process $\mathcal{P}$ of Example 14. If we take $\mathcal{U} = \{c, p, x, y, z, t\}$, then the relation associated to $\mathcal{P}$, called P, is obtained as follows

$$
\begin{array}{ll}
 & P \\
= & \quad \langle \text{ Definition 15 } \rangle \\
 & P_{\{p\}\{p,x\}} \sqcap \delta_{\underline{\quad}}^{\quad} \sqcup P_{\{p,x,y\}\{p,y\}} \sqcap \delta_{\underline{\quad}}^{\quad} \\
= & \quad \langle \text{ all calculations done } \rangle \\
 & \{ \quad c' = c \ \wedge \ p = 1 \ \wedge \ p' = 2 \ \wedge \ x' = 1 \ \wedge \ y' = y \ \wedge \ z' = z \ \wedge \ t' = t \\
 & \quad \vee \ c' = c \ \wedge \ p = 2 \ \wedge \ p' = 1 \ \wedge \ x' = x \ \wedge \ y < N \ \wedge \ y' = y + x \ \wedge \ z' = z \ \wedge \ t' = t \ \}
\end{array}
$$

Proposition 17. *Let $*$ and $\star$ be two binary relational operators. Let $\bullet$ be a binary set operator. Let $(\,(J, K \ :: \ P_{JK}\,), \alpha_P, E_P, \omega_P, F_P)$ and $(\,(J, K \ :: \ Q_{JK}\,), \alpha_Q, E_Q, \omega_Q, F_Q)$ be two relational processes. Let $\wr$ an operator over relational processes such that*

$$\mathcal{P} \wr \mathcal{Q} \triangleq (\,(J, K \ :: \ P_{JK} * Q_{JK}\,), \alpha_P \star \alpha_Q, E_P \bullet E_Q, \omega_P \star \omega_Q, F_P \bullet F_Q).$$

(a) If $*$, $\star$ and $\bullet$ are commutative, then $\wr$ is commutative.
(b) If $*$, $\star$ and $\bullet$ are associative, then $\wr$ is associative.

Proof.

$$
\begin{array}{lll}
\text{(a)} & \mathcal{P} \wr \mathcal{Q} & \\
 & = & \quad \langle \text{ hypothesis (definition of } \wr) \rangle \\
 & & (\,(J, K \ :: \ P_{JK} * Q_{JK}\,), \alpha_P \star \alpha_Q, E_P \bullet E_Q, \omega_P \star \omega_Q, F_P \bullet F_Q) \\
 & = & \quad \langle \ *, \star \text{ et } \bullet \text{ are commutative } \rangle \\
 & & (\,(J, K \ :: \ Q_{JK} * P_{JK}\,), \alpha_Q \star \alpha_P, E_Q \bullet E_P, \omega_Q \star \omega_P, F_Q \bullet F_P) \\
 & = & \quad \langle \text{ hypothesis (definition of } \wr) \rangle \\
 & & \mathcal{Q} \wr \mathcal{P}
\end{array}
$$

(b) $\mathcal{P} \wr (\mathcal{Q} \wr \mathcal{R})$

$=$ $\langle$ hypothesis (definition of $\wr$) $\rangle$

$\mathcal{P} \wr (\,(J, K \;::\; Q_{JK} * R_{JK}\,), \alpha_Q \star \alpha_R, E_Q \bullet E_R, \omega_Q \star \omega_R, F_Q \bullet F_R)$

$=$ $\langle$ hypothesis (definition of $\wr$) & $*$, $\star$ and $\bullet$ are associative $\rangle$

$(\,(J, K \;::\; P_{JK} * Q_{JK} * R_{JK}\,), \alpha_P \star \alpha_Q \star \alpha_R, E_P \bullet E_Q \bullet E_R,$
$\omega_P \star \omega_Q \star \omega_R, F_P \bullet F_Q \bullet F_R)$

$=$ $\langle$ hypothesis (definition of $\wr$) & $*$, $\star$ and $\bullet$ are associative $\rangle$

$(\,(J, K \;::\; P_{JK} * Q_{JK}\,), \alpha_P \star \alpha_Q, E_P \bullet E_Q, \omega_P \star \omega_Q, F_P \bullet F_Q) \wr$
$(\,(J, K \;::\; R_{JK}\,), \alpha_R, E_R, \omega_R, F_R)$

$=$ $\langle$ hypothesis (definition of $\wr$) $\rangle$

$(\mathcal{P} \wr \mathcal{Q}) \wr \mathcal{R}$

■

4 Interleaved parallel composition

The basic notion of modeling concurrency by "interleaving" was first used by Dijkstra [7], who also coined the term "interleaving" [8]. In this model, an execution of a system that contains two parallel processes is represented by an interleaving of the atomic instructions of the participating processes. This model reduces concurrency to nondeterminism due to the many possible interleavings. In each possible execution, only one action from one of the involved processes is performed at a time. With this requirement, this model provides a higher degree of protection from interference than is avaible when overlapping is allowed.

The following definition introduces the interleaved parallel composition operator.

Definition 18. Let $\mathcal{P}$ and $\mathcal{Q}$ be two relational processes such that

$$\mathcal{P} = (\,(J, K \;::\; P_{JK}\,), \alpha_P, E_P, \omega_P, F_P) \text{ and } \mathcal{Q} = (\,(J, K \;::\; Q_{JK}\,), \alpha_Q, E_Q, \omega_Q, F_Q).$$

The *interleaved parallel composition* of $\mathcal{P}$ and $\mathcal{Q}$, noted $\mathcal{P} \curlywedge \mathcal{Q}$, is defined as

$$\mathcal{P} \curlywedge \mathcal{Q} \triangleq (\,(J, K \;::\; P_{JK} \sqcup Q_{JK}\,), \alpha_P \sqcap \alpha_Q, E_P \cup E_Q, \omega_P \sqcap \omega_Q, F_P \cup F_Q).$$

■

From Definition 18, we note that an element R_{JK} of the family of relations of the relational process $\mathcal{P} \curlywedge \mathcal{Q}$ is the union of the corresponding (J, K)-determined relations of the argument relational processes. Also, a state is an input (output) state of $\mathcal{P} \curlywedge \mathcal{Q}$ if it is an input (output) state of one or the other argument process.

