How a membrane agent buys goods in a membrane store*

Rudolf Freund**, Marion Oswald and Thomas Schirk

(Institute of Computer Languages; Vienna University of Technology Favoritenstr. 9, Wien, Austria)

Abstract In this paper we consider a specific model of membrane systems, i.e. membrane systems with attributes. In these systems, the information is placed at the membranes in form of attributes, no objects are considered inside the membranes except for other membranes. The membrane system with attributes evolves according to rules that compute new values for the attributes from the attributes assigned to the membranes involved in the rule. The model of membrane systems with attributes allows us to specify business transactions in a precise way and to simulate different models for such transactions with a suitable tool for membrane systems with attributes.

Keywords: model, membrane systems with attributes.

During the last years, the area of membrane systems has evolved rapidly. Many variants of membrane systems or, as they now are usually called, P systems have been introduced and their computational power turned out to be very high even with quite small ingredients. For a first comprehensive overview of P systems see the monograph[1], for the actual state of the art we refer the reader to the P systems webpage[2].

Although many variants of P systems turned out to be computationally complete, applications of P systems just recently became of more interest, for example see Ref. [3]. As membrane systems are based on biological processes taking place in cells or between cells, also using the exchange of objects through membranes, it is quite natural that many applications also deal with simulating processes happening in nature. On the other hand, it also seems interesting to model processes outside this area of biology and to consider processes in a completely different area, e.g. in Păun's work[1)] a first approach was made in that direction modelling business transactions by membrane systems; this idea was continued in Refs. [4] and [5]. In contrast to Ref. [5], where interesting theoretical results on the 117 computational power of numerical P systems were obtained, in this paper we concentrate on the simulation of business transactions by a suitable model of membrane systems. The model of membrane systems we are going to consider below works with attributes assigned to the membranes, yet without any objects inside the membranes themselves. The function of the membrane system is based on the interaction between membranes in such a way that suitable functions allow for computing the next configuration of the system from the attributes assigned to the membranes involved by a rule; the attributes we deal with theoretically are not restricted to multisets or strings over a finite alphabet, i.e. we also allow real numbers, etc. at the same time having in mind a finite representation of such objects and their handling by a finitely described mechanism as we can expect for a simulation of everyday-life processes, i.e. even when not stating this explicitly, we assume all functions to be computable. The model of membrane systems with attributes considered in this paper is also used in Ref. [6] to model various business transactions; some of the examples considered below can be found there explained in more detail.

The rest of the paper is organized as follows: in the next section, we describe the model of membrane systems with attributes used in this paper and give a first example; moreover, we elaborate some theoretical features of the model of membrane systems with attributes, and finally, we exhibit some ideas how we can use membrane systems with attributes for specifying business transactions. In the succeeding section we then describe an example showing how this model of P systems can be used to describe business processes, i.e. we describe how an agent is sent out to buy some goods. In the last section, we summarize the main ideas presented in this paper and give an outlook to future research.

* Supported by FWF Project (Grant No. T225-N04)

** To whom correspondence should be addressed. E-mail: rudi@emcc.at

1) Păun R. Producers, retailers, and their investments. A membrane computing approach. Submitted

1 P systems with attributes

For the background of formal language theory, we refer the reader to monographs as Ref. [7]. We just recall a few well-known notations: V^* is the free monoid generated by the alphabet V under the operation of concatenation and the empty string, denoted by λ, as unit element; $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Z}$, and $\mathbb{R}$ denote the set of positive integers (natural numbers), the set of non-negative integers, i.e. $\mathbb{N}_0=\mathbb{N}\cup\{0\}$, the set of integers, and the set of real numbers, respectively.

The theoretical model of membrane systems we introduce in this paper is inspired by the many models considered in the area of P systems so far (especially we again refer to Refs. [1] and [2]). We shall assume the reader to be a little bit familiar with this area, although the definitions given in what follows will be self-contained. We just mention that in P systems we have a hierarchical membrane structure; the outermost membrane is called skin membrane. Every membrane encloses its region possibly containing other membranes; a membrane whose region does not contain any other membrane is called elementary. Membranes are represented by matching parentheses [ ]; to indicate that the vector of attributes a is assigned to a membrane, we simply write $[a]$. In contrast to the common definitions, we shall not use labels for the individual membranes, because the rules are not assigned to membranes with specific labels, but instead can be applied to any combination of membranes fulfilling the structural constraints given in the rule and having the attributes in the domains of the functions specified in the rule.

