Software Verification of Biomolecular Systems

Gabriel Ciobanu

Romanian Academy, Institute of Computer Science, Iaşi
gabriel@iit.tuiasi.ro

Summary. This chapter describes the kinetics of the sodium–potassium exchange pump in terms of the π-calculus process algebra. The π-calculus has the advantage of a software verification tool. We emphasize that this software tool is able to check various properties and to provide confidence in the formal description of the pump. This model checker is used to verify that our model of the pump is deadlock free. It is also used to prove that a detailed description with a large number of states has the same behaviour with a model with a smaller number of states. This simpler model can become a part of a larger system, and in this way we get a scalable and compositional abstraction for biomolecular systems.

Keywords: process algebra, π-calculus, model checking, temporal logic.

1 Introduction

One of the goals of the new field of "computational methods in system biology" is to model and simulate various biological and biochemical networks (metabolic networks, molecular networks, gene networks) that are so complex that they require a formal framework for an accurate representation. Recent work by Cardelli, Harel, Pnueli, Regev, Shapiro and others suggests that process algebras, in particular the π-calculus and mobile ambients, may become valuable tools in modelling and simulation of the biological systems where the interaction and mobility are important features. The field may have an important impact in understanding how the biological systems work, giving at the same time a way to describe, manipulate, and analyze them.

A real challenge for computer science is to understand what paradigm of computation the cell is using, and what are the appropriate tools to study it. An essential step is to find a good and appropriate abstraction. Such an appropriate abstraction for biological systems should be able to highlight the main properties of a system, ignoring unimportant aspects. The model should be relevant and understandable, providing a conceptual framework to express

desirable features of a system, and then to prove some properties. The molecular biology community is looking for a unifying abstraction for describing the dynamics of molecular systems and able to describe faithfully the interaction of various components, qualitative and quantitative reasoning, as well as similar behaviours of two related systems, providing a scalable approach to systems of higher levels.

It is reasonable to expect that, in order to model biological systems, we can adapt and apply the range of tools developed in concurrency theory from the formalisms of process algebras to the accompanying verification techniques for temporal logics. This is what we do in this chapter. We present a system of interacting entities as a system of computational interacting entities, and then use a process algebra to describe, simulate, and, more important, verify automatically various properties of molecular systems. We insist on the model checking aspects related to this representation of biomolecular systems.

The biomolecular system used in this chapter is the sodium–potassium exchange pump. The main function of the pump is to move sodium and potassium ions across a cell membrane, namely Na ions from inside to outside, and K ions from outside to inside. We use the π-calculus [10, 14] to describe the changing configurations and movements of the pump. We describe the molecular components as computational processes of the π-calculus, individual domains as communication channels, and molecular interaction as channel transmission according to the π-calculus rules. This abstraction helps us to reason about complex biological systems. The π-calculus has a software verification tool, and we emphasize that this software tool is able to check properties and to provide confidence in the formal description of the pump. The properties are checked upon their models, and their verification is based on the exhaustive search of the state space generated by the π-calculus model. The verification tool is able to prove that the behaviour of the π-calculus processes are faithful to the biological components.

2 Sodium–Potassium Exchange Pump

Cell membranes are crucial to the life of the cell. Defining the boundary of the living cells, membranes have various functions and participate in many essential cell activities including barrier functions, transmembrane signalling, and intercellular recognition. A substantial fraction of the energy available for life processes is provided across biological membranes, and it depends on the corresponding gradients. We refer to the sodium–potassium exchange pump which is a membrane-bound protein that establishes and maintains the high internal K^+ and low internal Na^+ concentrations in cells. It is an important physiologic process present in all animal cells. By using the energy from the hydrolysis of one molecule of ATP, the pump transports three Na^+ outside the cell, in exchange for two K^+ that are taken inside the cell. This exchange is critical in maintaining the osmotic balance of the cell, the resting membrane

potential of most tissues, and the excitable properties of muscle and nerve cells.

This molecular process concerns phenomena related to distribution, cooperation, but with mobility and adaptability as well. We describe the molecular interactions and conformational transformations of the sodium–potassium exchange pump in an explicit way. We manipulate formally the changing conformations and describe the corresponding dynamic systems using discrete mathematics instead of the usual partial differential equations. The transfer mechanisms are described in more detail, step by step.

The sodium–potassium pump is a primary active transport system driven by a cell membrane ATPase carrying sodium ions out and potassium ions in (Fig. 1). An animated representation of the pump is available on the web at http://arbl.cvmbs.colostate.edu/hbooks/molecules/sodium_pump.html.