4.1 Properties

First, according to Proposition 11(a), Proposition 12, and Definition 13, we conclude that $\mathcal{P} \curlywedge \mathcal{Q}$ is indeed a relational process.

Proposition 19. *The operator* $\natural$ *is commutative and associative.*

Proof. The operators $\cup$, $\sqcup$ and $\sqcap$ are commutative and associative. Hence, according to Proposition 17(a), we have the commutativity of $\natural$ and, according to Proposition 17(b), we have the associativity of $\natural$. ■

Proposition 20. *For any relational process* $\mathcal{P}$*, we have* $\mathcal{P} \natural \mathcal{P} = \mathcal{P}$*.*

Proof. This follows from the idempotence of $\sqcup$, $\sqcap$, and $\cup$. ■

Proposition 21 connects the notion of associated relation to the interleaved parallel composition operator.

Proposition 21. *Let* $\mathcal{P}$*,* $\mathcal{Q}$ *and* $\mathcal{R}$ *be relational processes such that* $\mathcal{R} = \mathcal{P} \natural \mathcal{Q}$*. Let* P*,* Q *and* R *the associated relations to* $\mathcal{P}$*,* $\mathcal{Q}$ *and* $\mathcal{R}$*, respectively. We have* $R = P \sqcup Q$*.*

Proof. It follows from Theorem 2(3).

■

Consider the 5-tuple

22. $\mathcal{O}_e \triangleq (\,(J,K \;::\; \top_{JK}\,), \bot\!\!\!\bot, \mathcal{U}, \bot\!\!\!\bot, \mathcal{U})$.

As defined, $\mathcal{O}_e$ is a relational process. From its family of (universal) relations, we see that $\mathcal{O}_e$ models a particular process which can do anything and which controls all the resources. Since its input (output) relation is empty and since the set E (F) is the universal set $\mathcal{U}$, the subset of states where the process $\mathcal{O}_e$ may start (stop) its behaviour is empty. This process is an absorbing element for interleaved concurrent composition, which is expressed by the following proposition.

Proposition 23. *Let* $\mathcal{P}$ *be a relational process. We have* $\mathcal{O}_e \natural \mathcal{P} = \mathcal{P} \natural \mathcal{O}_e = \mathcal{O}_e$*.*

Proof. This results from Definition 13, the commutativity of $\natural$ (*i.e.* Proposition 19), $P \sqcup \top = \top$, $P \sqcap \bot\!\!\!\bot = \bot\!\!\!\bot$, $E_P \subseteq \mathcal{U}$ and $F_P \subseteq \mathcal{U}$. ■

Consider the 5-tuple

24. $\mathcal{Z}_e \triangleq (\,(J,K \;::\; \bot\!\!\!\bot_{JK}\,), \mathbb{I}, \emptyset, \mathbb{I}, \emptyset)$.

As defined, $\mathcal{Z}_e$ is a relational process and it represents a process which is not doing anything and which does not control any resource. This relational process is a neutral element for interleaved concurrent composition.

Proposition 25.

$$\mathcal{Z}_e \natural \mathcal{P} = \mathcal{P} \natural \mathcal{Z}_e = \mathcal{P}.$$

Proof. The result follows from Proposition 19 (*i.e.* $\natural$ is commutative), Definition 13, $P \sqcup \bot\!\!\!\bot = P$, $\alpha_P \sqsubseteq \mathbb{I}$, $\omega_P \sqsubseteq \mathbb{I}$, and $S \cup \emptyset = S$. ■

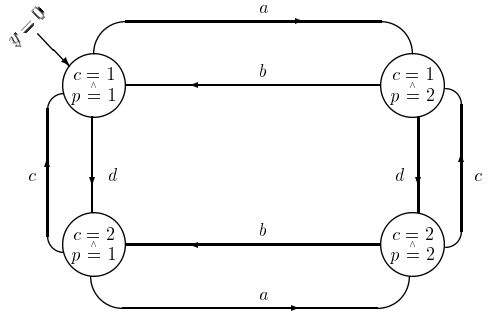
Example 26. We take the two relational processes of Example 14. Let $\mathcal{R}_e \triangleq \mathcal{P} \upharpoonright \mathcal{Q}$. Then,

$$\begin{aligned}&\mathcal{R}_e\\ =\ &\langle \text{ Definition 18 } \ \&\ \ \{p,y\}\cup\{c,y\}=\{p,c,y\}\ \ \&\ \ \bot\!\!\!\bot\sqcap\bot\!\!\!\bot=\bot\!\!\!\bot\ \ \&\ \ \emptyset\cap\emptyset=\emptyset\ \rangle\\ &(\{P_{\{p\}\{p,x\}},P_{\{p,x,y\}\{p,y\}},Q_{\{c,y\}\{c,y,z\}},Q_{\{c,t,z\}\{c,t\}}\},\alpha_P\sqcap\alpha_Q,\{p,c,y\},\bot\!\!\!\bot,\emptyset).\end{aligned}$$

The input relation of $\mathcal{R}_e$ is

$$\alpha_R=\alpha_P\sqcap\alpha_Q=\{c=1\ \wedge\ p=1\ \wedge\ y=0\}\sqcap\mathbb{I}.$$

The mathematical model of the system formed by the interleaving of the two processes modeled by $\mathcal{P}$ and $\mathcal{Q}$ is then $\mathcal{R}_e$. This system is graphically illustrated in Figure 2.