We now introduce the model of membrane systems we shall use in this paper: A P system with attributes (a PSA for short) is a construct

$$\Pi = ((\mu_0, I_0), R),$$

where μ_0 is the initial membrane structure; I_0 is a function assigning the initial attribute to every membrane in μ_0, and R are rules of the following forms: $([a]\to[a'], f_1)$—evolution; the new attribute a' of the membrane is computed from the current attribute a by the function f_1; $([a][b]\to[a'][b'], f_1, f_2)$—joint evolution; the new attributes a' and b' of the two membranes being in the same region are computed from the current attributes a and b by the functions f_1 and f_2, respectively; $([a[b]]\to[a'[b']], f_1, f_2)$—parent/child evolution; the new attributes a' and b' of the outer (the parent) and the inner (the child) membrane, respectively, are computed from the current attributes a and b by the functions f_1 and f_2, respectively; $([a[b]]\to[a'], f_1)$—deletion of inner membrane; the inner membrane is deleted, the new attribute a' of the outer membrane is computed from the current attributes a and b by the function f_1; $([a[b]]\to[b'], f_1)$ (where the outer membrane must not be the skin membrane)—deletion of outer membrane; the outer membrane is deleted, the new attribute b' of the inner membrane is computed from the current attributes a and b by the function f_1; $([a][b]\to[a'], f_1)$—join; two membranes join to a single one, the new attribute a' of the single membrane is computed from the current attributes a and b by the function f_1; $([a]\to[a'][b'], f_1, f_2)$ (where the original membrane must not be the skin membrane)—generation; the new attributes a' and b' of the two membranes are computed from the current attribute a by the functions f_1 and f_2, respectively; $([a]\to[a'[b']], f_1, f_2)$—generation of inner membrane; the new attributes a' and b' of the original membrane and the newly generated inner membrane, respectively, are computed from the current attribute a by the functions f_1 and f_2, respectively; $([a]\to[b'[a']], f_1, f_2)$ (where the original membrane must not be the skin membrane)—generation of outer membrane; the new attributes a' and b' of the original membrane and the newly generated outer membrane, respectively, are computed from the current attribute a by the functions f_1 and f_2, respectively; $([a][b]\to[a'[b']], f_1, f_2)$—phagocytosis; one membrane is enclosed by another one, the new attributes a' and b' of the two membranes are computed from the current attributes a and b by the functions f_1 and f_2, respectively; $([a[b]]\to[a'][b'], f_1, f_2)$—exocytosis; the outer membrane expels the inner one, the new attributes a' and b' of the two membranes are computed from the current attributes a and b by the functions f_1 and f_2, respectively.

The rules of the system Π are applied in a sequential way (e.g. see Ref. [8]), i.e. only one rule is applied in one derivation step to get a new configuration. We should like to emphasize that a rule of the form (p, f_1) or (p, f_1, f_2) can be applied if and only if the involved membranes fulfill the hierarchical constraints given by p and the function(s) (f_1, f_2) are defined for the actual values of the attributes assigned to the membranes involved in the rule.

Some of the rules can be combined to construct new rules. For example, phagocytosis $([a][b]\to[a'[b']], f_1, f_2)$ and exocytosis $([a[b]]\to[a'][b'], f_1, f_2)$ can be seen as the sequence of rules $([a][b]\to[a''], f_1'')$, $([a'']\to[a'[b']], f_1', f_2')$ and $([a[b]]\to[a''], f_1'')$, $([a'']\to[a'][b'], f_1', f_2')$, respectively.

Obviously, we could also allow more complex rules, e.g. a combination of twice using a rule of the form $([a]\to[a'][b'], f_1, f_2)$ would yield the more complex rule

$$([a]\to[a'][b'][c'], f_1, f_2, f_3)$$

yet we shall explain such more complex rules only in the context where it helps to make the exposition of an algorithm more concise and readable.