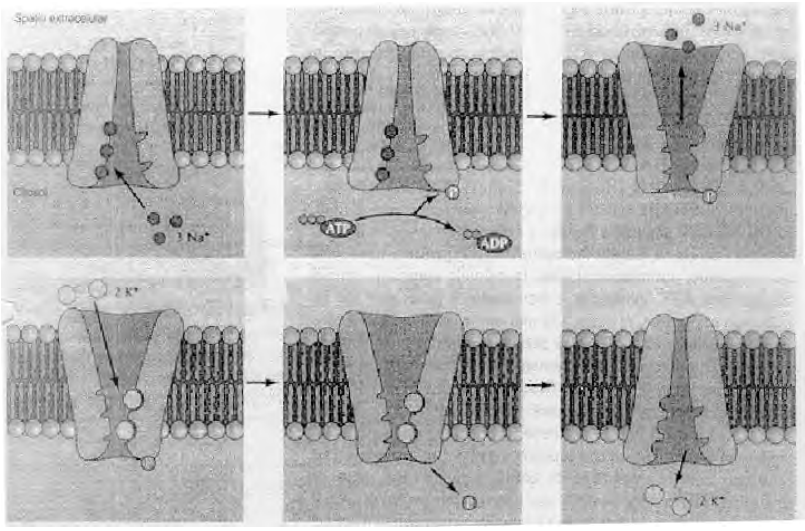

Fig. 1. The sodium–potassium exchange pump

The description given in Table 1 is known as the Albers–Post model. According to the Albers–Post cycle, the Na–K pump has essentially two conformations E_1 and E_2, which both may be phosphorylated or dephosphorylated. Ion transport is mediated by transitions between these conformations. In the table, $A + B$ means that A and B are present together (e.g., in a test tube). $A \cdot B$ means that A and B are bound to each other noncovalently. $E_2 \sim P$ indicates that the phosphoryl group is covalently bound to E_2. P_i is the inorganic phosphate group. $\rightleftharpoons$ indicates that the process can go either way, i.e., can proceed reversibly.

E_1 binds Na^+ to a high-affinity site available only from the inside (1). The binding of the sodium stimulates the enzyme to hydrolyze ATP (2), forming

Table 1. The Albers–Post model

$$E_1 + Na^+_{in} \rightleftharpoons Na^+ \cdot E_1 \quad (1)$$
$$Na^+ \cdot E_1 + ATP \rightleftharpoons Na^+ \cdot E_1 \sim P + ADP \quad (2)$$
$$Na^+ \cdot E_1 \sim P \rightleftharpoons Na^+ \cdot E_2 \sim P \quad (3)$$
$$Na^+ \cdot E_2 \sim P \rightleftharpoons E_2 \sim P + Na^+_{out} \quad (4)$$
$$E_2 \sim P + K^+_{out} \rightleftharpoons K^+ \cdot E_2 \sim P \quad (5)$$
$$K^+ \cdot E_2 \sim P \rightleftharpoons K^+ \cdot E_2 + P_i \quad (6)$$
$$K^+ \cdot E_2 \rightleftharpoons K^+ \cdot E_1 \quad (7)$$
$$K^+ \cdot E_1 \rightleftharpoons K^+_{in} + E_1 \quad (8)$$

a phosphorylated enzyme intermediate. Then conformation E_1 changes to E_2 (3): Na^+ is exposed to the outside surface and Na^+ binding is of a low-affinity type. Na^+ is then released to the outside (4). On the outside surface is a potassium binding site exposed by the E_2 phosphorylated enzyme. When K^+ binds (5), the phosphoenzyme P is hydrolyzed (6). This stimulates the enzyme to expose the potassium binding site to the inside surface of the membrane, changing its conformation from E_2 to E_1 (7). K^+ binding becomes of low affinity and we have the release of the potassium ions to the inside (8). The ATPase is now ready to bind Na^+ once more. Inside and outside in this mechanism refer to the inside and the outside of the cell plasma membrane in which the Na^+/K^+–ATPase resides.

Regarding the relationship between the kinetic parameters of the transport process and the efficiency of the pump, we can mention that the rate constants of competing steps (that would decrease the efficiency) are small. This ensures that the binding and the release of substrate occur at the proper point in the cycle. For example, the reaction $E_1 + ATP \rightleftharpoons E_1 \sim P + ADP$ of (2) is slower than the reaction described by (1). As a consequence, E_1 has enough time to bind sodium ions before undergoing the transition to E_2. Similar relationships among rate constants ensure that ions are released from the enzyme before they come back to the side at which they were initially bound. In other words, the slow rate constants channel the enzyme along a reaction path in which the hydrolysis of ATP is tightly coupled to the transport process.

3 The π-calculus

In computer science there exist many formalisms to describe mobile, concurrent, and distributed systems. Among them, the π-calculus is a widely accepted theory of interacting systems with dynamically evolving communication topology [9, 10, 14]. The π-calculus is a general model of computa-

tion which takes interaction as a primitive. It has a simple semantics and a tractable algebraic theory. The π-calculus computation is given by interaction matchings and their appropriate reduction rules. Both the sender and receiver offer their availability for interaction. Similar mechanisms work in computation and in biology [2, 4].

The π-calculus was introduced by Milner, Parrow, and Walker [11] as an attempt to describe concurrent communicating processes. It models networks in which messages are sent from one site to another site, possibly containing links to active processes or to other sites, and allowing dynamic reconfiguration among processes. It is able to describe complex systems, providing a conceptual framework and mathematical tools. The computational world of the π-calculus contains processes (also called agents) and channels (also called names or ports). The processes are denoted by $P, Q, \ldots$, and the channels by $x, y, \ldots$. There are two types of prefixes (or guards): the input prefix $x(y)$ to receive a name for y along the channel x, and the output prefix $\overline{x}\langle z\rangle$ to send the name z along the channel x. Interaction is established by a nondeterministic matching which dynamically binds "senders" to eligible "receivers". Even though there are many pairs which can satisfy the matching condition, only a single receiver gets the commitment of the sender. Thus processes can interact by using names they share. A name received in one interaction can be used in another; by receiving a name, a process can interact with processes which are unknown to it, but now they share the same channel name. The π-calculus mobility comes from its scoping of names and extrusion of names from their scopes.

Starting with atomic actions and simpler processes, complex processes can be constructed in many ways. The process expressions are defined by guarded processes, parallel composition $P|Q$, nondeterministic choice $P + Q$, replication $!P$, and a restriction operator $(\nu x)P$ creating a local fresh channel x for a process P. The evolution of a process is described in the π-calculus by a reduction relation over processes, also called reaction. This reaction relation contains those transitions that can be inferred from a set of rules.

Without loss of generality, we present in this section the monadic version of the π-calculus: this means that a message between two processes consists of exactly one name. The polyadic version allows messages of 0 or more names, and it can be expressed in terms of the monadic version [10]; as a consequence, we may use the polyadic version as well in our description of the biomolecular systems.

