Technical Report CoSBi 02/2006

The π-calculus with biological transactions

Federica Ciocchetta
University of Trento
fedecioc@dit.unitn.it

Corrado Priami
The Microsoft Research - University of Trento Centre for Computational and Systems Biology
priami@cosbi.eu

The π-calculus with biological transactions

Federica Ciocchetta
University of Trento
fedecioc@dit.unitn.it

Corrado Priami
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
priami@cosbi.eu

Abstract

In this work we extend the π-calculus with a simple class of transactions, suitable to model multi-reactant, multi-product chemical reactions. Some examples are reported to validate our extension.

1 Introduction

In the recent years, process algebras have been widely applied to model biological models [14, 11, 13, 16, 10, 6]. One of the most used process algebras in this field is the π-calculus [9], a formal language originally developed for specifying concurrent computational systems. In such systems, multiple processes interact each other by synchronized pair-wise communication on complementary channels and modify each other by transmitting channel from one process to another. In using the π-calculus for modeling biological reactions, any interaction (for instance the transformation of one molecule into another one, the formation of a complex, ...) is represented as a (synchronous) communication between two biological entities. A problem arises when reactions with more than two reactants are considered. These reactions are rare in nature, but are often used in biological models as abstractions of complex phenomena. Generally, they represent sequence of elementary steps, whose details are unknown or not important. A possible approach to deal with these reactions is to decompose them into a sequence of two-reactant reactions. For instance, the three-reactant reaction $R_1 + R_2 + R_3 \rightarrow P$ may be decomposed into the two elementary reactions $R_1 + R_2 \rightarrow complex(R_1, R_2)$ and $complex(R_1, R_2) + R_3 \rightarrow P$. The new element $complex(R1, R2)$ represents the intermediate complex of the two reactants R_1 and R_2. Following this approach, some problems arise. First, different orders of reactants are possible and there are $\frac{n!}{2}$ ways to decompose a reaction, where n is the number of reactants (and modifiers). Furthermore, it can happen that after the formation of the intermediate complex, the third reactant misses, leading to a deadlock. A possible way to solve this problem is to consider the first reaction as reversible (i.e.

$R1 + R2 \leftrightarrow complex(R1, R2)$), but this solution can lead to undesired behaviors.

Here a different approach is adopted. We propose to introduce *transactions* to model complex reactions. Generally speaking, a transaction is a component of a distributed system which must be executed as if were a single atomic action. It should not be interrupted at intermediate steps and satisfies safety and failure properties. They are used to define traditional transactions in databases or composition of subtasks in web service technologies. Recently there are several attempts to model transactions formally by using process algebras [1, 7, 3]. In [1] the πt-calculus is presented: it is an extended version of the asynchronous π-calculus to deal with long time transactions and offers failure handlers when interruptions are met. Another extension of the asynchronous π-calculus with long-time transactions, called *web-*π, is introduced in [7]. CSP is the process algebras adopted in [3] to model long-running transactions with traces.

In this work we focus on the *synchronous* π*-calculus* and enrich it with transactions. They have to satisfy some simple properties that are suitable for modeling complex reactions. We call these transactions *biological transactions* to distinguish them from more complex kinds. After a biological transaction has started, it executes in isolation and ends successfully. Besides, the result is visible only when the transaction ends. These properties can be formalized by using the concepts of *atomicity* and *serializability*. The former is summarized as *all or nothing*: a transaction is executed and finally commits or does nothing. In this work, this property is guaranteed only for a specific kind of systems, called *simple biological systems*. This limitation permits to take the calculus as simple as possible. The latter expresses the fact that different activities have the same effect whether they are executed in sequence or in parallel.

In the next section a brief description of the use of the π-calculus to model biological reactions is presented. In the section 3 the extended π-calculus with biological transactions is described and some results about atomicity and serializability are reported briefly. After that, some simple examples about the use of transactions in biology is shown (section 4). Finally, section 5 contains some final remarks and conclusions.

2 The π-calculus for biological models

This section presents the main concepts about the description of biological models using the π-calculus. Generally speaking, this process algebra has been defined to model mobile communication systems, based on the concept of name-passing. This means that communicating processes can exchange names over channels and, as result, they may change interconnection topology. For the description of the syntax and the semantics of the standard

π-calculus see for instance [9, 15].

Recently the π-calculus has been applied to represent and analyze a great variety of biological models, concerning for instance metabolic pathways, signal transduction networks and transcriptional circuits [14, 12, 8, 6]. For quantitative description and analysis the stochastic version has been considered [11].

The underlying idea of the application of the π-calculus to biology is the so-called *molecule-as-computation* abstraction [14]: it is possible to associate each biological entity and interaction to a specification in the calculus.