$a \triangleq (\{p\}, x'=1, \{p,x\})$
$b \triangleq (\{p,x,y\}, y<N\ \wedge\ y'=y+x, \{p,y\})$
$c \triangleq (\{c,y\}, y>0\ \wedge\ y'=y-1\ \wedge\ z'=1, \{c,y,z\})$
$d \triangleq (\{c,t,z\}, t'=f(z,t), \{c,t\})$

Fig. 2. A graphical illustration of the interleaved parallel composition of $\mathcal{P}$ and $\mathcal{Q}$

5 Totally synchronous parallel composition

The totally synchronous parallel composition expresses a concurrency mechanism where, if there is a system involved in a cooperation and executing an atomic action, then each one of the other participating systems has to take simultaneously an atomic action. The free product of transition systems [2] expresses a similar semantics of concurrency. An action is said to be atomic if it is not a sequence of actions. In general, an action formed by many atomic actions performed simultaneously is an atomic action. The representation of a communication action as a pair of complementary atomic actions is a particular case of this rule. This manner of seeing many atomic actions performed simultaneously as an atomic action is also used in the synchronous calculus [15].

Concerning the simultaneous modification of a resource by many processes, we consider two possibilities.

– The first one could be called "progress under full agreement": if the processes involved agree on the same modification, the modification is done, otherwise, the processes are blocked (no transition is possible). This is expressed by what is called in [13] the minimal totally synchronous parallel composition operator.

– The second could be called "anarchic progress under resource sharing": if the processes involved do not share any resource in a modification, the modification is done, otherwise, the processes may take a transition, but the final value of the variable denoting the shared resource is undetermined (is arbitrary). This is expressed by the maximal totally synchronous parallel composition operator presented in Definition 27 below. In this paper we present only this operator.

5.1 Maximal totally synchronous parallel composition

The term "maximal" qualifying here this parallel composition is not related to what we find in [12] [14] under the name *Maximal Parallelism*. In the maximal parallelism, all the communication events appear as soon as possible and no process is let in a waiting state when it could do some action. It is not the semantics associated to what we present in the next definition.

Definition 27. Let $\mathcal{P}$ and $\mathcal{Q}$ be two relational processes such that

$$\mathcal{P} = (\,(J, K \;::\; P_{JK}\,), \alpha_P, E_P, \omega_P, F_P) \text{ and } \mathcal{Q} = (\,(J, K \;::\; Q_{JK}\,), \alpha_Q, E_Q, \omega_Q, F_Q).$$

The *maximal totally synchronous parallel composition* of $\mathcal{P}$ and $\mathcal{Q}$, noted $\mathcal{P} \uparrow\uparrow \mathcal{Q}$, is defined as

$$\mathcal{P} \uparrow\uparrow \mathcal{Q} \triangleq (\,(J, K \;::\; R_{JK}\,), \alpha_P \sqcap \alpha_Q, E_P \cup E_Q, \omega_P \sqcap \omega_Q, F_P \cup F_Q)$$

where

$$R_{JK} = \sqcup(J_P, J_Q, K_P, K_Q \;:\; J = J_P \cup J_Q \;\wedge\; K = K_P \cup K_Q \;:\; P_{J_P K_P};\delta_{\frac{}{K_P \cap K_Q}} \sqcap Q_{J_Q K_Q};\delta_{\frac{}{K_P \cap K_Q}}).$$

■

In the definition of R_{JK}, the variables representing the resources that could provoque a conflict between the two processes, at the time of a modification, are given by $K_P \cap K_Q$. When there is a conflict, the operator $\uparrow\uparrow$ assigns an indeterminate final value to these variables. For instance, take $P_{\{\}\{x\}} = \{x' > 0\}$ and $Q_{\{x,y\}\{x,y\}} = \{x' = 4 \;\wedge\; y' = x + y\}$. The relation describing the actions obtained by the maximal totally synchronous parallel composition of the actions given by $P_{\{\}\{x\}}$ and $Q_{\{x,y\}\{x,y\}}$ is the relation $R_{\{x,y\}\{x,y\}} = \{y' = x + y\}$. Note that in spite of the fact that x is controlled in the modification, its final value is arbitrary.

The following equalities hold. They can be used to derive expressions equivalent to that of relation R_{JK} in Definition 27.

28. $P;\delta_{\frac{}{K_P \cap (K_Q \cup K_R)}} \sqcap (Q;\delta_{\frac{}{K_Q \cap K_R}} \sqcap R;\delta_{\frac{}{K_Q \cap K_R}});\delta_{\frac{}{K_P \cap (K_Q \cup K_R)}}$

$=$

$P;\delta_{\frac{}{K_Q \cup K_R}} \sqcap Q;\delta_{\frac{}{K_P \cup K_R}} \sqcap R;\delta_{\frac{}{K_P \cup K_Q}}$

$=$

$(P;\delta_{\frac{}{K_P \cap K_Q}} \sqcap Q;\delta_{\frac{}{K_P \cap K_Q}});\delta_{\frac{}{(K_P \cup K_Q) \cap K_R}} \sqcap R;\delta_{\frac{}{(K_P \cup K_Q) \cap K_R}}$

To be able to claim that the result of the maximal totally synchronous parallel composition of two relational processes is a relational process, we need to show that the relations R_{JK} of Definition 27 are (J, K)-determined. This follows from Definition 27, Theorem 2(1, 2), Proposition 9(12), Proposition 9(1) and Proposition 9(11). Also, using Proposition 12, we can prove that the input and the output relations satisfy both conditions of the definition of a relational process (*i.e.* Definition 13(b, c)). Hence, the maximal totally synchrounous parallel composition of two relational processes is a relational process.