Let $\mathcal{X} \subset N$ be an infinite countable set of *names*. The elements of $\mathcal{X}$ are denoted by $x, y, z \ldots$. The terms of this formalism are called processes and processes are denoted by $P, Q, R \ldots$.

Definition 1. *The* **processes** *over the set $\mathcal{X}$ of names and using the prefixes $\pi ::= \overline{x}\langle z\rangle \mid x(y) \mid \tau \mid [x = y]\pi$ are defined by*

$$P ::= 0 \mid \pi.P \mid P + Q \mid P \mid Q \mid !P \mid (\nu x)P$$

Processes evolve by performing interactions, and these interactions are given by their prefixes π. The output prefix $\overline{x}\langle z\rangle$ sends z along x; an input prefix $x(y)$ waits until a name is received along x and substitutes it for the bound variable y. τ is an unobservable action; τ can be thought of as expressing an internal action of a system. The match prefix $[x = y]\pi.P$ can evolve as $\pi.P$ if x and y are the same, and do nothing otherwise; 0 is the empty (do nothing) process. $P + Q$ represents a nondeterministic choice of P or Q. $P \mid Q$ represents the parallel composition of P and Q. A replicated process $!P$ denotes a process that allows us to generate arbitrary instances of P in parallel. The replication $!P$ can be expressed by recursive equations of parametric processes as well. The informal meaning of the restriction $(\nu x)P$ is that x is a local fresh channel for P.

The parallel composition $\overline{x}\langle z\rangle.P \mid x(y).Q$ describes the interaction along the channel x. An interaction is actually defined by a "sender" $\overline{x}\langle z\rangle.P$ and a "receiver" $x(y).Q$, and it can be represented by the transition

$$\overline{x}\langle z\rangle.P \mid x(y).Q \xrightarrow{\tau} P \mid Q\{z/y\}$$

This is a synchronous interaction, where the send operation is blocking: an output action cannot be passed without the simultaneous occurrence of its corresponding input action.

The prefix $x(y)$ binds the name y, and (νx) binds the name x. The definitions of free and bound names are standard. We denote by $fn(P)$ the set of the names with free occurrences in P, by $bn(P)$ the set of bound names of P, and by $n(P)$ the names of P. The same notations are used whenever we consider the input and output actions α. For instance, if $\alpha = x(y)$ then $bn(\alpha) = \{y\}$, and if $\alpha = \overline{x}\langle z\rangle$, then $n(\alpha) = \{x, z\}$ and $bn(\alpha) = \emptyset$. We denote by $P\{v/u\}$ the result of simultaneous substitution in P of all free occurrences of the name u by the name v, using α-conversion wherever necessary to avoid name capture. We denote by $=_\alpha$ the standard α-conversion.

A structural congruence relation is defined over the set of processes; this relation provides a static semantics of some formal constructions. The structural congruence deals with the aspects related to the structure and the names of the processes.

Definition 2. *The relation $\equiv$ over processes is called structural congruence and it is defined as the smallest congruence over processes which satisfies*

- $[x = x]\pi.P \equiv \pi.P$
- $P \equiv Q$ *if* $P =_\alpha Q$
- $P + 0 \equiv P$, $P + Q \equiv Q + P$, $(P + Q) + R \equiv P + (Q + R)$,
- $P \mid 0 \equiv P$, $P \mid Q \equiv Q \mid P$, $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$,
- $!P \equiv P \mid !P$
- $\nu x 0 \equiv 0$, $\nu x \nu y P \equiv \nu y \nu x P$,
  $\nu x(P \mid Q) \equiv P \mid \nu x Q$ *if* $x \notin fn(P)$.

The evolution of a process is described in the π-calculus by a reaction relation over processes. This reaction relation contains those transitions which

can be inferred from a set of rules. We can emphasize the use of specific prefixes by labelling the corresponding reaction step (see the corresponding rule of the definition). Accordingly, the interaction rule *com* is changed. We have the same meaning for τ, namely an interaction between a "sender" $\overline{x}\langle z\rangle$ and a "receiver" $x(y)$.

Definition 3. *The reaction relation over processes is defined as the smallest relation $\rightarrow$ satisfying the following rules, where α denotes an input or output prefix.*

$$\textit{struct: } \frac{P \equiv P' \; P \xrightarrow{\alpha} Q \; Q' \equiv Q}{P' \xrightarrow{\alpha} Q'} \qquad \textit{pre: } \alpha.P \xrightarrow{\alpha} P$$

$$\textit{par: } \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \; bn(\alpha) \cap fn(Q) = \emptyset \qquad \textit{sum: } \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\textit{com: } \frac{P \xrightarrow{\bar{x}\langle z\rangle} P' \quad Q \xrightarrow{x(y)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{z/y\}} \qquad \textit{match: } \frac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'}$$

$$\textit{res: } \frac{P \xrightarrow{\alpha} P'}{(\nu x)P \xrightarrow{\alpha} (\nu x)P'} \; x \notin n(\alpha) \qquad \textit{rep: } \frac{P|!P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'}$$

The most studied forms of behavioural equivalence in process algebras are based on the notion of bisimulation. Several definitions have been given in the literature for bisimilarity; we choose the notion of *open bisimilarity*, which is finer than many other equivalences. Its strong version is a congruence and has a simple axiomatization. More importantly, open bisimulation has an efficient characterization providing a technique for deciding bisimilarity of finite-state systems. The definition of open bisimilarity is given by using the labelled transition system defined by its reaction rules. Here we use here the so-called late-style transition system.