1.1 A first example

As a first example we specify a PSA comparing two numbers n and k (from $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{N}_0$, or even from $\mathbb{R}$; from a formal language point of view, the most reasonable choice is $\mathbb{N}_0$), given as attributes of two elementary membranes inside the skin membrane, and yielding the answer of the comparison as an attribute of the skin membrane ($1>2$ means that $n>k$, $2>1$ indicates that $k>n$, and $1=2$ says that both numbers are equal).

$$\Pi = ((\mu_0, I_0), R),$$
$$(\mu_0, I_0) = [(0)[(1,n)][(2,k)]],$$
$$R = \{([a[b]]\to[a'], f_1)\}$$

with

$$f_1:((\{0\})\times(\{1\}\times\mathbb{N}))\cup((\{0\}\times\mathbb{N})\times(\{2\}\times\mathbb{N}))\to(\{0\}\times\mathbb{N})\cup(\{0\}\times\{1>2, 2>1, 1=2\}),$$
$$f_1(0,1,n) = (0,n),$$
$$f_1(0,n,2,k) = \begin{cases}(0,1>2) \text{ if } n>k,\\ (0,2>1) \text{ if } k>n,\\ (0,1=2) \text{ if } n=k.\end{cases}$$

Each of the three membranes carries an attribute vector whose first component identifies it as the skin membrane (0) or as one of the inner membranes carrying the first (1) and the second (2) number to be compared. The comparison is done in two steps: in the first step, the skin membrane takes over the number n from (the second component of the attribute vector of) the first inner membrane and deletes this membrane; in the second step, this number n is compared with the number k assigned (as the second component of the attribute vector) to the second inner membrane, which is deleted now, whereas the result of the comparison of n and k appears as the second component of the final attribute vector of the skin membrane.

The same task of comparing two numbers can be accomplished by using one more complex rule that allows the skin membrane to interact with both inner membranes at the same time:

$$\Pi = ((\mu_0, I_0), R),$$
$$(\mu_0, I_0) = [(0)[(1,n)][(2,k)]],$$
$$R = \{([a[b][c]]\to[a'], f_1)\}$$

with

$$f_1:(\{0\})\times(\{1\}\times\mathbb{N})\times(\{2\}\times\mathbb{N})\to(\{0\}\times\{1>2, 2>1, 1=2\}),$$
$$f_1(0,1,n,2,k) = \begin{cases}(0,1>2) \text{ if } n>k,\\ (0,2>1) \text{ if } k>n,\\ (0,1=2) \text{ if } n=k.\end{cases}$$

For both cases, the initial configuration and the final configuration can be seen in Fig. 1 (the set $\{1>2, 2>1, 1=2\}$ in the second component in the attribute vector of the skin membrane of the final configuration indicates that exactly one element of that set is obtained).

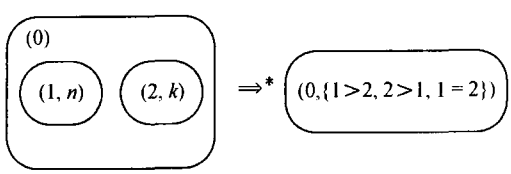

Fig. 1. Order relation between two numbers n and k.

1.2 Theoretical remarks

The model of P systems with attributes introduced above is very powerful from a theoretical point of view, because we have made no severe restrictions on the functions used to compute the new attributes in the rules. Yet even when restricting the power of these functions, PSA remain very powerful; as an example, let us consider PSA working on numbers from $\mathbb{N}_0$. Then without sophisticated functions, we can easily simulate register machines:

A deterministic n-register machine is a construct $M=(n, R, l_0, l_h)$, where n is the number of registers, R is a finite set of instructions injectively labelled with elements from $Lab(M)\subset\mathbb{N}_0$, l_0 is the initial/start label, and l_h is the final label; the instructions are of the following forms: $l_1:(A(r), l_2)$ (ADD instruction) Add 1 to the contents of register r

and proceed to the instruction (labelled with) l_2; l_1: $(S(r), l_2, l_3)$ (SUB instruction) If register r is not empty, then subtract 1 from its contents and go to instruction l_2, otherwise proceed to instruction l_3; l_h: halt (HALT instruction) Stop the machine. The final label l_h is only assigned to this instruction.