Proposition 29. *The operator $\Uparrow\!\Uparrow$ is commutative and associative.*

Proof. The commutativity of $\Uparrow\!\Uparrow$ is a consequence of the commutativity of $\cup, \cap$, and $\sqcap$. The proof of associativity follows.

$$
\begin{array}{ll}
 & \mathcal{P} \Uparrow\!\Uparrow (\mathcal{Q} \Uparrow\!\Uparrow \mathcal{R}) \\
= & \qquad \langle \text{ Definition 27 } \rangle \\
 & \mathcal{P} \Uparrow\!\Uparrow ((J_S, K_S \ :: \ \sqcup(J_Q, J_R, K_Q, K_R \ : \ J_S = J_Q \cup J_R \ \wedge \ K_S = K_Q \cup K_R \\
 & \ : \ Q_{J_Q K_Q} ;\delta_{\overline{K_Q \cap K_R}} \sqcap R_{J_R K_R} ;\delta_{\overline{K_Q \cap K_R}})), \boldsymbol{\alpha}_Q \sqcap \boldsymbol{\alpha}_R, E_Q \cup E_R, \boldsymbol{\omega}_Q \sqcap \boldsymbol{\omega}_R, F_Q \cup F_R) \\
= & \qquad \langle \text{ Definition 27 } \rangle \\
 & ((J, K \ :: \ \sqcup(J_P, J_S, K_P, K_S \ : \ J = J_S \cup J_P \ \wedge \ K = K_S \cup K_P \\
 & \ : \ P_{J_P K_P} ;\delta_{\overline{K_P \cap K_S}} \sqcap \ \sqcup(J_Q, J_R, K_Q, K_R \ : \ J_S = J_Q \cup J_R \ \wedge \ K_S = K_Q \cup K_R \\
 & \ : \ (Q_{J_Q K_Q} ;\delta_{\overline{K_Q \cap K_R}} \sqcap R_{J_R K_R} ;\delta_{\overline{K_Q \cap K_R}}) ;\delta_{\overline{K_P \cap K_S}}))), \boldsymbol{\alpha}_P \sqcap \boldsymbol{\alpha}_Q \sqcap \boldsymbol{\alpha}_R, \\
 & E_P \cup E_Q \cup E_R, \boldsymbol{\omega}_P \sqcap \boldsymbol{\omega}_Q \sqcap \boldsymbol{\omega}_R, F_P \cup F_Q \cup F_R) \\
= & \\
 & ((J, K \ :: \ \sqcup(J_P, J_Q, J_R, K_P, K_Q, K_R \\
 & \ : \ J = J_P \cup J_Q \cup J_R \ \wedge \ K = K_P \cup K_Q \cup K_R \\
 & \ : \ P_{J_P K_P} ;\delta_{\overline{K_P \cap (K_Q \cup K_R)}} \sqcap (Q_{J_Q K_Q} ;\delta_{\overline{K_Q \cap K_R}} \sqcap R_{J_R K_R} ;\delta_{\overline{K_Q \cap K_R}}) ;\delta_{\overline{K_P \cap (K_Q \cup K_R)}})), \\
 & \boldsymbol{\alpha}_P \sqcap \boldsymbol{\alpha}_Q \sqcap \boldsymbol{\alpha}_R, E_P \cup E_Q \cup E_R, \boldsymbol{\omega}_P \sqcap \boldsymbol{\omega}_Q \sqcap \boldsymbol{\omega}_R, F_P \cup F_Q \cup F_R) \\
= & \qquad \langle \text{ Equation 28 } \rangle \\
 & ((J, K \ :: \ \sqcup(J_P, J_Q, J_R, K_P, K_Q, K_R \\
 & \ : \ J = J_P \cup J_Q \cup J_R \ \wedge \ K = K_P \cup K_Q \cup K_R \\
 & \ : \ (P_{J_P K_P} ;\delta_{\overline{K_P \cap K_Q}} \sqcap Q_{J_Q K_Q} ;\delta_{\overline{K_P \cap K_Q}}) ;\delta_{\overline{(K_P \cup K_Q) \cap K_R}} \sqcap R_{J_R K_R} ;\delta_{\overline{(K_P \cup K_Q) \cap K_R}})), \\
 & \boldsymbol{\alpha}_P \sqcap \boldsymbol{\alpha}_Q \sqcap \boldsymbol{\alpha}_R, E_P \cup E_Q \cup E_R, \boldsymbol{\omega}_P \sqcap \boldsymbol{\omega}_Q \sqcap \boldsymbol{\omega}_R, F_P \cup F_Q \cup F_R) \\
= & \\
 & ((J, K \ :: \ \sqcup(J_S, J_R, K_S, K_R \ : \ J = J_S \cup J_R \ \wedge \ K = K_S \cup K_R \\
 & \ : \ \sqcup(J_P, J_Q, K_P, K_Q \\
 & \ : \ J_S = J_P \cup J_Q \ \wedge \ K_S = K_P \cup K_Q \\
 & \ : \ (P_{J_P K_P} ;\delta_{\overline{K_P \cap K_Q}} \sqcap Q_{J_Q K_Q} ;\delta_{\overline{K_P \cap K_Q}}) ;\delta_{\overline{K_S \cap K_R}}) \sqcap R_{J_R K_R} ;\delta_{\overline{K_R \cap K_S}})), \\
 & \boldsymbol{\alpha}_P \sqcap \boldsymbol{\alpha}_Q \sqcap \boldsymbol{\alpha}_R, E_P \cup E_Q \cup E_R, \boldsymbol{\omega}_P \sqcap \boldsymbol{\omega}_Q \sqcap \boldsymbol{\omega}_R, F_P \cup F_Q \cup F_R) \\
= & \qquad \langle \ \mathcal{S} = \mathcal{P} \Uparrow\!\Uparrow \mathcal{Q} \ \rangle
\end{array}
$$

$$
\begin{array}{l}
\quad (\,(J_S, K_S \;::\; \sqcup(J_P, J_Q, K_P, K_Q \;:\; J_S = J_P \cup J_Q \;\wedge\; K_S = K_P \cup K_Q \\
\quad :\; P_{J_P K_P} ; \delta_{\overline{K_P \cap K_Q}} \sqcap Q_{J_Q K_Q} ; \delta_{\overline{K_p \cap K_Q}}\,)\,), \\
\quad \alpha_P \sqcap \alpha_Q, E_P \cup E_Q, \omega_P \sqcap \omega_Q, F_P \cup F_Q) \\
\quad \Uparrow \mathcal{R} \\
= \qquad\qquad \langle \text{ Definition 27 } \rangle \\
\quad (\mathcal{P} \Uparrow \mathcal{Q}) \Uparrow \mathcal{R}
\end{array}
$$

■

Consider the 5-tuple

30. $\mathcal{O} \triangleq (\{\top_{\emptyset\emptyset}\}, \mathbb{I}, \emptyset, \mathbb{I}, \emptyset)$.

