What is an Event for Membrane Systems?*

Gabriel Ciobanu[1,2] and Dorel Lucanu[1]

[1] "A.I.Cuza" University of Iaşi, Faculty of Computer Science
[2] Romanian Academy, Institute of Computer Science, Iaşi
{gabriel, dlucanu}@info.uaic.ro

Summary. Event structures are formal models for parallel and nondeterministic systems. The use of the event structures defines formally where and how the parallelism and nondeterminism appear in systems. We study the event structure for membrane systems, considering both the causality and the conflict relations. We investigate how an event structures can be associated to an parallel evolution step. Two cases are considered: first the rules are applied over strings, and then rules are applied over multisets. We show that, if we apply the standard procedure of extracting event structures from labelled transition systems, then the meanings for event and causality are different for the two cases. The commutativity in the case of multisets introduces some "false" causalities which must be removed in order to capture the right parallelism of the membrane systems.

1 Introduction

Membrane systems [5] describe a computation mechanism inspired by cellular biology. Parallelism and nondeterminism represent the essential features of the membrane computation. Membrane computation is strongly parallel and nondeterministic mainly due to its way of applying the rules of a membrane to its objects. The rules associated with a compartment are applied to the objects from that compartment in a (maximally) parallel way, and the rules are chosen in a non-deterministic manner. Moreover, all compartments of the system evolve at the same time, and so we have two levels of parallelism.

In this paper we study the nature of parallelism and nondeterminism of the membrane systems in terms of a widely recognized formal model for parallelism and nondeterminism, namely in terms of event structures [7].

According to [6], the models for concurrency are classified according to the following criteria: whether they can represent the structure of systems or just their behaviours; whether they can faithfully take into account the difference between parallel and sequential computation (interleaving or non-interleaving model); and

* This work has been supported by the research grant CEEX 47/2005, Romania

whether they can represent the branching structure of processes related to nondeterministic choices (linear or branching time model). Event structures are chosen to represent the non-interleaving and branching-time models, and so they model the true concurrency (parallelism) and nondeterminism.

2 Event Structures

In event-based models, a system is represented by a set of events (action occurrences) together with some structure on this set, determining the causality relations between the events. The causality between actions is expressed by a partial order, and the nondeterminism is expressed by a conflict relation on actions. For every two events d and e it is specified either whether one of them is a prerequisite for the other, whether they exclude each other, or whether they may happen in parallel. The behaviour of an event structure is formalized by associating to it a family of configurations representing sets of events which occur during (partial) runs of the system.

A parallel (concurrent) step is simultaneously executing several rules, each of them producing events which end up in the resulting event configuration. These steps are presumably cooperating to achieve a goal, and so they are not totally independent. They synchronize at certain points, and this is reflected in the events produced.

There are many levels of granularity at which one might describe the events that occur as a process executes. At the level of the components from which the system is composed, computation consists of events which reflect the rules. At a higher level, the system might be viewed in terms of parallel executions of its components. Thus, when we talk about the events which occur within a system, it is understood that we know the granularity of the representation, and that events are encoded to this granularity (degree of precision).

Definition 1 (Event Structure).
An event structures is a triple $(E, \leq, \#)$ *where*

- E *is a set of events,*
- $\leq \subseteq E \times E$ *is a partial order, the* causality relation*,*
- $\# \subseteq E \times E$ *is an irreflexive and symmetric relation, the* conflict relation*, satisfying the* principle of conflict heredity*:*

$$\forall e_1, e_2, e_3 \in E.\ e_1 \leq e_2 \wedge e_1 \# e_3 \Rightarrow e_2 \# e_3.$$

A more detailed presentation of event structures can be found in [8].

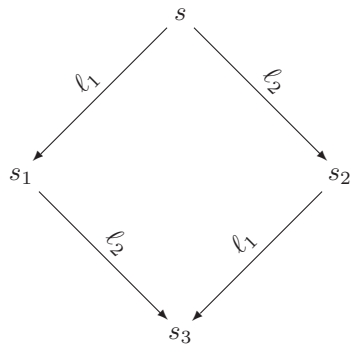
2.1 Event Structure Associated to a Labelled Transition System

There are many cases when the operational semantics of a system is given by means of a labelled transition system (lts) describing all possible sequential computations. In order to study the concurrency properties, we must determine the event structure defined by such a labelled transition system.