2.1 The molecule-as-computation abstraction

According to the molecule-as-computation abstraction, a biochemical network can be modeled by a computational system, composed of a collection of concurrent processes, representing biological entities (for instance genes, proteins or small molecules). Compartments are represented by restricted communication scopes. Any biological interaction corresponds to a communication among the molecules involved. The interaction elements are viewed as channels or communication ports on the molecule and an interaction event may occur only when a complementary pair of channels is shared by the interacting molecules. The fundamental rule in this translation is the one that express the communication among two parallel processes. Following the ideas above, the formation of a complex may be modeled by two processes in parallel, representing the two (simple) molecules. These processes communicate over a channel x_{12} and along it a (private) channel x_c can be sent from the first molecule to the other one. The two molecules may continue to exclusively communicate on that channel and are thus linked. The specifications of this example are:

$$S \;=\; P_{R1} \mid P_{R2} \quad P_{R1} = \nu x_c \overline{x_{12}}\langle x_c\rangle.R \quad P_{R2} = x_{12}(w).Q$$

After the communication we obtain the complex $\nu x_c(R \mid Q\{^{x_c}/_w\})$. It is worth noting that if a generic reaction $R_1 + R_2 \rightarrow P$ is given, without any information about the structure of the biological entities involved, we may simplify the specifications by using a communication between the processes without the exchange of names.

Three-reactant reaction: $R_1 + R_2 + R_3 \rightarrow P$

This reaction may be decomposed into the following two (sequential) reactions:

$$R_1 + R_2 \;\rightarrow\; complex(R_1, R_2) \tag{1}$$

$$complex(R_1, R_2) + R_3 \;\rightarrow\; P \tag{2}$$

The reactants R_1 and R_2 communicate yielding to the $complex(R_1, R_2)$ that may communicate with the third reactant R_3 to produce. A representation in π-calculus is the following one:

$$\begin{array}{rcl} S & = & P_{R_1} \mid P_{R_2} \mid P_{R_3} \\ P_{R_1} & = & \overline{x_{12}}\langle\rangle.\overline{x_{123}}\langle\rangle.P_P \quad P_{R_2} = x_{12}().\ \mathsf{nil} \quad P_{R_3} = \overline{x_{123}}\langle\rangle.\ \mathsf{nil} \end{array}$$

The problem in this approach is that the reaction may start but may stop at the intermediate step (representing the formation of $complex(R_1, R_2)$) if the third reactant R_3 misses. A possible way to avoid this is to add the reaction $complex(R_1, R_2) \rightarrow R_1 + R_2$ (i.e. the reaction (1) is supposed to be reversible). In this case, after the first communication, there are two possibilities: the former is a communication between the complex and the the third reactant R_3 to obtain the product P, the latter is a communication inside the complex to come back to the two reactants R_1 and R_2.

$$\begin{array}{rcl} S & = & P_{R_1} \mid P_{R_2} \mid P_{R_3} \\ P_{R_1} & = & \overline{x_{12}}\langle\rangle.\,P_{complex} \quad P_{R_2} = x_{12}().\ \mathsf{nil} \quad P_{R_3} = x_{123}().\ \mathsf{nil} \\ P_{complex} & = & \overline{x_{123}}\langle\rangle.P_P + (\overline{x_{complex}}\langle\rangle.P_{R_1} \mid x_{complex}().P_{R_2}) \end{array}$$

It is worth noting that some undesired behaviors may be obtained. For instance it is possible to have an infinite sequence of transitions in which there is the formation of the complex followed by its decomplexation. Besides, there is also here the problem to decide the order of reactant interactions.

3 The π-calculus with biological transactions

In this section we show how to enrich the π-calculus with transactions. In this way it would be possible to model reactions with more than two reactants as they were atomic actions. A simple version of the *synchronous π-calculus* is considered as the basic calculus to extend.

3.1 Biological transactions: properties

Transactions are the basic mechanisms for modeling database transactions and for composing web-services in orchestration and choreography languages. Recently, some proposals to enrich the π-calculus and other process algebras with transactions to formally model web service complex activities have been defined [1, 7, 3]. Different properties and features must considered according to the field of application. For instance, it may be opportune to introduce compensation processes for activity failure or to use nested processes or to add mechanisms to deal with timeouts. In modeling biological phenomena, the transactions need to satisfy some simple properties. It would be desirable that the transaction does not stop at intermediate steps (for the lack

of opportune processes) and it works as an atomic action. Besides, it is requested that the reaction results would be visible only after the transaction has ended. Formally, the properties that are of interest for the transactions are:

- *Atomicity*: a sequence of transaction actions is performed either in its entirety or else not at all.
- *Serializability (or isolation)*: A transaction should not make its effect visible to other transactions until it commits. The concurrent transactions are serializable, they appear to occur one-at-a-time.

The former property is limited only to specific (but widely-used) systems to make the calculus as simple as possible. In these systems, called *simple biological systems*, it is not possible to have processes that may execute infinitely in parallel with a transaction. Finally, neither compensation processes nor nested transactions nor timeout mechanisms are used. We refer to the transactions described here as *biological transactions*, to distinguish them from database and web service ones.