Because $\top_{\emptyset\emptyset} = \top = \delta_\emptyset ; \top_{\emptyset\emptyset} ; \delta_\emptyset$ and $\delta_\emptyset ; \mathbb{I} ; \delta_\emptyset \sqcap \mathbb{I} = \mathbb{I}$, the 5-tuple $\mathcal{O}$ is a relational process. It models a process which does not control any resource and it is able to perform only actions that do not need any resource for their accomplishment (as a *skip* statement — which is a trivial do-nothing statement —). Also, it is a process such that each of its states is both an input state and an output state.

The following proposition claims that the relational process $\mathcal{O}$, given by Equation 30, is a neutral element for the operator $\Uparrow$.

Proposition 31. *Let $\mathcal{P}$ be any relational process. We have $\mathcal{O} \Uparrow \mathcal{P} = \mathcal{P} \Uparrow \mathcal{O} = \mathcal{P}$.*

Proof. This results from Proposition 29 (*i.e.* $\Uparrow$ is commutative), Definition 27 and Proposition 9(3). ■

Consider the 5-tuple

32. $\mathcal{Z} \triangleq (\,(J, K \;::\; \bot_{JK}\,), \bot, \mathcal{U}, \bot, \mathcal{U})$.

It is a relational process, since $\bot = \delta_\mathcal{U} ; \bot ; \delta_\mathcal{U} \sqcap \mathbb{I}$ and $\bot_{JK} = \bot = \delta_J ; \bot_{JK} ; \delta_K$. The following proposition asserts that it is an absorbing element for $\Uparrow$.

Proposition 33. *Let $\mathcal{P}$ be a relational process. We have $\mathcal{Z} \Uparrow \mathcal{P} = \mathcal{P} \Uparrow \mathcal{Z} = \mathcal{Z}$.*

Proof. The result follows from Proposition 29, Definition 13, and Definition 27. ■

The operator $\Uparrow$ is not idempotent. Indeed, let $\mathcal{P} \triangleq (\{\{x' = 1\}_{\emptyset,\{x\}}\}, \alpha_P, E_P, \omega_P, F_P)$, where α_P, E_P, ω_P and F_P are arbitrary. We have

$$\mathcal{P} \Uparrow \mathcal{P} = (\{\top_{\emptyset,\{x\}}\}, \alpha_P, E_P, \omega_P, F_P).$$

First, we note that $\mathcal{P} \Uparrow \mathcal{P} \neq \mathcal{P}$. Second, we note that there is an interference leading to an indeterminate final value of x. This interference could represent well the reality of what could happen when x models a section of railway and the value 1 indicates the authorisation to let a train take this section. Imagine the final state of this section of railway when each one of these processes sends a train on it.

The following proposition claims that the operator $\Uparrow$ distributes over $\natural$.

Proposition 34. *Let $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{R}$ be relational processes. We have*

$$\mathcal{P} \Uparrow (\mathcal{Q} \upharpoonright \mathcal{R}) = (\mathcal{P} \Uparrow \mathcal{Q}) \upharpoonright (\mathcal{P} \Uparrow \mathcal{R}).$$

Proof. This results from Definition 18, Definition 27, and Theorem 2(2, 3). ∎

Example 35. We take again the relational processes of Example 14. Let $\mathcal{R}_{mx} \triangleq \mathcal{P} \Uparrow \mathcal{Q}$. Then,

$$\begin{array}{ll} & \mathcal{R}_{mx} \\ = & \qquad \langle \text{ Definition 27 } \ \& \ \{p,y\} \cup \{c,y\} = \{p,c,y\} \ \ \& \ \ \bot\!\!\!\bot \sqcap \bot\!\!\!\bot = \bot\!\!\!\bot \ \ \& \ \ \emptyset \cap \emptyset = \emptyset \ \rangle \\ & ((R_{\{c,p,y\}\{c,p,x,y,z\}}, R_{\{c,p,t,z\}\{c,p,x,t\}}, R_{\{c,p,x,y\}\{c,p,y,z\}}, R_{\{c,p,x,y,t,z\}\{c,p,y,t\}}), \alpha_{R_{mx}}, \{p,c,y\}, \bot\!\!\!\bot, \emptyset) \end{array}$$

where

– $\alpha_{R_{mx}} = \alpha_P \sqcap \alpha_Q = \{c = 1 \ \wedge \ p = 1 \ \wedge \ y = 0\} \sqcap \mathbb{I}$,

– $$\begin{array}{ll} & R_{\{c,p,y\}\{c,p,x,y,z\}} \\ = & \qquad \langle \text{ Definition 27 } \ \& \ \{p,x\} \cap \{c,y,z\} = \emptyset \ \rangle \\ & P_{\{p\}\{p,x\}} ; \delta_{\overline{\emptyset}} \sqcap Q_{\{c,y\}\{c,y,z\}} ; \delta_{\overline{\emptyset}} \\ = & \qquad \langle \ \delta_{\overline{\emptyset}} = \delta_u = \mathbb{I} \ \ \& \ \ P ; \mathbb{I} = P \ \rangle \\ & P_{\{p\}\{p,x\}} \sqcap Q_{\{c,y\}\{c,y,z\}} \\ = & \qquad \langle \text{ by replacing } P_{\{p\}\{p,x\}} \text{ and } Q_{\{c,y\}\{c,y,z\}} \text{ by their value } \rangle \\ & \{c = 1 \ \wedge \ c' = 2 \ \wedge \ p = 1 \ \wedge \ p' = 2 \ \wedge \ x' = 1 \ \wedge \ y > 0 \ \wedge \ y' = y - 1 \ \wedge \ z' = 1\} \end{array}$$