Let $(S, \rightarrow, L, s_0)$ be a transition system, where S is the set o states, $\rightarrow$ is the transition relation consisting of triples $(s, \ell, s') \in S \times L \times S$, often written as $s \xrightarrow{\ell} s'$, L is the set of labels (actions), and s_0 is the initial state. A (sequential) *computation* is a sequence $s_0 \xrightarrow{\ell_1} s_1 \ldots \xrightarrow{\ell_n} s_n$ such that $(s_{i-1}, \ell_i, s_i) \in \rightarrow$.

Definition 2 (Events in a lts).
Let $\sim$ be the smallest equivalence satisfying: if (s, ℓ_1, s_1), (s, ℓ_2, s_2), (s_1, ℓ_2, s_3), (s_2, ℓ_1, s_3), $\in \rightarrow$ and ($\ell_1 \neq \ell_2$ or $s_1 \neq s_2$), then $(s, \ell_1, s_1) \sim (s_2, \ell_1, s_3)$.
An event *is a $\sim$-equivalence class written as $[s, \ell, s']$.*

Intuitively, two transitions are equivalent iff they are occurrences of the same event. The relation $\sim$ can be easier understood from the following picture:



We have two events which may occur in any order, i.e., the two events are concurrent.

Definition 3 (Configuration in a lts).
A configuration *is a multiset of events $[s_{i-1}, \ell_i, s_i]$ corresponding to a computation $s_0 \xrightarrow{\ell_1} s_1 \ldots \xrightarrow{\ell_n} s_n$.*

Since all computations start from s_0, each prefix of a configuration is also a configuration.

Theorem 1 (Event Structure of a lts). *[4]*
$(S, \rightarrow, L, s_0)$ can be organized as an event structure.

Proof (Sketch). The event structure $(E, <, \#)$ is defined by:

- E is the set of events as defined in Definition 2;
- $e_1 < e_2$ if every configuration which contains e_2 also contains e_1;
- $e_1 \# e_2$ if there is no configuration containing both e_1 and e_2.

Example 1. The events defined by the lts in Figure 1 are:

$$E_1 = \{(s_0, \ell_1, s_1)\} \qquad E_4 = \{(s_1, \ell_2, s_4), (s_3, \ell_2, s_5)\}$$
$$E_2 = \{(s_0, \ell_2, s_2)\} \qquad E_5 = \{(s_2, \ell_1, s_6)\}$$
$$E_3 = \{(s_1, \ell_3, s_3), (s_4, \ell_3, s_5)\} \qquad E_6 = \{(s_4, \ell_1, s_6)\}$$

A configuration describe a (partial) computation expressed in terms of events. This lts defines the following configurations:

$$E_1 \qquad E_2\ E_5$$
$$E_2 \qquad E_1\ E_3\ E_4$$
$$E_1\ E_3 \qquad E_1\ E_4\ E_3$$
$$E_1\ E_4 \qquad E_1\ E_4\ E_6$$

We have $E_1 < E_3$ because any configuration containing E_3 also contains E_1. Since any occurrence of E_3 is always after an occurrence of E_1, it follows that there is causal relationship between the two events. We get $E_1 < E_4 < E_6$ and $E_2 < E_5$ in a similar way.
Since there is no a configuration containing both E_1 and E_2, it follows that there is a conflict between the two events, i.e., $E_1 \# E_2$. We get $E_1 \# E_5$, $E_2 \# E_3$, $E_2 \# E_4$, $E_2 \# E_6$, $E_5 \# E_3$, $E_5 \# E_4$, and $E_5 \# E_6$ in a similar way.