3.2 Syntax and Semantics

A process (indicated with the capital letter P, Q, R,...) is defined in the π-calculus with biological transactions by the following syntax:

$$P ::= \mathsf{nil} \mid \pi.\, P \mid P|P \mid \nu y P \mid !\, P \mid t[P]$$

$$\pi ::= x(z) \mid \overline{x}\langle z\rangle \mid start(t)[P] \mid end(t)$$

where we suppose a countable infinite set of (channel, message) names $\mathcal{N}$ (ranged over lower-case by letters *x, y, z, ...*) and a countable set of transaction names $\mathcal{T}$ (ranged over by lower-case letters *t, t', t"*, ...), with $\mathcal{T} \cap \mathcal{N} = \emptyset$. The processes generated by the grammar above differ from standard π-calculus processes in *t[P]* and in the prefixes *start(t)*[P] and *end(t)*. These elements are intended to represent transactions. All other processes have the same meaning as in standard π-calculus. For details see [9, 15]. As for the new terms, the process $t[P]$ indicates the transaction of name t described by the *internal process* P. The other possible processes outside the transaction are called *external* (to the transaction) processes. As for prefixes, the first two ones represent the usual input and the output prefixes, respectively. The prefix $start(t)[P]$ is used to indicate the start of a transaction of name *t*. The process P inside the square brackets indicates a (structural congruent) external process that is necessary for the reaction to start. This point will be explained better when the semantics rules are described. In the same way, the prefix $end(t)$ indicates that the transaction t ends successfully. The π-calculus definitions of *name substitution* (in $\mathcal{N}$) and of *free* and

bound names (denoted by fn(-) and bn(-), respectively) are extended to the processes generated by the above syntax in the obvious way. Substitution functions for *transaction names* (in $\mathcal{T}$) are also defined similarly.

An operational reduction semantics that makes use of both a *structural congruence* and *reduction relation* is given.

The *structural congruence* (indicated by $\equiv$) is the smallest relations which satisfy the following laws:

1. $P_1 \equiv P_2$ provided P_1 is an α-converse of P_2
2. $P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$, $P_1 \mid P_2 \equiv P_2 \mid P_1$, $P \mid \mathsf{nil} \equiv P$
3. $\nu z\, \nu w\, P \equiv \nu w\, \nu z\, P$, $\nu z\ \mathsf{nil} \equiv \mathsf{nil}$, $\nu y\, (P_1 \mid P_2) \equiv P_1 \mid \nu y\, P_2$ provided $y \notin \mathsf{fn}(P_1)$
4. $!\, P \equiv P \mid !\, P$
5. $t[P_1] \equiv t'[P_2]$ provided that $P_1 \equiv P_2$
6. $start(t)[P_0].P_1 \equiv start(t')[P_0'].P_1'$ provided that $P_0 \equiv P_0'$ and $P_1 \equiv P_1'$

The first four points describe the usual congruence laws of the π-calculus. As for the new rules, the law (5) expresses the congruence among transactions: two transactions are congruent if the respective internal processes P_1 and P_2 are congruent. The name used for the transaction is only indicative, the important is that a fresh name is used every time a reaction start or subjected to a substitution on transaction names. The law (6) expresses the congruence among two processes with a start prefix. They are congruent if the respective processes used to describe them are congruent. It is worth noting that in this case we do not consider the rule $t[\pi.P] \equiv \pi.t[P]$, with π an input or output action. An explanation of this choice is reported below when the axiom comm is described.

The *reduction relation* $\longrightarrow$ is the smallest over processes obtained by applying the axioms and rules in Table 1.

The first four rules are the standard π-calculus ones. Here we report some observations on comm, for the other ones see ([9, 15]). The rule (comm) describes the communication among two parallel processes. Since we cannot transform $t[\pi.P]$ into $\pi.t[P]$, the communication is possible only between two standard processes both outside or both inside the transaction. Indeed, if we supposed $t[\pi.P] \equiv \pi.t[P]$, it would be possible to make a communication between the internal processes of the transaction and processes outside or among two different transactions (the atomicity and isolation properties fail).

The axiom (tstart) describes the start of a transaction. The process P_1 indicates some of the actions involved. We introduce the process P_0 inside the square brackets to specify the elements to block, since necessary for the execution of the transaction. In this way, after the starting of the transaction,

(comm)	$x(w).\,P_1 \mid \overline{x}\langle z\rangle.\,P_2 \mid P_3 \longrightarrow P_1\{z/w\} \mid P_2 \mid P_3$
(par)	$\dfrac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$
(res)	$\dfrac{P \longrightarrow P'}{\nu w\, P \longrightarrow \nu w\, P'}$
(struct)	$\dfrac{P_1 \equiv P_1' \qquad P_1' \longrightarrow P_2' \qquad P_2 \equiv P_2'}{P_1 \longrightarrow P_2}$
(tstart)	$P_0' \mid start(t)[P_0].P_1 \longrightarrow t'[P_1\{t'/t\} \mid P_0]$ where t' is fresh and $P_0 \equiv P_0'$
(tend)	$t[end(t).P_1 \mid P_2] \longrightarrow P_1 \mid P_2$
(tred)	$\dfrac{P \longrightarrow P'}{t[P] \longrightarrow t[P']}$

Table 1: Axioms and rules for the reduction relation for π-calculus with biological transactions.

they cannot be consumed by other external processes. When an external process P_0' structural congruent to the process P_0 is found, the transaction may start and P_0' is put inside it. Finally, the result is a transaction of name t' (t' must be a fresh name) described by the internal process $P_1\{^{t'}/_t\}|P_0'$. The axiom (tend) describes the successful end of the transaction t. As result of the transaction, the process P_1 (after it) and the other internal processes in parallel are given. The last rule expresses how the transaction is reduced: the reductions of the transaction $t[P]$ is described by the reductions of the internal process P.