Definition 4. *A relation $\mathcal{S}$ defined over processes is called an open simulation if for all P, Q whenever $P \mathcal{S} Q$ then for all substitutions σ the following holds:*

$$\textit{if } P\sigma \xrightarrow{\alpha} P', \textit{ there exists } Q' \textit{ such that } Q\sigma \xrightarrow{\alpha} Q' \textit{ and } P' \mathcal{S} Q'.$$

$\mathcal{S}$ is an open bisimulation if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are open simulations. Two processes P and Q are open bisimilar $P \sim Q$ if there exists an open bisimulation $\mathcal{S}$ that relates them, i.e., $P \mathcal{S} Q$.

The bisimilarity between two processes can be verified automatically by a software program called Mobility Workbench. The bisimulation used in the verification process is called *weak open bisimilarity*. It allows the basic verification technique for proving properties of the concurrent communicating systems modelled in the π-calculus[1]. Let $\Longrightarrow \stackrel{def}{=} \xrightarrow{\tau}^*$ be the transitive and

[1] Note, however, that the usual model checking techniques are still applicable where the state space of a process is finite.

reflexive closure of the $\xrightarrow{\tau}$ relation and let $\stackrel{\alpha}{\Longrightarrow} \stackrel{def}{=} \Longrightarrow$ whenever $\alpha = \tau$ and $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$ if $\alpha \neq \tau$. The *weak open bisimulation* denoted by $\approx$ is defined exactly as the open bisimulation, replacing $Q\sigma \xrightarrow{\alpha} Q'$ with $Q\sigma \stackrel{\alpha}{\Longrightarrow} Q'$.

4 Formal Description of the Sodium–Potassium Pump

We have briefly presented a computational model of the Na–K exchange pump in [5]. The equations of the Albers–Post model are translated into an appropriate operational semantics which can describe both protein interactions (conformational transformations) and membrane transportation occurring in the pump mechanism. Here we refine the model and emphasize the automated verification associated with the computational model. In this way we show how the properties of the Na–K pump can be automatically checked.

Table 2. The π-calculus description

$$Inside(side1, Na) = \overline{side1}\langle Na\rangle.side1(j).Inside(side1, Na)$$

$$Phase1(side1, side2, ATP) = side1(i).ATP.\overline{side2}\langle i\rangle.side2(j).\overline{side1}\langle j\rangle. \\ Phase1(side1, side2, ATP)$$

$$Phase2(side2, side3, P) = side2(i).\overline{side3}\langle i\rangle.side3(j).\overline{P}.\overline{side2}\langle j\rangle. \\ Phase2(side2, side3, P)$$

$$Outside(side3, K) = side3(i).\overline{side3}\langle K\rangle.Outside(side3, K)$$

$$Energy(ATP, P) = \overline{ATP}.P.Energy(ATP, P)$$

$$System(side1, side2, side3, Na, K, ATP, P) = (\nu\ side1\ side2\ side3\ ATP\ P) \\ (Inside(side1, Na) \mid Phase1(side1, side2, ATP) \mid Phase2(side2, side3, P) \mid \\ Outside(side3, K) \mid Energy(ATP, P))$$

$$PUMP(side1, side3, Na, K) = \\ (\nu\ ATP\ P\ side2)\,(System(side1, side2, side3, Na, K, ATP, P))$$

Generally speaking, the molecular components could be treated as computational processes where their individual domains correspond to communication channels. The complementary molecular domains that allow their interaction can be modelled as the ends of a channel (one end for input, and another for output). In this way, molecular interaction coincides with communication and channel transmission. The membrane transport system involves both information and energy. Consequently, we assume that the π-calculus can model the interactions of the Na–K exchange pump. These interactions are defined syntactically and they have a clear operational semantics given by

the π-calculus reaction relation. In this way it is possible to define and study rigorously the behaviour of the pump. The molecular conformational shapes are explicitly modified, and the capabilities of the interacting components are dynamically changed in this model.

Table 2 presents the computational model of the Albers–Post mechanism. Using the reduction rules of the π-calculus, we can describe the dynamics of the pump, step by step. Figure 2 can help to understand these steps. In this figure, s1 is side1, s2 is side2, and s3 is side3 of the previous π-calculus description.

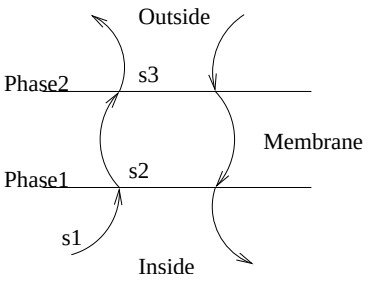

Fig. 2.

First the Na ions bind to the ATPase in conformation E_1. The Na ions are transmitted by the agent $Inside$ along channel $side1$. According to the reduction rules and the transition system of the π-calculus, the system evolution is given by the fact that

$$Inside \xrightarrow{\overline{side1}\langle Na\rangle} side1(j).Inside$$

and

$$Phase1 \xrightarrow{side1(i)} ATP.\overline{side2}\langle i\rangle.side2(j).\overline{side1}\langle j\rangle.Phase1$$

inferred with rule *pre*. Consequently,

$$Inside \,|\, Phase1 \,|\, Phase2 \,|\, Outside \,|\, Energy \xrightarrow{\tau} side1(j).Inside \,|$$

$$ATP.\overline{side2}\langle Na\rangle.side2(j).\overline{side1}\langle j\rangle.Phase1 \,|\, Phase2 \,|\, Outside \,|\, Energy$$

by rules *com* and *par* applied to the above transitions.