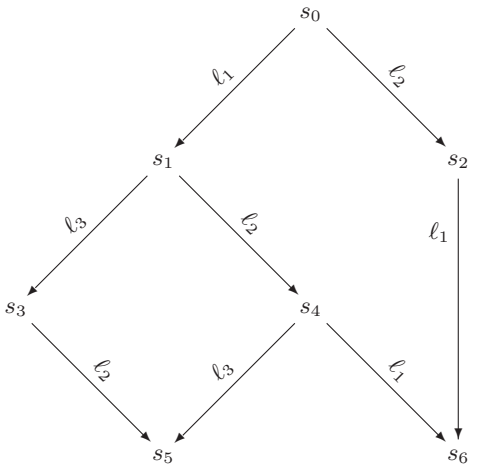


Fig. 1. A lts example

3 Event Structure of an Evolution Step

We assume that the reader is familiar with membrane systems (see [5] for a detailed presentation). A membrane structure and their contents, represented as multisets of objects, identify a configuration of a P system. By a nondeterministic and parallel use of rules, the system can pass to another configuration; such a step is called a transition. A sequence of transitions constitutes a computation. Because of the nondeterminism of the application of rules, starting from an initial configuration, we can get several successful computations.

Membrane computation is strongly parallel and nondeterministic mainly due to its way of applying the rules of a membrane to its objects. The rules associated with a compartment are applied to the objects from that compartment in a (maximally) parallel way, and the rules are chosen in a non-deterministic manner. We consider here only membrane systems with a single membrane.

Example 2. Let us consider a simple membrane consisting of the following three rules

$$\ell_1 : a \to b$$
$$\ell_2 : b \to a$$
$$\ell_3 : ab \to d$$

and having the content $aabc$. We investigate the space of all sequential rewritings corresponding to the application of rules in the evolution step $aabc \overset{mpr}{\Rightarrow} bbac$ in order to discover the events of this step. The exact definitions for $\overset{mpr}{\Rightarrow}$ is given in the next subsections.

3.1 Non-commutative Case

We first assume that the sequential rewritings are executed over non-commutative words (strings).

A context is a string of the form $w \bullet w'$, where w, w' are strings of objects, and $\bullet$ is a special symbol. Each rewriting step $wuw' \to wvw'$ is uniquely determined by the context $w \bullet w'$ and the rule $\ell : u \to v$. Therefore the transition $(wuw', \ell, wvw') = wuw' \xrightarrow{\ell} wvw'$ is denoted by $(w \bullet w', \ell)$.

The maximal parallel rewriting over strings is defined as follows: $w \overset{mpr}{\Rightarrow} w'$ if and only if there are $\ell_1, \ldots, \ell_n$, $\ell_i : u_i \to v_i$ $(i = \overline{1,n})$ such that $w = w_0u_1w_1 \ldots u_nw_n$, $w' = w_0v_1w_1 \ldots v_nw_n$ and $w_0w_1 \ldots w_n$ irreducible (no rule can be applied).

We consider first an example. The space of all sequential rewritings for the evolution step of Example 2 is represented in Figure 2.a.

By Definition 4, we have the following three independent events:

$$A = \{(\bullet abc, \ell_1), (\bullet bbc, \ell_1), (\bullet bac, \ell_1), , (\bullet aac, \ell_1)\}$$
$$B = \{(a \bullet bc, \ell_1), (b \bullet bc, \ell_1), (b \bullet ac, \ell_1), , (a \bullet ac, \ell_1)\}$$
$$C = \{(aa \bullet c, \ell_2), (ab \bullet c, \ell_2), (bb \bullet c, \ell_2), , (ba \bullet c, \ell_2)\}.$$