We do not need an abort action as we assume that the internal process of a transaction is *well-defined* (i.e. it reduces to a parallel composition of processes among which the $end(t).P$ is presented and so the transaction ends successfully).

3.3 Atomicity and Serializability

Some results are reported showing that biological transactions satisfy the desired atomicity (limited to *simple biological systems*) and serializability properties. Indeed, they are general criteria for proving the correctness of the transactions. We follow the same approach proposed in [2] and adapt it to our case. First of all, we add labels to each transaction to show what kind of action it represents. The labels (ranged over by l, l', ...) are defined by the set $L = \{comm(x), t : comm(x), t : start, t : end\}$ where $x \in \mathcal{N}$ and $t \in \mathcal{T}$. The first label represents a communication between two processes outside a transaction along the channel x, while the others refer to actions involving a transaction of name t. They are the communication between two processes along x inside the transaction t, the start and the end of t. The reduction relation defined in 3.2 may be extended by considering these labels in the obvious way. Two functions are introduced to show the transaction names in a given label and to denote the set of transactions (processes of the form $t[P]$) in a process P, respectively. The former (from L to $\mathcal{T}$) is defined as $tr(t : comm(x)) = tr(t : start) = tr(t : end) = t$ and $tr(comm(x)) = \perp$ (the symbol $\perp$ stays for undefined). The latter one (from processes to $\mathcal{T}$) is defined as $actT(P) = \{t|t[P_0]$ for some P_0 is a subterm of $P\}$. As follows we use the notation $P_0 \xrightarrow{\sigma} P_n$ to denote the sequence $P_0 \xrightarrow{l_1} P_1 \xrightarrow{l_2} P_2... \xrightarrow{l_n} P_n$, where $\sigma = l_1...l_n$. The transaction sequence $P \xrightarrow{\sigma} P'$, with $actT(P) = actT(P') = \emptyset$, is serialized iff $l_i = t : comm(x)$ or $l_i = t : start$ implies $l_{i+1} = t : comm(x)$ or $l_{i+1} = t : end$ for $i = 1, ...(n-1)$. The internal processes are supposed to be *well-defined*. At this point we claim the following results:

Lemma 1. *If* $P \xrightarrow{l_1} P' \xrightarrow{l_2} P''$, *with* $tr(l_1) \neq tr(l_2)$ *and* $l_1 = comm(x)$ *then there exists* P''' *such that* $P \xrightarrow{l_2} P''' \xrightarrow{l_1} P''$.

This lemma shows that two transactions involving any two actions (not in the same transaction) are permutable.

Proof. In the proof we consider the following result deriving from the reduction rules:

Lemma 2. *Given* $P \xrightarrow{l} P'$*, we have that*

- *if* $l = comm(x)$ *then* $P \equiv x(z).P_1|\overline{x}\langle w\rangle.P_2|Q$ *and* $P' \equiv P_1\{^w/_z\}|P_2|Q$
- *if* $l = t : comm(x)$ *then* $P \equiv t[x(z).P_1|\overline{x}\langle w\rangle.P_2|Q]|R$ *and* $P \equiv t[P_1\{^w/_z\}|P_2|Q]|R$
- *if* $l = t : start$ *then* $P \equiv start(t')[P_0].P_1|Q$ *and* $P' \equiv t'[P_0|P_1\{^{t'}/_t\}]|Q$
- *if* $l = t : end$ *then* $P \equiv t[end(t).P_0|P_1]|Q$ *and* $P' \equiv P_0|P_1|Q$

Considering lemma 3.3, since $l_1 = t : comm(x)$, we have that a process of kind $t[P_1]$ is a sub-term of P, $t[P_1']$ is a sub-term of P' and $t[P_1] \xrightarrow{l_1} t[P_1']$, for opportune P_1 and P_1'. Secondly, since $tr(l_1) \neq tr(l_2)$ the label l_2 refers either to a transaction different from t or to a transition not involving transactions. Besides, there exists a process P_2 and a process P_2' such that $P_2 \xrightarrow{l_2} P_2'$ and P_2 is a sub-term of P' and of P. From the previous observations, it is inferred that $P \equiv t[P_1]|P_2$, $P' \equiv t[P_1']|P_2$ and $P'' \equiv t[P_1']|P_2'$. Furthermore, $P \equiv P_2|t[P_1]$ (by applying structural congruence). If the rule par is first applied considering $P_2 \xrightarrow{l_2} P_2'$, we obtain $P \xrightarrow{l_2} P_2'|t[P_1]$ and finally by considering $t[P_1] \xrightarrow{l_1} t[P_1']$, we derive the following transition sequence: $P \xrightarrow{l_1} P_2'|t[P_1] \xrightarrow{l_1} P_2'|t[P_1'] \equiv t[P_1']|P_2' \equiv P''$. The process $P_2'|t[P_1]$ is the process P''' we are looking for. □