The pump requires some energy to proceed any further. This step corresponds to the second equation of the Albers–Post model (Table 1). We have an interaction between the complementary $\overline{ATP}$ of $Energy$ and ATP of $Phase1$. Using the rules *pre*, *com*, and *par*, the whole system evolves to

$$\xrightarrow{\tau} side1(j).Inside \,|\, \overline{side2}\langle Na\rangle.side2(j).\overline{side1}\langle j\rangle.Phase1 \,|$$

$$Phase2 \,|\, Outside \,|\, P.Energy$$

The pump suffers a conformation change. Now $\overline{side2}\langle Na\rangle$ of $Phase1$ interacts with $side2(i)$ of $Phase2$:

$$\xrightarrow{\tau} side1(j).Inside \,|\, side2(j).\overline{side1}\langle j\rangle.Phase1 \,|$$
$$\overline{side3}\langle Na\rangle.side3(j).\overline{P}.\overline{side2}\langle j\rangle.Phase2 \,|Outside \,|\, P.Energy$$

$\overline{side3}\langle Na\rangle$ of $Phase2$ interacts with $side3(i)$ of $Outside$, and the Na ions finish their movement across the membrane and are released outside:

$$\xrightarrow{\tau} side1(j).Inside \,|\, side2(j).\overline{side1}\langle j\rangle.Phase1 \,|$$
$$side3(j).\overline{P}.\overline{side2}\langle j\rangle.Phase2 \,|\, \overline{side3}\langle K\rangle.Outside \,|\, P.Energy$$

The pump transports the Na ions out of the cell in exchange for K ions entering the cell. We are now in the second phase of the process when the pump is open to the outside and it has a high affinity for the K ions. As a consequence it accepts any K ion available from the outside of the cell. $\overline{side3}\langle K\rangle$ sends K ions from $Outside$ to $Phase2$ (interacting with $side3(j)$ of $Phase2$):

$$\xrightarrow{\tau} side1(j).Inside \,|\, side2(j).\overline{side1}\langle j\rangle.Phase1 \,|\, \overline{P}.\overline{side2}\langle K\rangle.Phase2 \,|$$
$$Outside \,|\, P.Energy$$

At the previous stage of the conformational cycle, P is transferred from ATP to the carboxyl of a glutamate or aspartate residue, forming a "high energy" anhydride linkage given by P. Now the phosphate P is released by hydrolysis, and the conformation changes opening to inside. This process is reflected by the interaction between P of the $Energy$ process and $\overline{P}$ of $Phase2$:

$$\xrightarrow{\tau} side1(j).Inside \,|\, side2(j).\overline{side1}\langle j\rangle.Phase1 \,|\, \overline{side2}\langle K\rangle.Phase2 \,|$$
$$Outside \,|\, Energy$$

$\overline{side2}\langle K\rangle$ sends the K ions to $side2(j)$ of $Phase1$:

$$\xrightarrow{\tau} side1(j).Inside \,|\, \overline{side1}\langle K\rangle.Phase1 \,|\, Phase2 \,|\, Outside \,|\, Energy$$

Finally, $\overline{side1}\langle K\rangle$ of $Phase1$ interacts with $side1(j)$ of $Inside$. The K ions are inside the cell, and the pump is now ready to start a new cycle:

$$\xrightarrow{\tau} Inside \,|\, Phase1 \,|\, Phase2 \,|\, Outside \,|\, Energy$$

5 Model Checking

In this section we focus on a model checking tool for the π-calculus, emphasizing the bisimulation equivalences and properties as deadlocks. We present the Mobility Workbench code of the Na–K pump, providing a Mobility Workbench session to verify and analyze the π-calculus description.

The model checking approach to verification [7, 12] is to abstract out key elements of the software and to verify just these elements. These key abstractions are binary predicates, and various techniques and structures have been developed to automatically and efficiently check the abstract elements against specified properties. Given the underlying reliance on binary abstractions, it is no surprise that model checking is being used in the analysis of digital electronic circuits, but it has also proved effective in the software domain, particularly in the areas of protocol analysis, the behaviour of reactive systems, and for checking concurrent systems. We intend to apply this approach to the π-calculus model of the Na–K pump.

An extensive theory was developed for the π-calculus many years before its use as an abstraction for biomolecular systems. Based on the theory, methods and tools for formal verification have been developed, and they were used to implement a software tool able to verify the properties of complex concurrent systems described in the π-calculus. Mobility Workbench (MWB) is a model checking tool for manipulating and analysing mobile concurrent systems described in the π-calculus [13, 17, 18].

The π-calculus processes are called agents in MWB. An important functionality of the MWB is to decide the strong and weak open bisimilarity between two systems described as agents, as well as checking deadlocks and other properties expressed as μ-calculus formulas. The syntax of the μ-calculus formulas contains the truth values, conjunction, disjunction and negation, universal and existential quantifiers, temporal operators such as "eventually" and "always", the least fixpoint and the greatest fixpoint operators. More details of the description of the syntax and the semantics of the μ-calculus formulas are given in [8,17], and at http://www.it.uu.se/research/docs/fm/mobility/mwb. The propositional μ-calculus is a powerful temporal logic capable of embedding CTL* (and therefore CTL and LTL) [7]. Its formulas correspond either to properties of states which can be statically checked for each state, or to temporal properties which are described in terms of computation paths. The infinite computations are expressed by using the least and the greatest fixpoints. The least fixpoints correspond to eventuality properties, and the greatest fixpoints to global properties. However, it is not easy to express various properties of a (biomolecular) system as μ-calculus formulas. It would be very useful to have a friendly interface to specify the properties. We are working on such a software interface. Here we utilize the MWB commands that are easy to use.