– $$\begin{array}{ll} & R_{\{c,p,t,z\}\{c,p,x,t\}} \\ = & \qquad \langle \text{ Definition 27 } \ \& \ \{p,x\} \cap \{c,t\} = \emptyset \ \rangle \\ & P_{\{p\}\{p,x\}} ; \delta_{\overline{\emptyset}} \sqcap Q_{\{c,t,z\}\{c,t\}} ; \delta_{\overline{\emptyset}} \\ = & \qquad \langle \ \delta_{\overline{\emptyset}} = \delta_u = \mathbb{I} \ \ \& \ \ P ; \mathbb{I} = P \ \rangle \\ & P_{\{p\}\{p,x\}} \sqcap Q_{\{c,t,z\}\{c,t\}} \\ = & \qquad \langle \text{ by replacing } P_{\{p\}\{p,x\}} \text{ and } Q_{\{c,t,z\}\{c,t\}} \text{ by their value } \rangle \\ & \{c = 2 \ \wedge \ c' = 1 \ \wedge \ p = 1 \ \wedge \ p' = 2 \ \wedge \ x' = 1 \ \wedge \ t' = f(z,t)\} \end{array}$$

– $$\begin{array}{ll} & R_{\{c,p,x,y\}\{c,p,y,z\}} \\ = & \qquad \langle \text{ Definition 27 } \ \& \ \{p,y\} \cap \{c,y,z\} = \{y\} \ \rangle \\ & P_{\{p,x,y\}\{p,y\}} ; \delta_{\overline{\{y\}}} \sqcap Q_{\{c,y\}\{c,y,z\}} ; \delta_{\overline{\{y\}}} \\ = & \qquad \langle \text{ by replacing } P_{\{p,x,y\}\{p,y\}} \text{ and } Q_{\{c,y\}\{c,y,z\}} \text{ by their value } \rangle \\ & \{p = 2 \ \wedge \ p' = 1 \ \wedge \ y < N \ \wedge \ y' = y + x\} ; \delta_{\overline{\{y\}}} \\ & \sqcap \{c = 1 \ \wedge \ c' = 2 \ \wedge \ y > 0 \ \wedge \ y' = y - 1 \ \wedge \ z' = 1\} ; \delta_{\overline{\{y\}}} \\ = & \qquad \langle \ \delta_J ; P ; \delta_K ; \delta_{\overline{L}} = \delta_J ; P ; \delta_{K \sqcap \overline{L}} = \delta_J ; P ; \delta_{K - L} \ \ \& \ \text{ all calculations done } \rangle \\ & \{c = 1 \ \wedge \ c' = 2 \ \wedge \ p = 2 \ \wedge \ p' = 1 \ \wedge \ y < N \ \wedge \ y > 0 \ \wedge \ z' = 1\} \end{array}$$

Note that y' is not present in the expression of $R_{\{c,p,x,y\}\{c,p,y,z\}}$ in the previous line (its value is thus arbitrary), in spite of the fact that the system controls the resource represented

by the variable y. This is caused by the interference of $\mathcal{P}$ (attempting $y' = y + x$) with $\mathcal{Q}$ (attempting $y' = y - 1$).

– $R_{\{c,p,x,y,t,z\}\{c,p,y,t\}}$

$= \quad \langle$ Definition 27 & $\{p,y\} \cap \{c,t\} = \emptyset$ $\rangle$

$P_{\{p,x,y\}\{p,y\}};\delta_{\overline{\emptyset}} \sqcap Q_{\{c,t,z\}\{c,t\}};\delta_{\overline{\emptyset}}$

$= \quad \langle$ $\delta_{\overline{\emptyset}} = \delta_{\mathcal{U}} = \mathbb{I}$ & $P;\mathbb{I} = P$ & values of $P_{\{p,x,y\}\{p,y\}}$ and $Q_{\{c,t,z\}\{c,t\}}$ $\rangle$

$P_{\{p,x,y\}\{p,y\}} \sqcap Q_{\{c,t,z\}\{c,t\}}$

$= \quad \langle$ all calculations done $\rangle$

$\{c = 2 \ \wedge \ c' = 1 \ \wedge \ p = 2 \ \wedge \ p' = 1 \ \wedge \ y < N \ \wedge \ y' = y + x \ \wedge \ t' = f(z,t)\}$

The relational process $\mathcal{R}_{mx}$ is graphically illustrated in Figure 3.

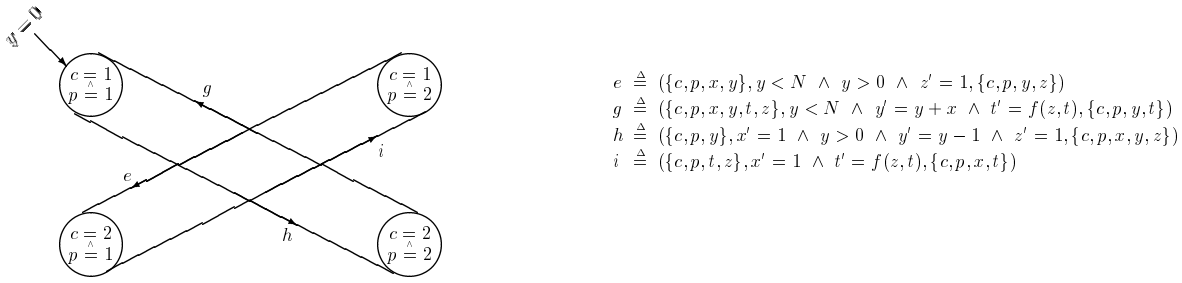


Fig. 3. A graphical illustration of the maximal totally synchronous parallel composition of $\mathcal{P}$ and $\mathcal{Q}$

6 True-concurrency composition

First, we introduce a generic operator expressing true concurrency by using the interleaved composition operator and any of the synchronous composition operators. Then we give some of its properties. Finally, we instantiate the generic operator by choosing the totally synchrounous maximal parallel composition operator in its definition.