Theorem 1 (Seralizability). *If* $P \xrightarrow{\sigma_1} P'$ *with* $actT(P) = actT(P') = \emptyset$*, then there exists a permutation* σ_2 *of* σ_1 *such that* $P \xrightarrow{\sigma_2} P'$ *is serialized.*

Proof. Let σ_1 be $l_1 l_2 ....l_n$ with $n \geq 1$. We have to show that there exists a permutation $\sigma_2 = l_1' l_2' l_3' ....l_n'$ of σ_1 such that $P \xrightarrow{\sigma_2} P'$ is serialized. The proof is by induction on the length of σ_1. Some preliminary definitions and results are given.

The *length* of σ is defined as:

- $length(\sigma) = 1$ if $\sigma = l$
- $length(\sigma) = n$ if $\sigma = l_1 l_2 ....l_n$

Besides, the following two assertions are derived from the reduction rules:

Assertion 1 If $P \xrightarrow{l} P'$ with $act_T(P) = \{t\}$ and $l = t : end$ then $act_T(P') = \emptyset$

Assertion 2 If $P \xrightarrow{l} P'$ with $act_T(P) = \emptyset$ and $tr(l) = \perp$ then $act_T(P') = \emptyset$

Given $n = length(\sigma_1)$, we consider the two cases:

1. If $n = 1$, the only possibility is that $\sigma_1 = l_1$ and $tr(l_1) = \perp$ and it is obviously serialized.

2. If $n > 1$, we have two possibilities:

   - $tr(l_i) = \perp$ for $\forall i \in [1, ..., n]$, it is obviously serialized.
   - There exist (at least two) labels in σ_1 for which $tr(l) = t$. Let be l_k (with $k < (n-1)$) the first transition label with $tr(l_k) \neq \perp$. We have that $l'_i = l_i$ for $i = 1, ..., k$. As $actT(P_0) = \emptyset$ (from *assertion 2*), it follows that $actT(P_i) = \emptyset$ for $i < k$. From this, we can deduce that $l_k = t : start$. Consider the label $l_{(k+1)}$. There are three different cases:

     (a) if $l_{(k+1)} = t : end$ then $l'_{(k+1)} = l_{(k+1)}$. In the case $(k + 1) = n$ we conclude $\sigma_2 = l_1 l_2 ... l_k l_{(k+1)}$. Otherwise, since $Act_{(k+1)} = \emptyset$ from the *assertion 1*, for the inductive assumption on $P_{(k+1)} \xrightarrow{\sigma'_1} P_n$ there exists σ'_2 for which the transition sequence is serialized. We conclude that $\sigma_2 = l_1 l_2 ... l_{(k+1)} \sigma'_2$.

     (b) If $_{(k+1)} = t : comm(x)$ then $l'_{(k+1)} = l_{(k+1)}$. Then we need to consider the following elements until $l = t : end$ is found. According to the kind of labels, at each step we proceed as in (*b*) or (*c*). When $l = t : end$ is found we apply the inductive assumption on the rest of the sequence as seen in (a).

     (c) If $tr(l_{(k+1)}) \neq t$, we consider the first s such that $tr(l_{(k+s)}) = t$ (with $s > 1$). At this point the *lemma 1* is applied $(s - 1)$ times and we finally obtain the sequence $P_{(k+1)} \xrightarrow{l_{(k+s)}} P'_{(k+2)} \xrightarrow{l_{(k+1)}} ... P_{(k+s)} \xrightarrow{\sigma'_1} P'$. The transaction labeled $l_{(k+s)}$ moves at the $(k + 1)$ position and $l'_{(k+1)} = l_{(k+s)}$. If $l_{(k+s)} = t : end$ then $Act(P'_{(k+s)}) = \emptyset$ (from *assertion 1*) and we may applied the inductive assumption on $P_{(k+s)} \xrightarrow{\sigma'_1} P'$ as seen for (a). Otherwise, we consider the next element and proceed as seen until $t : end$ is found.

□

Corollary 1. *If $P \xrightarrow{\sigma_1} P'$ and $actT(P) = \{t\}$ then there exists σ_2 and σ_3, with $\sigma_2\sigma_3$ a permutation of σ_1, such that for each $l \in \sigma_2$ $tr(l) = \{t\}$ and $P \xrightarrow{\sigma_2} P'' \xrightarrow{\sigma_3} P'$ where $actT(P'') = \emptyset$.*

Proof. It follows directly from the application of Lemma 3.3 to move all the transitions involving t to the first positions. We can say that σ_2 is given by the sequence of labels so found ($tr(l) = t$ for each of them for construction) and σ_3 is given by the rest of the labels. Besides, $\sigma_2\sigma_3$ is clearly a permutation of σ_1. □

Given a sequence of transitions involving one or more transactions, it is so possible to obtain a permutated sequence that is serialized, i.e. in which the transactions may be executed one after the other. As for the corollary, it says that if in a sequence of transitions some refer to a transaction t, it is possible to permutate the transitions such that all the actions involving t are not interleaved by other actions.