Therefore the π-calculus description of the Na–K pump has the advantage of an automated verification tool. Once we have used the π-calculus to describe a biomolecular system, we may use a computer program to verify

various properties, using open bisimulation and propositional μ-calculus. In the previous section we have presented the π-calculus description of the Na–K pump, together with the dynamics of the pump according to the π-calculus reaction rules. We now present the MWB descriptions of the pump (Table 3), and then verify some properties. As a matter of notation, the ν operator is denoted by ^, and $\overline{side1}\langle Na\rangle$ is written as 'side1<Na>. Note that we use the polyadic π-calculus; any information is sent along the channels ATP and P, i.e., the pairs 'ATP, ATP and 'P, P are used for the synchronization of two processes (in our case, the processes Energy and Phase1 along ATP, and Energy and Phase2 along P).

Table 3. The MWB code of the Na–K pump

```
agent Inside(side1,Na) = 'side1<Na>.side1(j).Inside(side1,Na)
agent Phase1(side1,side2,ATP) = side1(i).ATP.'side2<i>.side2(j).'side1<j>.
          Phase1(side1,side2,ATP)
agent Phase2(side2,side3,P) = side2(i).'side3<i>.side3(j).'P.'side2<j>.
          Phase2(side2,side3,P)
agent Outside(side3,K) = side3(i).'side3<K>.Outside(side3,K)
agent Energy(ATP,P) = 'ATP.P.Energy(ATP,P)
agent System(side1,side2,side3,Na,K,ATP,P) = (^side1,side2,side3,ATP,P)
        (Inside(side1,Na) | Phase1(side1,side2,ATP) | Phase2(side2,side3,P) |
        Outside(side3,K) | Energy(ATP,P))
agent PUMP(side1,side3,Na,K) =
          (^ATP,P,side2)(System(side1,side2,side3,Na,K,ATP,P))
```

MWB may check whether the components of the system are deadlock-free (i.e., they are working well and they do not stop accidentally), or may check the bisimilarity of two systems. In the MWB session of Table 4 the reader can see that the whole system is deadlock-free. It is useful to know that the size of the whole system reduces drastically when we make the variables local (by using the ν operator) to the system, and in this way restrict the interaction with other components. In the definition of the system, the variables side1, side2, side3, and also ATP and P are local, and the size reduces to 8 states. If no variable is local, then the size is 3864. If only side1 is local, the size reduces to 220. If side1 and side2 are local, the size is 40. If side1, side2, and side3 are local, the size is 16. Finally, if side1, side2, side3, atp, and p are local, the size is 8.

The bisimulation equivalence is a very useful mechanism, able to provide a scalable approach. Bisimulation could be checked with MWB. Many other specific properties can also be verified with MWB. A user can formulate various properties or questions regarding the described system, translate them into corresponding formulas, and then verify them by using MWB.

To conclude, MWB provides a software alternative of lab experiments whenever we have a faithful description of a biomolecular systems. The π-

calculus supports both qualitative and quantitative properties. We can verify certain qualitative assertions on biomolecular systems by verifying a corresponding formula using MWB.

Table 4. An MWB session

```
 The Mobility Workbench
(MWB'99, version 4.134, built Fri Apr 11 2003)
MWB: input "biospec.mwb"
MWB: env
agent Energy(atp,p) = 'atp.p.Energy<atp,p>
agent Inside(side1,na) = 'side1<na>.side1(j).Inside<side1,na>
agent Outside(side3,k) = side3(i).'side3<k>.Outside<side3,k>
agent Phase1(side1,side2,atp) = side1(i).atp.'side2<i>.side2(j).
'side1<i>.Phase1<side1,side2,atp>
agent Phase2(side2,side3,p) = side2(i).'side3<i>.side3(j).'p.
'side2<j>.Phase2<side2,side3,p>
agent System(side1,side2,side3,na,k,atp,p) =
(^side1,side2,side3,atp,p) (Inside<side1,na>
| Phase1<side1,side2,atp> | Phase2<side2,side3,p> | Outside<side3,k>
| Energy<atp,p>)
agent PUMP(side1,side3,na,k) = (^atp,p,side2)
System <side1,side2,side3,na,k,atp,p>
agent TempPhase1(side1,side2,atp,na) = (^side1)(Inside<side1,na> |
Phase1<side1,side2,atp>)
agent TempPhase2(side1,side2,atp,na,side3,p) = (^side2)
(TempPhase1<side1,side2,atp,na> | Phase2<side2,side3,p>)
agent TempPhase3(side1,side2,atp,na,side3,p,k) = (^side3)
(TempPhase2<side1,side2,atp,na,side3,p> | Outside<side3,k>)
agent TempPhase4(side1,side2,atp,na,side3,p,k) = (^atp,p)
(TempPhase3<side1,side2,atp,na,side3,p,k> | Energy<atp,p>)

MWB: weq System TempPhase4
The two agents are equal.
Bisimulation relation size = 16.
MWB: eq System TempPhase4
The two agents are equal.
Bisimulation relation size = 8.
MWB: weq System PUMP
The two agents are NOT equal.

MWB: deadlocks System
No deadlocks found.
MWB: deadlocks PUMP
No deadlocks found.
```

Moreover, we can compare two π-calculus descriptions to determine the degree of similarity of their behaviour. This allows scalability without losing transparency and useful details. We can describe components taking into consideration many details and verifying various properties. Then we use bisimulation to confirm that the detailed description has the same behaviour with a much simpler description. This simpler description is integrated as a small part of a larger system, and so on.

5.1 Methodology

We present here some elements related to the methodology of using MWB and the process of verification. Let us suppose that we have a large, complex biomolecular system. Using the π-calculus, we start by describing small components in detail. The components are then considered together, and we refer to this faithful description as *the model*. At this point we have two possibilities.