Definition 36. Let

$$\mathcal{P} = (\,(J,K \ :: \ P_{JK}\,), \alpha_P, E_P, \omega_P, F_P) \text{ and } \mathcal{Q} = (\,(J,K \ :: \ Q_{JK}\,), \alpha_Q, E_Q, \omega_Q, F_Q)$$

be two relational processes. The *true-concurrency composition* of $\mathcal{P}$ and $\mathcal{Q}$, noted $\mathcal{P} \between \mathcal{Q}$, is defined as $\mathcal{P} \between \mathcal{Q} \triangleq \mathcal{P} \between \mathcal{Q} \between (\mathcal{P} \parallel \mathcal{Q})$, where $\parallel$ is a totally synchronous parallel composition operator that is commutative, associative and distributes over $\between$. ∎

Proposition 37. *If $\mathcal{P} \parallel \mathcal{Q}$ is a relational process, then $\mathcal{P} \between \mathcal{Q}$ is a relational process.*

Proof. This follows from the fact that

$$\wedge\,(\mathcal{P}, \mathcal{Q} \;:\; \mathcal{P}, \mathcal{Q} \text{ relational processes } \;:\; \mathcal{P} \between \mathcal{Q} \text{ is a relational process }).$$

■

In the sequel, we assume that the totally synchronous parallel composition operators (*i.e.* $\Uparrow, \|$) have a higher precedence than $\between$. Then, we have

$$\mathcal{P} \Vdash \mathcal{Q} \;\triangleq\; \mathcal{P} \between \mathcal{Q} \between (\mathcal{P} \parallel \mathcal{Q}) \;=\; \mathcal{P} \between \mathcal{Q} \between \mathcal{P} \parallel \mathcal{Q}.$$

This expression indicates that at any moment $\mathcal{P}$, or $\mathcal{Q}$, or both simultaneously, are accomplishing an action.

Proposition 38. *The operator* $\Vdash$

(a) is commutative,
(b) is associative,
(c) distributes over $\between$.

Proof.

(a) This follows from Definition 36 and the commutativity of $\between$ and $\|$.
(b) Associativity follows from Definition 36, the associativity of $\between$, the distributivity of $\|$ over $\between$, the associativity of $\|$ and the commutativity of $\between$.
(c) This follows from Definition 36, the associativity of $\between$, the distributivity of $\|$ over $\between$, the commutativity and associativity of $\between$, and Proposition 20. ■

Using the operators $\between$ and $\Uparrow$, the following definition introduces the maximal true-concurrency composition operator.

Definition 39. Let

$$\mathcal{P} = (\,(J, K \;::\; P_{JK}\,), \alpha_P, E_P, \omega_P, F_P) \text{ and } \mathcal{Q} = (\,(J, K \;::\; Q_{JK}\,), \alpha_Q, E_Q, \omega_Q, F_Q)$$

be two relational processes. The *maximal true concurrency composition* of $\mathcal{P}$ and $\mathcal{Q}$, noted $\mathcal{P} \pitchfork \mathcal{Q}$, is defined as $\mathcal{P} \pitchfork \mathcal{Q} \triangleq \mathcal{P} \between \mathcal{Q} \between \mathcal{P} \Uparrow \mathcal{Q}$. ■

We already showed that $\mathcal{P} \Uparrow \mathcal{Q}$ is a relational process if $\mathcal{P}$ et $\mathcal{Q}$ are relational processes. Then, according to Proposition 37, $\mathcal{P} \pitchfork \mathcal{Q}$ is a relational process.

Also, according to Proposition 29, Proposition 34 and Proposition 38, the operator $\pitchfork$ is commutative, associative and distributes over $\between$.

Example 40. We take the relational processes of Example 14. Let $\mathcal{R}_{tx} \triangleq \mathcal{P} \between \mathcal{Q}$. Then,

$$
\begin{array}{ll}
 & \mathcal{R}_{tx} \\
= & \quad \langle \text{ Definition 39 } \rangle \\
 & \mathcal{P} \upharpoonright \mathcal{Q} \upharpoonright \mathcal{P} \Uparrow \mathcal{Q} \\
= & \quad \langle \ \mathcal{P} \upharpoonright \mathcal{Q} \text{ is calculated in Example 26 and called } \mathcal{R}_e \quad \& \quad \mathcal{P} \Uparrow \mathcal{Q} \text{ is calculated} \\
 & \quad \text{in Example 35 and called } \mathcal{R}_{mx} \ \rangle \\
 & \mathcal{R}_e \upharpoonright \mathcal{R}_{mx} \\
= & \quad \langle \text{ Definition 18 } \quad \& \quad \text{all calculations done } \rangle \\
 & ((P_{\{p\}\{p,x\}}, P_{\{p,x,y\}\{p,y\}}, Q_{\{c,y\}\{c,y,z\}}, Q_{\{c,t,z\}\{c,t\}}, R_{\{c,p,y\}\{c,p,x,y,z\}}, R_{\{c,p,t,z\}\{c,p,x,t\}}, \\
 & R_{\{c,p,x,y\}\{c,p,y,z\}}, R_{\{c,p,x,y,t,z\}\{c,p,y,t\}}), \alpha_P \sqcap \alpha_Q, \{p,c,y\}, \bot\!\!\!\bot, \emptyset)
\end{array}
$$

where $P_{\{p\}\{p,x\}}, P_{\{p,x,y\}\{p,y\}}, Q_{\{c,y\}\{c,y,z\}}, Q_{\{c,t,z\}\{c,t\}}, \alpha_P$ and α_Q are given in Example 26 and the others terms are given in Example 35. The relational process $\mathcal{R}_{tx}$ is graphically illustrated in Figure 4.