Concerning atomicity, we first need to define *simple biological systems*, for which the property is guaranteed. These are systems where it is not possible to have processes that could execute infinitely in parallel with a transaction. If more general systems were considered, it could happen that a transaction starts but does not commit in a finite number of steps as its transitions are interleaved by transitions involving external processes. For instance, if the transitions $S_1 \xrightarrow{l_1} S_2$ and $S_2 \xrightarrow{l_2} S_1$ are considered, then it could happen that the following sequence is obtained: $P'|S' \xrightarrow{l_1} P'|S_2 \xrightarrow{l_2} P'|S_1 \xrightarrow{l_1} P'|S_2...$ and so on infinitely. The transaction has started, but does not complete, at least in a finite time. We consider the following theorem:

Theorem 2 (Atomicity). *Let be $P = start(t)[P_0].P_1|P_0'$, with P_0' structural congruent to P_0 and $P_0'|P_1\{^{t'}/_t\}$ well defined. If $P \xrightarrow{l} P'$ with $l = (t' : start)$ and $P' = t'[P_0'|P_1\{^{t'}/_t\}]$, then always $P'|S \xrightarrow{\sigma} P''|S'$ for any(finite length) sequence σ, where $P'|S$ is a simple biological system and P'' indicates the final process of the transaction.*

Proof. Given the process $P' = t'[P_0'|P_1\{^{t'}/_t\}]$ as the internal process is well-defined from the hypothesis, there exists a sequence of n transitions $\sigma_0 = l_1^0 l_2^0 ... l_n^0$ such that $P' \xrightarrow{\sigma_0} P''$ where P'' is the final result of the transaction. As the system S' is a simple biological system, it always reduce to a final process in a finite number of steps and we indicate with σ_1 any possible finite length label sequence. Consider $P'|S$. All possible reductions are obtained from the possible transitions of the two sub-processes and by applying the rule par. Consequently, σ is defined by considering the labels in σ_0 and σ_1 according to the possible order of action selection. From this observation, we conclude that the sequence σ has finite length. □

If we wanted to extend the theorem to more general models, it would be necessary to add a new action to abort the transactions after some time and give an opportune compensation to rollback to the initial situation.

4 Application of transactions to model reactions

In this section, some explicative examples about the application of transactions to model biological reactions are presented. These examples are about:

- *three-reactant reactions*
- *two-modifier one-reactant reactions*
- *one reaction from the citric acid cycle*

Finally, some observations about the introduction of the non-deterministic choice are reported.

4.1 Examples about reactions

4.1.1 Three-reactant reaction $R_1 + R_2 + R_3 \rightarrow P$

The reaction $R_1 + R_2 + R_3 \rightarrow P$ represents a generic reaction with three reactants and one product. In section 2.1, a possible way to translate the reaction using the standard π-calculus has been shown and discussed. Here we describe how to model it by using biological transactions. The reaction is represented by using transactions as:

$$\begin{aligned} T_{123} &= start(t)[P_2|P_3].P_1|P_2|P_3 \\ P_1 &= \overline{x_{12}}\langle\rangle.\,\overline{x_{123}}\langle\rangle.\,end(t).P_{Pr} \\ P_2 &= x_{12}().\ \mathsf{nil} \quad P_3 = x_{123}().\ \mathsf{nil} \end{aligned}$$

where P_{Pr} represents the process describing the product P. The system $S = S' \mid T_{123}$ (where S' indicate a process different from P) may be reduced as:

$$\begin{aligned} T_{123} \mid S' &\longrightarrow t'[\overline{x_{12}}\langle\rangle.\,\overline{x_{123}}\langle\rangle.\,end(t').P_{Pr} \mid x_{12}().\ \mathsf{nil} \mid x_{123}().\ \mathsf{nil}] \mid S' \\ &\longrightarrow t'[\overline{x_{123}}\langle\rangle.\,end(t').P_{Pr} \mid \mathsf{nil} \mid x_{123}().\ \mathsf{nil}] \mid S' \\ &\longrightarrow t'[end(t').P_{Pr} \mid \mathsf{nil} \mid \mathsf{nil}] \mid S' \\ &\longrightarrow P_{Pr} \mid \mathsf{nil} \mid \mathsf{nil} \mid S' \equiv P_{Pr} \mid S' \end{aligned}$$

The term with the start of the transaction describes initially the activity of only one reactant (R_1) and the actions of the other two reactants R_2 and R_3 are described by (initially) external processes. Only after the transaction t' has started, all the processes involved in it move inside t'. This permits the composition of the model from the processes describing each species, as in π-calculus. Finally, the approach proposed is easily extendible to reactions with more reactants and products.