A register machine can easily be simulated by a PSA, for example, in the following way: The registers are represented as elementary membranes carrying the attributes $(i, c(i))$ where i is the number of the register and $c(i)$ is its contents; the finite control is situated at the skin membrane with the attribute vector $(0, l)$ where l is the current instruction label. Hence, we construct the PSA

$$\Pi = ((\mu_0, I_0), R')$$

with the initial configuration

$$[(0, l_0)[(1, c_0(1))]\cdots[(n, c_0(n))]]$$

where $c_0(i)$, $1 \leqslant i \leqslant n$, are the initial values of the registers i, and

$$\begin{aligned} R' = \{&([a[b]] \to [a'[b']], f_1, f_2) \\ &\mid l_1: (A(r), l_2) \in R, \\ &f_1(0, l_1, r, k) = (0, l_2), f_2(0, l_1, r, k) \\ &= (r, k+1)\} \\ \cup \{&([a[b]] \to [a'[b']], f_1, f_2) \\ &\mid l_1: (S(r), l_2, l_3) \in R, \\ &f_1(0, l_1, r, k) = (0, l_2), f_2(0, l_1, r, k) \\ &= (r, k-1) \text{ for } k > 0, \\ &f_1(0, l_1, r, 0) = (0, l_3), \\ &f_2(0, l_1, r, 0) = (r, 0)\}, \end{aligned}$$

where all f_i are functions

$$\begin{aligned} f_i: &((\{0\} \times Lab(M)) \times (\{1, \cdots, n\} \times \mathbb{N}_0)) \\ &\to ((\{0\} \times Lab(M)) \times (\{1, \cdots, n\} \times \mathbb{N}_0)) \end{aligned}$$

It is obvious that the PSA Π constructed in that way simulates the given register machine M; moreover, Π halts (which means that no rule can be applied anymore) if and only if M halts. In that way, Π can compute functions as M does, and the result can be seen as the second component of the attribute vector assigned to the membrane corresponding to the respective register in the final configuration. As can be seen from the construction of the rules, only rules of one kind are needed, i.e. of the form $([a[b]] \to [a'[b']], f_1, f_2)$—parent/child evolution. Moreover, the functions are quite simple, too, the main power lying in the possibility to distinguish between the case of an empty register and a non-empty register.

Obviously, this variant for simulating register machines is not the only one that can simulate a variant of P systems well known from the literature, yet we do not follow this line anymore in this paper. Instead, we turn our attention to use PSA for specifying specific processes in business life.

1.3 Using PSA for economic models

A relatively new approach of economic problems is to focus on the single individual and not on the whole mechanism. There exist many economic procedures that calculate only the totality of economic processes, a good example being the price calculation of companies: the price setting depends on the demands and needs of the customers and, of course, also on the price of the material needed to produce an article; yet the demand usually is not created by a single person, but by many customers. Many business analysts try to find out why the homo economicus acts how he acts (e.g. see Ref. [9]). A big part of these economic studies are computer simulations.

We now establish P Systems with attributes as a formal specification tool for business processes. The membranes symbolize individuals, but also companies and their goods, etc. The interactions between customers and vendors or companies are captured by the change of the membrane structure as well as by the functions in the rules. The attributes assigned to the membranes can represent not only the identity and the function of a membrane, but also features like the colour of the hair of a person or the price of a good. Hence, in the succeeding section we shall exhibit a depictive example how P systems with attributes can model specific business transactions. Moreover, we should like to point out that we are developing a simulation tool based on the theoretical model described in this paper for running randomized simulations of some specific business transactions in the way economic studies are carried out as computer simulations in the traditional way.

2 How a membrane agent buys membrane chocolate in a membrane store

In this section we describe how the process of buying goods can be formulated within the framework of P systems with attributes.

Within the closed world represented by the skin membrane, we consider membranes for customers,

stores, agents, and goods (e.g. chocolate). In the representation of the example given below, we only list those attributes that are necessarily involved in the transaction we want to describe. We start with the following initial configuration (see Fig. 2):

$[0(0)[(\text{store}, money_{store})\ [(\text{chocolate}, price)]]$
$[(\text{customer}, money_{customer}, \text{happy})]]$

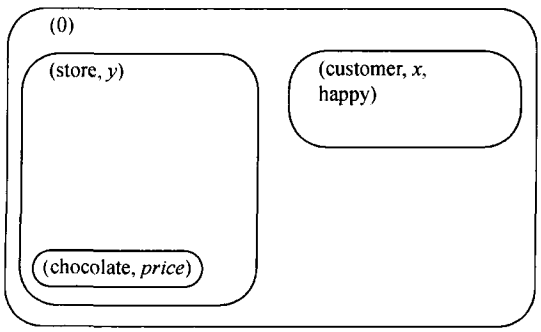

Fig. 2. Initial membrane structure.