Generally speaking, a system interacts with its environment. We may characterize the system from an external observer's viewpoint, i.e., considering only the observable interaction of the system with its environment. Then we can check for weak open bisimulation between the model and this simplified description. If they are bisimilar, we may include the simpler description as a part of a larger description where the details concern the interaction between the components already described. In this way we get a very useful scalable formalism able to describe faithfully large biological systems, component by component. For instance, considering our description of the Na–K pump, the specification $PUMP$ is bisimilar to $System$, i.e., $PUMP \approx System$. This means that a pump with a never-ending source of energy behaves externally, with regard to the flux of transported ions through the membrane as an agent whose carrier behaviour is desirable, and reflects the general understanding of how the pump works.

Another possibility is to check certain properties of the system. These properties are expressed using a logic. We refer to these properties as the *specifications*, and we can check if the specifications satisfy the model. This verification is automatic, i.e., done by a computer. Moreover, when a formula does not hold, the verification tool MWB can provide an error trace or a counterexample. For instance, considering our description of the Na–K pump, we may verify that System works properly forever, and it does not deadlock. MWB has a predefined name *deadlocks* for the corresponding μ-calculus formula expressing that the transition system of the described system has successors for each of its states (a process deadlocks if its transition system has states with no successors). Whenever we want to check that a system is deadlock-free, we can run the MWB program and simply write *deadlocks System*, expecting the answer *No deadlocks found*.

6 A More Detailed Description of the Pump

In this section we provide a more detailed description of the Na–K exchange pump. The new feature is given by the existence of occluded states $E_1^P \cdot (3Na^+)_{occ}$ and $E_2 \cdot (2K^+)_{occ}$ (Fig. 3 and Fig. 4). This description refines the activity of the pump, opening new perspectives.

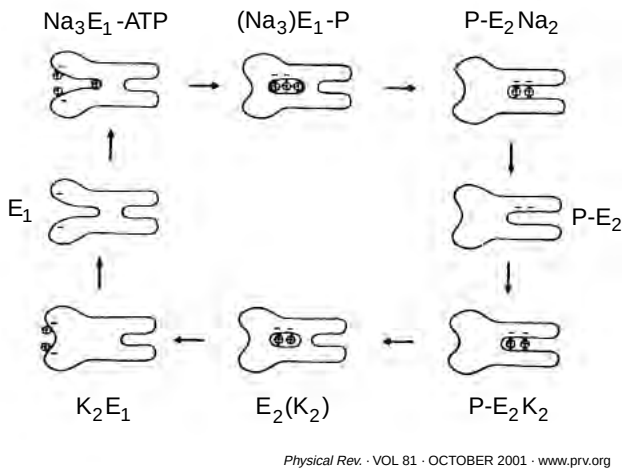

Fig. 3. The sodium–potassium pump with occluded states

We refine the Albers–Post model, adding the occluded states in which the the pump is unable to exchange ions in the surrounding media. We have the same two conformations E_1 and E_2; these conformations correspond to the mutually exclusive states in which the pump exposes ion binding sites alternatively on the cytoplasmic (E_1) and extracellular (E_2) sides of the membrane. Ion transport is mediated by transitions between these conformations.

Let us consider an initial state following the release of K ions to the cytosol (Fig. 4, left middle). The pump is in the conformation E_1, it is associated with ATP, and it has its cation binding sites empty and open to the intracellular space. In this situation, the affinity is high for Na^+ and low for K^+. Consequently, three Na^+ ions binds to the intracellular cation sites; this corresponds to the first equation of Table 5.

The binding of Na^+ is followed by the transfer of the γ phosphate of ATP to the aspartate residue of the pump structure. During this process Na ions are occluded (Fig. 4, right up). Thereafter the pump undergoes a conformational change to the E_2 state and loses its affinity for Na^+. The Na ions are subsequently released; first one Na ion is released during the conformational change from E_1 to E_2 when the cation binding sites are oriented toward the extracellular side. The pump is in the E_2^P state, and the affinity for Na ions is very

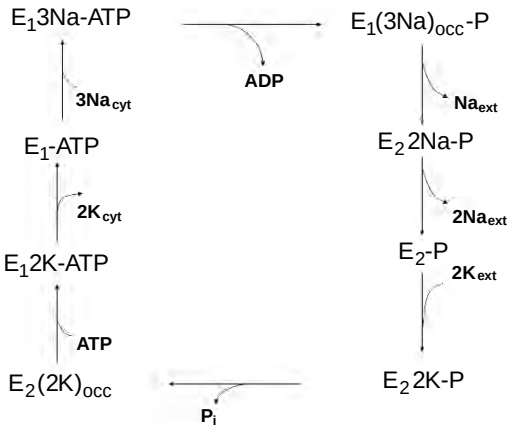

Fig. 4. The steps in the evolution of the pump

Table 5. The Albers–Post cycle with occluded states

$$E_1 \cdot ATP + 3Na^+_{cyt} \rightleftharpoons E_1 \cdot ATP \cdot 3Na^+ \quad (9)$$
$$E_1 \cdot ATP \cdot 3Na^+ \rightleftharpoons E_1^P \cdot (3Na^+)_{occ} + ADP \quad (10)$$
$$E_1^P \cdot (3Na^+)_{occ} \rightleftharpoons E_2^P \cdot 2Na^+ + Na^+_{ext} \quad (11)$$
$$E_2^P \cdot 2Na^+ \rightleftharpoons E_2^P + 2Na^+_{ext} \quad (12)$$
$$E_2^P + 2K^+_{ext} \rightleftharpoons E_2^P \cdot 2K^+ \quad (13)$$
$$E_2^P \cdot 2K^+ \rightleftharpoons E_2 \cdot (2K^+)_{occ} + P_i \quad (14)$$
$$E_2 \cdot (2K^+)_{occ} + ATP \rightleftharpoons E_1 \cdot ATP \cdot 2K^+ \quad (15)$$
$$E_1 \cdot ATP \cdot 2K^+ \rightleftharpoons E_1 \cdot ATP + 2K^+_{cyt} \quad (16)$$