4.1.2 Reaction of kind $R + E_1 + E_2 \rightarrow P + E_1 + E_2$

This reaction represents an enzymatic reaction with one reactant R and two enzymes E_1 and E_2. This kind of reaction is generally used as an abstraction of a sequence of elementary reactions. Transactions are useful when we do not know how to decompose the reaction in meaningful elementary steps and as a consequence would be desirable to represent the whole reaction as an atomic one. Following the approach proposed previously, the reaction may be modeled as:

$$T_{RE_1E_2} = start(t)[P_2'|P_3'].P_1 \mid P_2' \mid P_3'$$

where P_1 is defined as in the previous example, $P_2' = x_{12}().P_2'$ and $P_3' = x_{123}().P_3'$.

A possible reduction of the system $S = S'|T_{RE_1E_2}$ is given by:

$$\begin{array}{rcl} T_{RE_1E_2} \mid S' & \longrightarrow & t'[\overline{x_{12}}\langle\rangle.\,\overline{x_{123}}\langle\rangle.\,end(t').P_{Pr} \mid x_{12}().\,P_2' \mid x_{123}().\,P_3'] \mid S' \\ & \longrightarrow & t'[\overline{x_{123}}\langle\rangle.\,end(t').P_{Pr} \mid P_2' \mid x_{123}().\,P_3'] \mid S' \\ & \longrightarrow & t'[end(t').P_{Pr} \mid P_2' \mid P_3'] \mid S' \\ & \longrightarrow & P_{Pr} \mid P_2' \mid P_3' \mid S' \end{array}$$

4.1.3 One example from citric acid cycle

The citric acid cycle (also known as the Krebs cycle) is part of a metabolic pathway involved in the chemical conversion of carbohydrates, fats and proteins into carbon dioxide and water to generate a form of usable energy. The model (taken from the KEGG metabolic pathway database, [5]) consists of a series of chemical reactions of central importance in all living cells that involves a lot of proteins, molecules and enzymes. Here we focus only on the first reaction of the cycle, where the *Acetyl-CoA* (the activated acetic acid) reacts with *oxaloacetate* (the four carbon carboxylic acid) to form *citrate* and *CoA*. The reaction is catalyzed by the enzyme *citrate-synthase* (CS). The reaction as the following form:

$$Acetyl\_CoA + Oxoloacitate + CS \rightarrow CoA + Citrate + CS$$

The reaction is represented by using transactions as:

$$\begin{array}{rcl} T_{first} & = & start(t)[Ox|CitSynt](AcCoA)|Ox|CitSynt \\ AcCoA & = & \overline{x_{12}}\langle\rangle.\,\overline{x_{123}}\langle\rangle.\,end(t).(CoA|Citrate) \\ Ox & = & x_{12}().\,\mathsf{nil} \\ CitSynt & = & x_{123}().\,CitSynt \end{array}$$

where *CoA* and *Citrate* represent the processes describing the two products. Supposing that the initial system is of the form $S = T_{first}|S'$, one of the possible reduction is:

$$\begin{aligned} T_{first} \mid S' &\longrightarrow t'[\overline{x_{12}}\langle\rangle.\,\overline{x_{123}}\langle\rangle.\,end(t').(CoA|Citrate) \mid x_{12}().\,\mathsf{nil} \mid \\ &\quad x_{123}().\,CitSynt] \mid S' \\ &\longrightarrow t'[\overline{x_{123}}\langle\rangle.\,end(t').(CoA|Citrate) \mid \mathsf{nil} \mid x_{123}().\,CitSynt] \mid S' \\ &\longrightarrow t'[end(t').(CoA|Citrate) \mid \mathsf{nil} \mid CitSynt] \mid S' \\ &\longrightarrow (CoA|Citrate) \mid \mathsf{nil} \mid CitSynt \mid S' \end{aligned}$$

where $(CoA|Citrate) \mid \mathsf{nil} \mid CitSynt \mid S' \equiv CoA|Citrate|CitSynt \mid S'$.

4.2 Non-deterministic choice

In describing biological models, it often happens that a biological entity (as a gene or a protein) is involved in more than one reaction and only one of the possible reaction may be selected. This situation is modeled in π-calculus by using the operator choice $+$. The process $P + Q$ (to be added to the process syntax definition) describes a process that behaves either as P or as Q. For instance, if the system is composed of three proteins A, B, C, involved in the reactions $r_1 : A + B \to C$ and $r_2 : A + C \to C$, the protein A may be consumed in both the reactions. The system may be described by $S = P_A \mid P_B \mid P_C$, with $P_A = \overline{x_{AB}}\langle\rangle.P_C + \overline{x_{AC}}\langle\rangle.\,\mathsf{nil}$, $P_B = x_{AB}().\,\mathsf{nil}$ and $P_C = x_{AC}().P_C$.