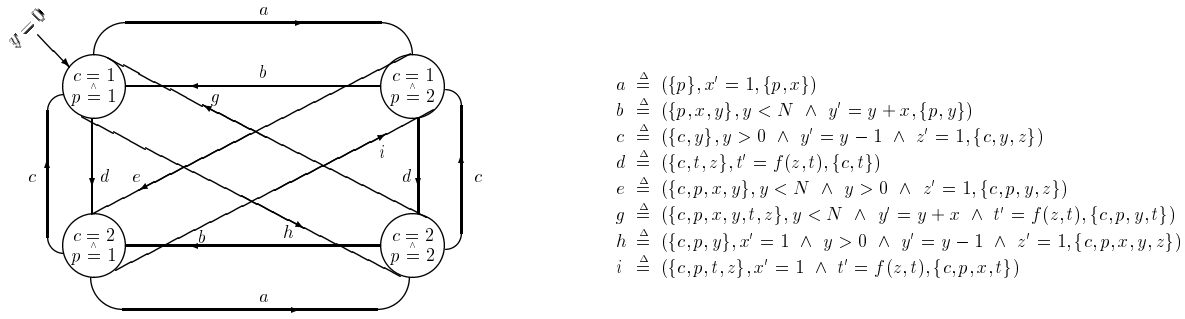


Fig. 4. A graphical illustration of the maximal true-concurrency composition of $\mathcal{P}$ and $\mathcal{Q}$

7 Conclusion

We have presented a process model suitable for representing both sequential an concurrent systems, since it represents a process (system) as an open system, and the associated relation gives a description of this system perceived as being closed. We have then defined two operators, each representing an extreme perception of concurrency. One reduces concurrency to an interleaved behaviour and the other reduces concurrency to a totally synchronous behaviour. Then, by combining these operators, we have defined an operator expressing true concurrency. When many processes interfere on a given resource in order to modify it, each in its way, the maximal totally synchronous parallel composition operator and the maximal true-concurrency composition operator express this situation by letting the final value of

the variable modelling this resource being indeterminate. Hence, it allows the detection of interferences between processes.

In order to be able to use the relational processes to model large-scale systems, our future work aims at equipping them with a documentation inspired by what is presented for sequential systems in [18] and which allowed the study of large-scale critical systems [17]. Concerning the operators expressing concurrency, we aim at using them in requirements engineering to model concurrent scenarios.

References

[1] H. Andréka and I. Németi. On cylindric-relativized set algebras. In A. Dold and B. Eckmann, editors, *Cylindric Set Algebras*, volume 883 of *Lecture Notes in Mathematics*, pages 131–315. Springer-Verlag, 1981.

[2] A. Arnold. *Systèmes de transitions finis et sémantique des processus communicants*. Masson, 1992.

[3] G. Boudol and I. Castellani. Concurrency and atomicity. *Theoretical Computer Science*, 59(1,2):25–84, 1988.

[4] L. Chin and A. Tarski. Distributive and modular laws in the arithmetic of relation algebras. *University of California Publications*, 1:341–384, 1951.

[5] J. W. de Bakker, J. N. Kok, J. J. C. Meyer, E. R. Olderog, and J. I. Zucker. Contrasting themes in the semantics of imperative concurrency. In J. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Current trends in concurrency*, volume 224 of *Lecture Notes in Computer Science*, pages 51–121, Berlin, 1986. Springer-Verlag.

[6] J. Desharnais. Monomorphic characterization of n-ary direct products. In *3rd Seminar on Relational Methods in Computer Science*, pages 359–368, Hammamet, Tunisia, January 1997.

[7] E. W. Dijkstra. Solution of a problem in concurrent programming control. *Comm. ACM*, 8(9):569, 1965.

[8] E. W. Dijkstra. Hierarchical ordering of sequential processes. *Acta Informatica*, 1:115–138, 1971.

[9] L. Henkin, J. D. Monk, and A. Tarski. *Cylindric Algebras, Part I.* North-Holland Publishing Company, Amsterdam, 1971.

[10] L. Henkin, J. D. Monk, and A. Tarski. Cylindric set algebras and related structures. In A. Dold and B. Eckmann, editors, *Cylindric Set Algebras*, volume 883 of *Lecture Notes in Mathematics*, pages 1–129. Springer-Verlag, 1981.

[11] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall International Series in Computer Science, 1985.

[12] J. Hooman and J. Widom. A temporal-logic based compositional proof system for real-time message passing. In E. Odijk, M. Rem, and J.-C. Syr, editors, *PARLE'89 Prarllel Architectures and Languages Europe*, volume II, pages 424–441, New York, 1989. Springer-Verlag.

[13] R. Khédri. *Concurrence, bisimulation et équation d'interface : une approche relationnelle*. PhD thesis, Faculté des études supérieures de l'Université Laval, Québec, Canada, 1998.

[14] L. Y. Liu and R. K. Shyamasundar. Static analysis of real-time distributed systems. *IEEE Transactions on Software Engineering*, 16(4):373–388, April 1990.

[15] R. Milner. *Communication and Concurrency*. Prentice Hall International Series in Computer Science, 1989.

[16] E.-R. Olderog. Operational Petri net semantics for CCSP. In *Advances in Petri Nets*, volume 266 of *Lecture Notes in Computer Science*, pages 196–223, Berlin, 1987. Springer-Verlag.

[17] D. L. Parnas, G. J. K. Asmis, and J. Madey. Assessment of safety-critical software in nuclear power plants. *Nuclear Safety*, 32(2):189–198, April-June 1991.

[18] D. L. Parnas, J. Madey, and M. Iglewski. Precise documentation of well-structured programs. *IEEE Transactions on Software Engineering*, 20(12):948–976, December 1994.

[19] C. A. Petri. *Kommunikation mit automaten*. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIm No.2, 1962. Also published as technical report: Communication with Automata. RADC-TR-65-377, Vol.1, Suppl.1, Applied Data Research, Princeton, NJ, 1966.

[20] W. Reisig. *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.

[21] G. Schmidt and T. Ströhlein. *Relations and Graphs*. EATCS Monographs in Computer Science, Springer-Verlag, 1993.

[22] R. J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions*. PhD thesis, Vrije Universiteit te Amsterdam, 1990.