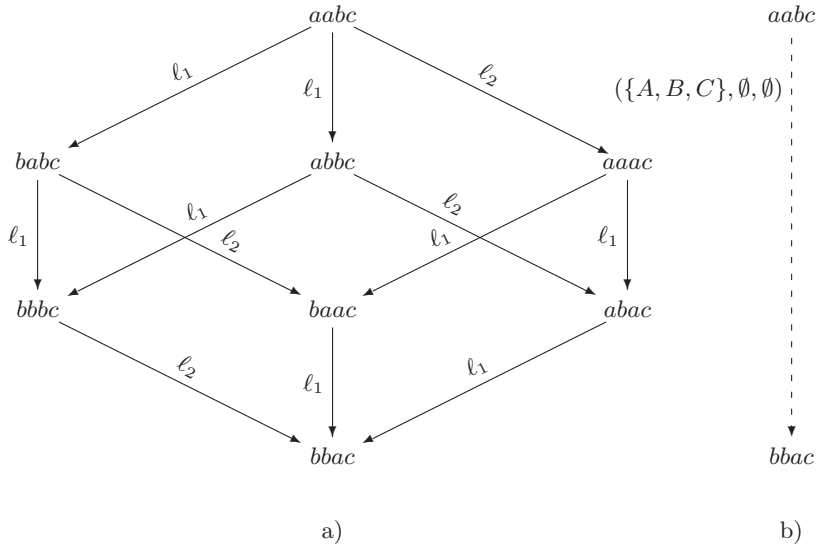


Fig. 2. Lts corresponding to $aabc \overset{mpr}{\Rightarrow} bbac$

Each event corresponds to the application of a certain evolution rule at a certain position in the string. The resulting event structure corresponds very well to the following description: distribute the object to the rules and then apply the evolutions rules in parallel. The parallel execution of all involved evolution rules is possible because the corresponding events are independent (no causalities, no conflicts). Figure 2.b represents the fact that *bbac* is obtained from *aabc* using the computation space described by the event structure $(\{A, B, C\}, \emptyset, \emptyset)$. Any permutation of A, B, C is a computation (in terms of event structures) in this space.

We give now the formal definition for the event structure associated to a mpr-step over strings.

Definition 4. *The labelled transition system associated to $w \overset{mpr}{\Rightarrow} w'$ is given by all the sequential rewritings starting from w and ending in w'. The event structure $ES(w, w')$ associated to $w \overset{mpr}{\Rightarrow} w'$ is the event structure associated to its labelled transition system.*

Theorem 2. *The event structure $ES(w, w') = (E, <, \#)$ associated to $w \overset{mpr}{\Rightarrow} w'$ consists of only independent events, i.e., $< = \emptyset$ and $\# = \emptyset$.*

Proof. We have $w \overset{mpr}{\Rightarrow} w'$ iff $w = w_0u_1w_1 \dots u_nw_n$, $w = w_1v_1w_2 \dots v_nw_n$, $\ell_i : u_i \to v_i$ is an evolution rule, for $i = 1, \dots, n$, and $w_1 \dots w_0w_1 \dots w_n$ is irreducible. The conclusion of the theorem follows by the fact that $[w_0 \dots \bullet\, w_i \dots w_n, \ell_i]$ is a configuration (any of events can occur first).

3.2 Commutative Case

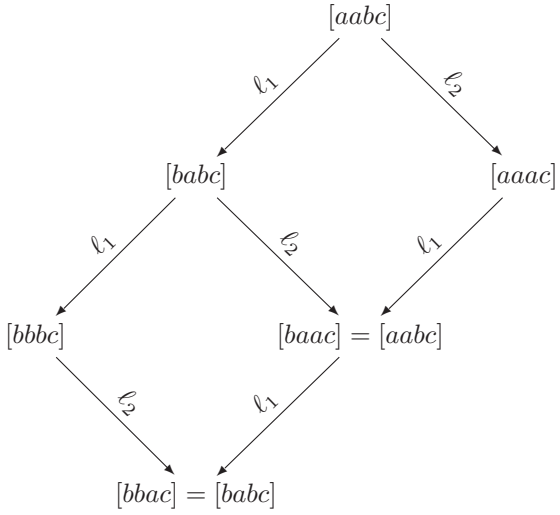
We assume now that the sequential rewritings are executed over commutative words (multisets).

We write $w =_c w'$ if and only if w' is obtained from w by a permutation of the objects, i.e., w and w' are equal modulo commutativity. Let $[w]$ denote the $=_c$-equivalence class of w, i.e., $[w] = \{w' \mid w =_c w'\}$.