The skin membrane will never change, it just carries the attribute (0) showing that it is the skin membrane. Within this closed world, we start with a store having some budget $money_{store}(y)$ and offering chocolate at a price of *price*. The customer has some budget $money_{customer}(x)$ to buy goods, and the status (initially being happy) indicates his needs. Obviously, this is only a small part of the closed world, e.g. usually we would have not only one piece of chocolate in the store, but we restrict ourselves to depict those objects which will be handled during the buying process. If, for some reasons from outside, the customer's status changes from happy to need_chocolate, then (see Fig. 3) the customer creates an agent (to goto_store) for buying a piece of chocolate costing at most p (we use $\mathbb{M}$ for the set of money values we are dealing with, e.g. decimal number representations with a fraction of two digits, up to a maximal amount *max*):

$([c] \to [c'][a'], f_1, f_2),$
$f_1: (\{\text{customer}\} \times \mathbb{M} \times \{\text{need\_ chocolate}\})$
$\to (\{\text{customer}\} \times \mathbb{M} \times \{\text{wait\_ for\_ chocolate}\}),$
$f_2: (\{\text{customer}\} \times \mathbb{M} \times \{\text{need\_ chocolate}\})$
$\to (\{\text{agent}\} \times \mathbb{M} \times \{\text{buy\_ chocolate}\}$
$\times \{\text{goto\_ store}\}),$
$f_1(\text{customer}, x, \text{need\_ chocolate})$
$= (\text{customer}, x - p, \text{wait\_ for\_ chocolate}),$
$f_2(\text{customer}, x, \text{need\_ chocolate})$
$= (\text{agent}, p, \text{buy\_ chocolate}, \text{goto\_ store}).$
$([s][a] \to [s'[a']], f_1, f_2),$
$f_1: ((\{\text{store}\} \times \mathbb{M})$
$\times (\{\text{agent}\} \times \mathbb{M} \times \{\text{buy\_ chocolate}\}$
$\times \{\text{goto\_ store}\})) \to (\{\text{store}\} \times \mathbb{M}),$
$f_2: ((\{\text{store}\} \times \mathbb{M}) \times (\{\text{agent}\}$
$\times \mathbb{M} \times \{\text{buy\_ chocolate}\} \times \{\text{goto\_ store}\}))$
$\to (\{\text{agent}\} \times \mathbb{M} \times \{\text{buy\_ chocolate}\}$
$\times \{\text{in\_ store}\}),$
$f_1(\text{store}, y, \text{agent}, p, \text{buy\_ chocolate}, \text{goto\_ store})$
$= (\text{store}, y),$
$f_2(\text{store}, y, \text{agent}, p, \text{buy\_ chocolate}, \text{goto\_ store})$
$= (\text{agent}, p, \text{buy\_ chocolate}, \text{in\_ store}).$

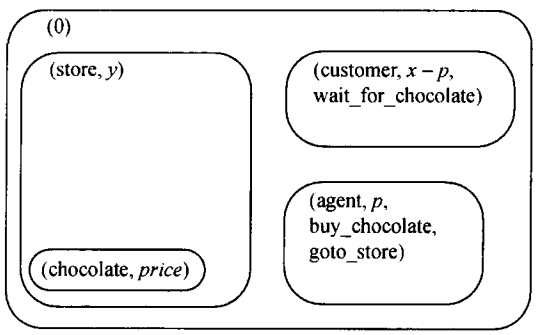

Fig. 3. Creation of an agent.

The agent now enters the store (see Fig. 4) and checks whether he can afford the chocolate offered here provided there is some chocolate available. He buys the chocolate if available in the store at a price of no more than the money p he has got from the customer (see Fig. 5); we here do not consider the frustrating case that the agent is not able to buy chocolate.