If a new reaction $r_3 : \quad A + E + F \to D$ is added, a transaction may be defined to model it, as seen in 4. We indicate with T_{AEF} the process to describe the transaction. The specification of the system composed of only one reactant for type and described by the three reactions r_1, r_2 and r_3 is:

$$S = (T_{AEF} + P_A)|P_B|P_C|P_E|P_F$$

where $T_{AEF} = (start(t)[P_E|P_F].\,P'_A)$ with $P'_A = \overline{x_{AE}}\langle\rangle.\,\overline{x_{AEF}}\langle\rangle.\,end(t).\,P_D$, $P_E = x_{AE}().\,\mathsf{nil}$, $P_F = x_{AEF}().\,\mathsf{nil}$. P_B and P_C are the ones defined previously.

Respect to the definition of the calculus presented in 3, we must add the process $P + Q$ to the syntax definition and modify the rule comm (in the usual way), the axiom tstart and the axiom tend. The new rules are:

(tstartChoice) $\quad (P_1 + Q_1)|...|(P_n + Q_n)|S|start(t)[P'_1|...|P'_n].P_0 + Q_0$
$\longrightarrow S \mid t'[P_0\{^{t'}/_t\}|P_1|...|P_n]$
where t' fresh and $P_i \equiv P'_i$ and for $i = 1, .., n$

(tEndChoice) $\quad t[end(t).P_1 + Q_1|P_2] \longrightarrow P_1|P_2$

where Q_1, ..., Q_n and Q_0 are the processes not used for the transaction.

5 Discussion and conclusions

This work presents an extension of the synchronous π-calculus with transactions to model multiple-reactant multiple-product reactions. These reactions, rare in nature, are quite frequent in biological models. Some problems arise when we model them by using a sequence of pair-wise communication in the standard π-calculus, as it cannot be represented atomically. A possibility to override this problem is to consider transactions, since they allow to represent a sequence of elementary actions as an atomic one. The transactions considered here have simple features suitable for our purposes. They are called *biological transactions* to distinguish them from the more complex ones used in web-service and database fields. An extended calculus has been defined and then some examples about its application have been shown and discussed.

In this work, we focus only on qualitative models. In the future we aim to define a stochastic version of this extended calculus. This would make possible to model complex reactions as atomic actions also in the quantitative case. The approach to follow is the one proposed in [11] for the standard stochastic π-calculus. The idea is to enrich each prefix with a rate, that represents the parameter of an exponential distribution that characterizes the stochastic behavior of the activity corresponding to the associate prefix. In addition, the semantics should be modified to consider the quantitative information. As in [11], the reference algorithm is the one proposed in Gillespie [4], but a generalized version should be considered, as in transactions more than two processes are involved. Concerning the rule par, we should refer to the possible subcases, that correspond to the possible actions. In addition to the functions to count the number of input and output processes on the channel x, we would need to define a function to count the number of processes congruent to a given one. A critical point is how to count the input and output on a channel x when a communication is enabled inside a transaction. A possible idea is to consider all the inputs and the outputs enabled in the system and not only the ones inside the transaction. All these ideas will be better developed and formalized in a future work. Another possible extension of this paper is to introduce opportune abort actions and compensation mechanisms. This would permit to extend the atomicity property to more general systems.

References

[1] L. Bocchi, C. Laneve, and G. Zavattaro. A calculus for long-running transactions. In *Proc. of FMOODS'03*, volume 2884 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[2] N. Busi and G. Zavattaro. On the serializability of transactions in javaspaces. In *Proc. of CONCOORD'01*, volume 54 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[3] M. Butler, T. Hoare, and C. Ferreira. A trace semantics for long-running transactions. In *Proc. of 25 year of CSP*, volume 3525 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.

[4] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[5] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.*, (28):27–30, 2000.

[6] C. Kuttler, J Niehren, and R. Blossey. Gene regulation in the π-calculus: simulating cooperativity at the lambda switch. In *Second international workshop on concurrent models in molecular biology (BioConcur 2004)*, 2004.

[7] C. Laneve and G. Zavattaro. Foundations of web transactions. In *Proc. of FOSSACS 2005*, volume 3441 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.

[8] P. Lecca, C. Priami, P. Quaglia, B. Rossi, C. Laudanna, and G. Costantin. Language modeling and simulation of autoreactive lymphocytes recruitment in inflamed brain vessels. *SIMULATION: Transactions of The Society for Modeling and Simulation International.*, 80, 2004.

[9] R. Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, 1999.

[10] C. Priami and P. Quaglia. Beta binders for biological interactions. In *Computational Methods in Systems Biology '04 (CMSB04)*, 2004.

[11] C. Priami, A. Regev, W. Silverman, and E. Shapiro. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80(1):25–31, 2001.

[12] A. Regev. Representation and simulation of molecular pathways in the stochastic π-calculus. In *Proceedings of the 2nd workshop on Computation of Biochemical Pathways and Genetic Networks*, 2001.

[13] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. Bioambients: An abstraction for biological compartments. *Theoretical Computer Science*, 325(1):141–167, September 2004.

[14] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In *Proceedings of the Pacific Symposium of Biocomputing 2001*, volume 6, pages 459–470, 2001.

[15] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[16] V.Danos and J.Krivine. Formal molecular biology done in CCS-R. In *BioConcur '03, Workshop on Concurrent Models in MolecularBiology*, 2003.