A context is a string of the form $[\bullet w]$, where w is a multiset of objects, and $\bullet$ is a special symbol. It is easy to see now that the position of $\bullet$ in a context is not important, and therefore we write $\bullet$ at the beginning. Each rewriting step $[uw] \to [vw]$ is uniquely determined by the context $[\bullet w]$ and the rule $\ell : u \to v$. Therefore the transition $([uw], \ell, [vw]) = [uw] \xrightarrow{\ell} [vw]$ is denoted by $([\bullet w], \ell)$.

The maximal parallel rewriting over multisets is defined as follows: $[w] \overset{mpr}{\Rightarrow} [w']$ iff there are $\ell_1, \ldots, \ell_n$, $\ell_i : u_i \to v_i$ $(i = \overline{1,n})$ such that $[w] = [u_1 \ldots u_n r]$, $[w'] = [v_1 \ldots v_n r$ and r is irreducible (no rule can be applied).

Again we start with an example. The space of all rewritings for the evolution step in Example 2 is:



We also have three events, but they are not totally independent:

$$\overline{A} = \{([\bullet abc], \ell_1), ([\bullet aac], \ell_1)\}$$
$$\overline{B} = \{([\bullet bbc], \ell_1), ([\bullet bac], \ell_1)\}$$
$$\overline{C} = \{([\bullet abc], \ell_2), ([\bullet bac], \ell_2), ([\bullet bbc]\ell_2)\}$$
$$\overline{A} < \overline{B}$$

An event corresponds now to the application of an evolution rule at an arbitrary position. The position in strings cannot be used anymore to distinguish between

events. Moreover, between the events $\overline{A}$ and $\overline{B}$ we have a causal dependency: $\overline{B}$ may occur only after $\overline{A}$. In fact, $\overline{A}$ can be read as "the first application of the evolution rule ℓ_1" and $\overline{B}$ as "the second application of the evolution rule ℓ_1". We notice that the use of commutativity law changes dramatically the meaning of an event.

Definition 5. *The labelled transition system associated to* $[w] \stackrel{mpr}{\Rightarrow} [w']$ *is given by all the sequential rewritings starting from* $[w]$ *and ending in* $[w']$.

Theorem 3. *The event structure* $(E, <, \#)$ *associated to the labelled transition system defined by* $[w] \stackrel{mpr}{\Rightarrow} [w']$ *has the following properties:*

- $[[\bullet w_1], \ell_1] < [[\bullet w_2], \ell_2]$ *if and only if* $\ell_1 = \ell_2$, $[[\bullet w_1], \ell_1]$ *corresponds to the* i*-th application of the rule* ℓ_1, $[[\bullet w_2], \ell_1]$ *corresponds to the* j*-th application of the rule* ℓ_1, *and* $i < j$;
- $\# = \emptyset$.

"Maximal parallel" means "no causal dependency" between the application of the rules. We may conclude either that working with multisets is not a good solution at this granularity, namely it is not possible to determine all the parallel rules applied in a mpr-step, or the procedure which determines the event structure from a lts finds "false" causalities for the particular case when the states are given by multisets. We believe that the later one is true; the causality relation given by $\cdot$-th application of a rule (when it is applied more than once) is artificial. Therefore we remove the false causal dependency in the definition of the event structure associated to a mpr-step.

Definition 6. *The event structure associated to* $[w] \stackrel{mpr}{\Rightarrow} [w']$ *is* $ES([w], [w']) = (E, \emptyset, \emptyset)$, *where* $(E, <, \emptyset)$ *is the event structure associated to the labelled transition system defined by* $[w] \stackrel{mpr}{\Rightarrow} [w']$.

4 Event Structure of a Membrane

In this section we determine the event structure given by a membrane. Since the definition for the event concept is different for the two algebraic structures used for contents, we get two definitions for the event structure of a membrane.