$([a][b] \to [a'[b']], f_1, f_2),$
$f_1: ((\{\text{agent}\} \times \mathbb{M} \times \{\text{buy\_ chocolate}\}$
$\times \{\text{in\_ store}\}) \times (\{\text{chocolate}\} \times \mathbb{M}))$
$\to (\{\text{agent}\} \times \mathbb{M} \times \mathbb{M} \times \{\text{chocolate\_ bought}\}),$
$f_2: ((\{\text{agent}\} \times \mathbb{M} \times \{\text{buy\_ chocolate}\}$
$\times \{\text{in\_ store}\}) \times (\{\text{chocolate}\} \times \mathbb{M}))$
$\to (\{\text{chocolate}\}),$
$f_1(\text{agent}, p, \text{buy\_ chocolate}, \text{in\_ store}, \text{chocolate}, price)$
$= (\text{agent}, p, price, \text{chocolate\_ bought}),$
$f_2(\text{agent}, p, \text{buy\_ chocolate}, \text{in\_ store}, \text{chocolate}, price)$
$= (\text{chocolate}).$
$([s[a]] \to [s'][a'], f_1, f_2),$
$f_1: ((\{\text{store}\} \times \mathbb{M}) \times (\{\text{agent}\}$
$\times \mathbb{M} \times \mathbb{M} \times \{\text{chocolate\_ bought}\}))$
$\to (\{\text{store}\} \times \mathbb{M}),$

f_2:(({|store|} × $\mathbb{M}$) × ({|agent|}
× $\mathbb{M}$ × $\mathbb{M}$ × {|chocolate_ bought|}))
→ ({|agent|} × $\mathbb{M}$ × {|got_ chocolate|}),
f_1(store, y, agent, p, *price*, chocolate_ bought)
= (store, y + *price*),
f_2(store, y, agent, p, *price*, chocolate_ bought)
= (agent, *p-price*, got_ chocolate).

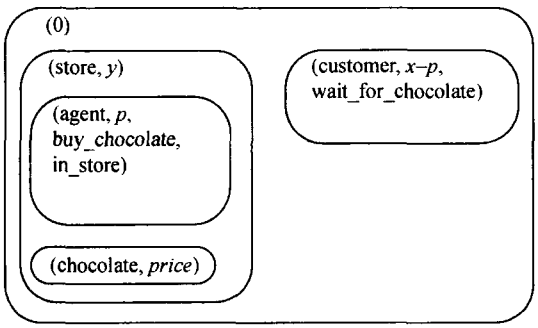

Fig. 4. The agent enters the store.

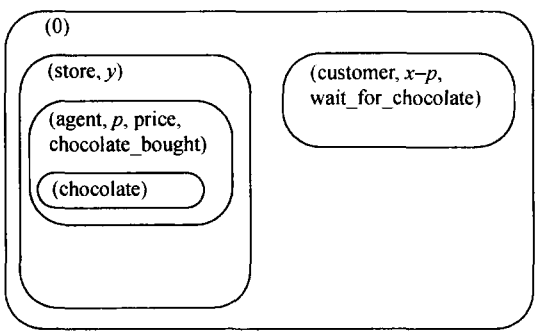

Fig. 5. The agent buys chocolate.

When leaving the store (see Fig. 6), the agent pays for the chocolate and returns back to the customer (see Fig. 7).

([c][a] → [c'[a']], f_1, f_2),
f_1:(({|customer|} × $\mathbb{M}$ × {|wait_ for_ chocolate|})
× ({|agent|} × $\mathbb{M}$ × {|got_ chocolate|}))
→ ({|customer|} × $\mathbb{M}$ × {|get_ chocolate|}),
f_2:(({|customer|} × $\mathbb{M}$ × {|wait_ for_ chocolate|})
× ({|agent|} × $\mathbb{M}$ × {|got_ chocolate|}))
→ ({|agent|} × {|chocolate|}),
f_1(customer, z, wait_ for_ chocolate, agent,
q, got_ chocolate)
= (customer, z + q, get_ chocolate),
f_2(customer, z, wait_ for_ chocolate, agent,
q, got_ chocolate)
= (agent, chocolate).

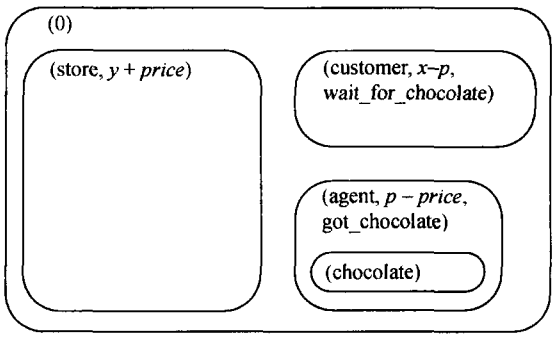

Fig. 6. The agent pays and leaves the store.