We first note that the notation for events is not longer suitable for the case of membranes. We assume that we have a membrane with two evolution rules: $\ell : a \to b$ and $\ell' : b \to a$. Let us consider the computation $aa \stackrel{mpr}{\Rightarrow} bb \stackrel{mpr}{\Rightarrow} aa \stackrel{mpr}{\Rightarrow} bb$. Since the events of $aa \stackrel{mpr}{\Rightarrow} bb$ occur always before the events of $bb \stackrel{mpr}{\Rightarrow} aa$, and the events of $bb \stackrel{mpr}{\Rightarrow} aa$ occur before the events of $aa \stackrel{mpr}{\Rightarrow} bb$, then we get $[a\bullet, \ell] < [b\bullet, \ell'] < [a\bullet, \ell]$, i.e., the causality relation $<$ is cyclic. Therefore each event $[c, \ell]$ in $ES(w, w')$ is denoted with a new fresh name e, and we define $action(e) = [c, \ell]$. In this way, the event sets corresponding to different computation steps are disjoint.

4.1 Non-commutative Case

If the commutativity law is not suitable at this level of granularity, then a question must be answered: who (and how) computes the correct permutation of the objects such that the maximal parallel rewriting is possible? For instance, if the membrane in Example 2 has the content *bbac*, then this content must be permuted to *babc* in order to have $bbac =_c babc \overset{mpr}{\Rightarrow} acc$.

The answer is given by the nondeterministic allocation of the available resources to the rules before the execution of a mpr-step. Such an allocation of resources is considered as an event appearing before a mpr-step. This event consists in the nondeterministic choice of a permutation of available resources, and reflect the nondeterminism of the membrane computation. Once such a permutation is selected, then the next mpr-step is applied over the corresponding string corresponding to this permutation.

We informally describe the event structure of a membrane M. A computation in M is of the form $w_0 =_c w_0' \overset{mpr}{\Rightarrow} w_1 =_c w_1' \overset{mpr}{\Rightarrow} \cdots$, where w_0 is the initial contents of M. A string (contents) w is *reachable* in M iff there is a computation from w_0 to w. A reachable string w is *ready-to-fire* if there is a w' such that $w \overset{mpr}{\Rightarrow} w'$.

The event structure $ES(M) = (E_M, <_M, \#_M)$ associated to a membrane M is defined as follows:

1. for each reachable w and each w' such that $w \overset{mpr}{\Rightarrow} w'$, $ES(w, w') \subseteq ES(M)$ (we recall that the events in $ES(w, w')$ are renamed);
2. for each reachable w and each ready-to-fire permutation w' of w, $w \neq w'$, we consider in E_M a distinct event e with $action(e)$ equal to $w =_c w'$, and
   a) for each reachable w_1 such that $w_1 \overset{mpr}{\Rightarrow} w$ and for each event e_1 in $ES(w_1, w)$ we have $e_1 < e$, and
   b) for each w_2 such that $w' \overset{mpr}{\Rightarrow} w_2$ and for each event e_2 in $ES(w', w_2)$ we have $e < e_2$;
3. if $e_1, e_2 \in E_M$ such that $action(e_1)$ is $w =_c w_1$ and $action(e_2)$ is $w =_c w_2$ with $w_1 \neq w_2$, then we have $e_1 \# e_2$;
4. if $w \overset{mpr}{\Rightarrow} w_1$, $w \overset{mpr}{\Rightarrow} w_2$ and $w_1 \neq w_2$, then we have $e_1 \# e_2$ in $ES(M)$ for each e_1 in $ES(w, w_1)$ and e_2 in $ES(w, w_2)$.

Example 3. Let M be the membrane presented in Example 2. We consider the computation subspace given by $aabc \overset{mpr}{\Rightarrow} bbac =_c babc \overset{mpr}{\Rightarrow} acc$ and $aabc \overset{mpr}{\Rightarrow} bcc \overset{mpr}{\Rightarrow} acc$. We denote by D the event corresponding to $bbac =_c babc$, by $(\{E, F\}, \emptyset, \emptyset)$ the event structure corresponding to $babc \overset{mpr}{\Rightarrow} acc$, by $(\{A', B'\}, \emptyset, \emptyset)$ the event structure corresponding to $aabc \overset{mpr}{\Rightarrow} bcc$, and by $(\{E'\}, \emptyset, \emptyset)$ the event structure corresponding to $bcc \overset{mpr}{\Rightarrow} acc$. The whole event structure corresponding to the computation subspace is represented in Figure 3.

The events given by item 2 are essential for the definition of the causality relationship. Such an example is the event D in Example 3. We have $A, B, C < D$ because D is causally dependent on A, B, C. The non-deterministic allocation of

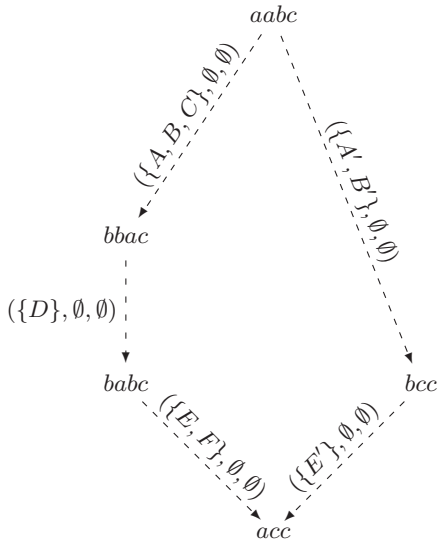


Fig. 3. Event structure corresponding to a membrane computation subspace over strings

the resources to rules is given only after all the evolution rules from the previous mpr-step are completely applied. In this way, all ready-to-fire strings obtained from *bbac* have the same probability. However, it is debatable whether we have to distinguish between ready-to-fire strings producing essentially the same computations, e.g., the computation starting from *babc*, *abbc*, *bcab*, *abcb*, *cbab*, and *cabb* are bisimilar. We also have $D < E, F$ because the evolution rules of the next step can be applied only after the resources are allocated to rules. We assume for the moment that $action(E) = [\bullet abc, \ell_2]$ and $action(F) = [b \bullet c, \ell_3]$. By transitivity, we get $A < F$ even if the application of the rule ℓ_3 in F does not use resources produced by the application of the rule ℓ_1 in A.

The conflict relation # is given by the nondeterministic allocation of the available resources to the rules before the execution of a mpr-step (item 3), and the causality relation is given by the repeating evolution steps (item 4). Again, it is debatable if the events corresponding to $bbac =_c babc$, $bbac =_c abbc$, $bbac =_c bcab$, $bbac =_c abcb$, $bbac =_c cbab$, and $bbac =_c cabb$ are really in conflict.

4.2 Commutative Case

In terms of multisets, a computation in M is of the form $[w_0] \stackrel{mpr}{\Rightarrow} [w_1] \stackrel{mpr}{\Rightarrow} \cdots$, where $[w_0]$ is the initial contents of M. A multiset (contents) w is *reachable* in M iff there is a computation from $[w_0]$ to $[w]$.

The construction of the event structure $\overline{ES}(M) = (E_M, <_M, \#_M)$ associated to a membrane M is simpler than that for the case of strings:

1. for each reachable $[w]$ and each $[w']$ such that $[w] \overset{mpr}{\Rightarrow} [w']$, $ES([w],[w']) \subseteq \overline{ES}(M)$;
2. if $[w] \overset{mpr}{\Rightarrow} [w_1]$, $[w] \overset{mpr}{\Rightarrow} [w_2]$ and $[w_1] \neq [w_2]$, then we have $e_1 \# e_2$ in $ES(M)$ for each e_1 in $ES([w],[w_1])$ and e_2 in $ES([w],[w_2])$;
3. if $[w] \overset{mpr}{\Rightarrow} [w_1]$, $[w_1] \overset{mpr}{\Rightarrow} [w_2]$, then we have $e_1 < e_2$ in $ES(M)$ for each e_1 in $ES([w],[w_1])$ and e_2 in $ES([w_1],[w_2])$.