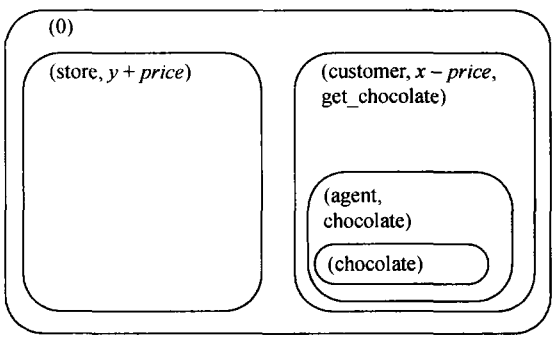

Fig. 7. The agent returns back to the customer.

Finally, the agent releases the chocolate (see Fig. 8), and after having eaten the chocolate, the customer is happy again (see Fig. 9).

([a[b]] → [b'], f_1),
f_1:(({|agent|} × {|chocolate|})
× ({|chocolate|})) → ({|chocolate|}),
f_1(agent, chocolate, chocolate)
= (chocolate).
([c[b]] → [c'], f_1),
f_1:(({|customer|} × $\mathbb{M}$ × {|get_ chocolate|})
× ({|chocolate|})) → ({|customer|}
× $\mathbb{M}$ × {|happy|}),
f_1(customer, z, get_ chocolate, chocolate)
= (customer, z, happy).

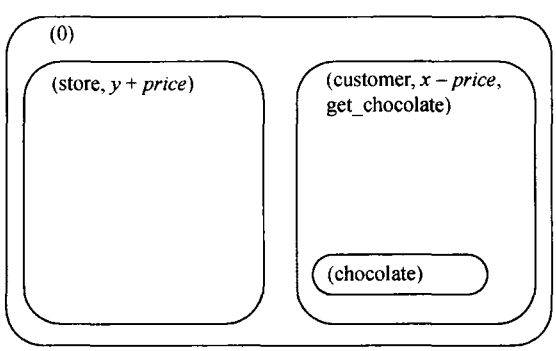

Fig. 8. The chocolate is released.

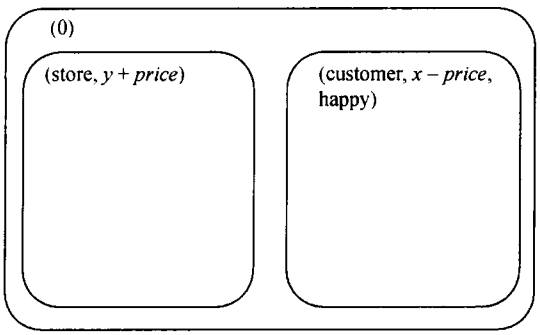

Fig. 9. The customer is happy again.

3 Summary and future research

In this paper we have proposed the model of membrane systems with attributes as a specification tool for business processes. From a theoretical point of view, this model is a very rigorous one, the systems evolving only by eventually changing the membrane structure and by the interaction of the attributes assigned to the membranes; a rule only changes the attributes of the membranes involved in the rule, no objects are involved inside the membranes.

For business transactions like auctions different strategies can be found in the literature. Hence, it is of some interest to specify these variants within the same formal framework and then to let some simulations run that allow for comparing the possible behaviour of these variants. As the membrane systems with attributes work in the sequential derivation mode, the implementation of this model works with a reasonable time behaviour, and we are going to use it for testing various strategies for business transactions like auctions as well as for running simulations of different strategies for other business transactions.

References

1 Păun Gh. Membrane Computing—An Introduction. Berlin: Springer, 2002
2 The P systems webpage: http://psystems.disco.unimib.it
3 Ciobanu G, Păun Gh and Pérez-Jiménez MJ. Applications of Membrane Computing. Berlin: Springer, 2005
4 Păun Gh and Păun R. Membrane computing as a framework for modeling economic processes. Proc. SYNASC 05, IEEE Press, 2005, 11—18
5 Păun Gh, Păun R. Membrane computing and economics: Numerical P systems. Fundamenta Informaticae, 2006, 73(1—2): 213—227
6 Schirk T. Modelling Specific Business Transactions by Membrane Systems with Attributes. Diploma thesis, draft, 2006
7 Salomaa A, Rozenberg G. Handbook of Formal Languages. Berlin: Springer, 1997
8 Freund R. Asynchronous P systems. In: Membrane Computing. International Workshop WMC5. Lecture Notes in Computer Science 3365, Springer (2005), 36—62
9 Varian HR. Intermediate Microeconomics, 5th ed. New York: W. W. Norton & Company Inc., 2000