Example 4. We consider the computation subspace given by $[aabc] \overset{mpr}{\Rightarrow} [bbac] \overset{mpr}{\Rightarrow} [acc]$ and $[aabc] \overset{mpr}{\Rightarrow} [bcc] \overset{mpr}{\Rightarrow} [acc]$. We denote by $(\{\overline{E}, \overline{F}\}, \emptyset, \emptyset)$ the event structure corresponding to $[babc] \overset{mpr}{\Rightarrow} [acc]$, by $(\{\overline{A'}, \overline{B'}\}, \emptyset, \emptyset)$ the event structure corresponding to $[aabc] \overset{mpr}{\Rightarrow} [bcc]$, and by $(\{\overline{E'}\}, \emptyset, \emptyset)$ the event structure corresponding to $[bcc] \overset{mpr}{\Rightarrow} [acc]$. The whole event structure corresponding to the computation subspace is represented in Figure 4.

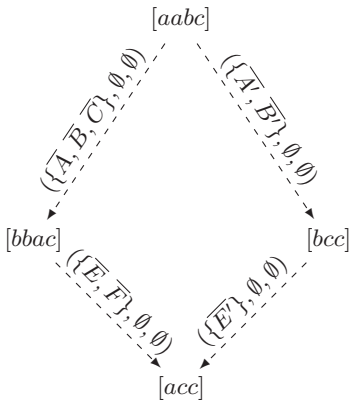


Fig. 4. Event structure corresponding to a membrane computation space over strings

The allocation of the resources to the rules is not longer given by an explicit event. It is given by the causality relation. For instance, in Example 4 we have $A, B, C < E, F$. The conflict relation is generated only by the mpr-steps starting from the same multiset (contents) and producing different new multisets (contents).

5 Conclusion

In this paper we study the event structure for membrane systems, considering both the causality and the conflict relations. We investigate how an event structures can be associated to an parallel evolution step. We found that the meaning of an event

depends on the algebraic structure used for the contents of membranes: string or multiset.

The construction of the event structures for the cases of priorities and promoters can be reduced to the case of maximal parallel rewriting. A computation step $w \overset{pri}{\Rightarrow} w'$ ($[w] \overset{pri}{\Rightarrow} [w']$) in the presence of priorities is the same with $w \overset{mpr}{\Rightarrow} w'$ ($[w] \overset{mpr}{\Rightarrow} [w']$) by taking into account only the rules of maximal priority applicable on w. Similarly, a computation step $w \overset{prom}{\Rightarrow} w'$ ($[w] \overset{prom}{\Rightarrow} [w']$) in the presence of promoters is the same with $w \overset{mpr}{\Rightarrow} w'$ ($[w] \overset{mpr}{\Rightarrow} [w']$) where the rules requiring promoters not in w are not considered.

References

1. O. Andrei, G. Ciobanu, D. Lucanu. A Structural Operational Semantics of the P Systems, In *Membrane Computing. WMC6*, Lecture Notes in Computer Science vol.3850, Springer, 32–49, 2006.
2. O. Andrei, G. Ciobanu, D. Lucanu. Operational Semantics and Rewriting Logic in Membrane Computing, *Electronic Notes of Theoretical Computer Science* 156:57–78, 2006.
3. O.Andrei, G.Ciobanu, D.Lucanu. A Rewriting Logic Framework for Operational Semantics of Membrane Systems. *Theoretical Computer Science* 373:163–181, 2007.
4. M. Nielsen, G. Rozenberg, P.S. Thiagarajan. Transition Systems, Event Structures, and Unfoldings. *Information and Computation* 118(2):191–207, 1995.
5. Gh. Păun. *Membrane Computing. An Introduction*. Springer, 2002.
6. V. Sassone, M. Nielsen, G. Winskel. Models for Concurrency: Towards a Classification. *Theoretical Computer Science* 170:297–348, 1996.
7. G. Winskel. Event structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, Lecture Notes in Computer Science vol.255, 325–392, Springer, 1987.
8. G. Winskel. An introduction to event structures. In *REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Lecture Notes in Computer Science vol.354, 364–397, Springer, 1989.