low; the two remaining Na ions are released into the extracellular medium. The binding sites now have a high affinity for K^+. Two external K^+ ions can bind; this corresponds to (13) of Table 5, and to the right down corner of Fig. 4. The binding of K^+ induces the dephosphorylation of the E_2^P conformation. The release of the inorganic phosphate into the intracellular medium is accompanied by the occlusion of the K^+ ions (14). ATP is then bound and this allows a conformational change to E_1 and K ions are deoccluded. The affinity for K ions reduces and they are released into the intracellular medium (cytosol). The pump protein is now ready to initiate a new cycle.

The π-calculus description of the Albers–Post cycle with occluded states is given in Table 6. This description allows biologists to follow the behaviour of the system step by step; this is the way biologists think of the systems they

study. We can say that the models based on differential equations are not so intuitive as the π-calculus descriptions.

Table 6. The MWB code of the pump with occluded states

```
agent To1(s,i) = s<i>. To1(s,i)
agent To2(s,i,j) = To1(s,i) | To1(s,j)
agent From1(s,i) = s(i). From1(s,i)
agent From2(s,i,j) = From1(s,i) | From1(s,j)
agent Tos1(s1a,s1b,na,naplus,no) = To1(s1b,na) | To2(s1a,naplus,no)
agent Inside(s1a,s1b,na,naplus,u,v,no) = 's1b<na>.('s1a<naplus> |'s1a<no>).
      From2(s1a,u,v).Inside(s1a,s1b,na,naplus,u,v,no)
agent Phase1(s1a,s1b,s2a,s2b,atp,i,j,u,v,t1) = s1b(t1).From2(s1a,i,j).atp.'s2b<t1>.
   To2(s2a,i,j).From2(s2a,u,v).To2(s1a,u,v).Phase1(s1a,s1b,s2a,s2b,atp,i,j,u,v,t1)
agent Phase2(s2a,s2b,s3a,s3b,p,i,j,u,v,t1) = s2b(t1).From2(s2a,i,j).'s3b<t1>.
   To2(s3a,i,j).From2(s3a,u,v).'p.To2(s2a,u,v).Phase2(s2a,s2b,s3a,s3b,p,i,j,u,v,t1)
agent Outside(s3a,s3b,kplus,i,j,k) = s3b(t1).From2(s3a,i,j).To2(s3a,kplus,k).
   Outside(s3a,s3b,kplus,i,j,k)
agent Energy(atp,p) = 'atp.p.Energy(atp,p)
agent System(s1a,s1b,s2a,s2b,s3a,s3b,na,naplus,kplus,atp,p,u,v,no,k,i,j,t1) =
   (^s1a,s1b,s2a,s2b,s3a,s3b,atp,p)  (Inside(s1a,s1b,na,naplus,u,v,no)  |
   Phase1(s1a,s1b,s2a,s2b,atp,i,j,u,v,t1) | Phase2(s2a,s2b,s3a,s3b,p,i,j,u,v,t1) |
   Outside(s3a,s3b,kplus,k) | Energy(atp,p))
agent Pump(s1a,s1b,s3a,s3b,na,naplus,kplus,no,k,i,j,t1) = (^s2a,s2b,atp,p) Sys-
   tem(s1a,s1b,s2a,s2b,s3a,s3b,na,naplus,kplus,atp,p,no,k,i,j,t1)
```

Figure 5 can help to understand the π-calculus description of the Na–K pump with occluded states. It is possible to describe the dynamics of the pump according to the reaction rules of the π-calculus, as in Sect. 4. The verification of properties for the new description is done by using MWB, similarly to Sect. 5.

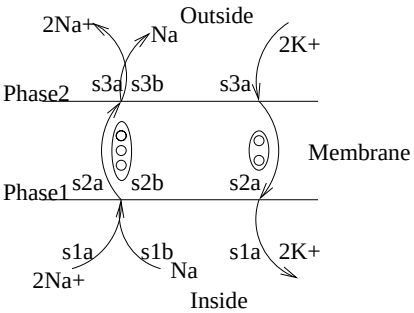

Fig. 5.

7 Conclusions

Various approaches from mathematics and computer science have been used for the description of molecular processes. The use of the π-calculus or other process algebras to model the molecular interaction is quite new. In computer science, the π-calculus is a widely accepted model of interacting systems with dynamically evolving communication topology. The π-calculus has a well-defined semantics and an appropriate algebraic theory. We think the π-calculus is a formalism capable to describe many biomolecular processes.

As far as we know, the first papers using the π-calculus to describe molecular processes were [1] and [3], followed by the successful papers [15, 16] that represent and simulate biomolecular processes, giving a set of steps taken to adapt, extend and implement a core language to describe the requirements of biochemical systems, first on a qualitative and then on a quantitative, stochastic scale. In [1], the π-calculus is used to describe DNA methylation. In [3] the so-called molecular structures are defined, and it is proved that they have the same expressive power as the π-calculus (which has the same computational power as Turing machines). A more detailed approach is presented in [4].

In this chapter we motivate the use of the π-calculus as an adequate formalism for automated verification of biomolecular systems. We describe the dynamics of the sodium–potassium exchange pump, an important physiologic process present in all animal cells. We present a computational model based on molecular interaction, which can cope with phenomena beyond the classical approach, including a new scalability providing by bisimulation. We manipulate formally the changing conformations and describe explicitly the corresponding dynamic systems using discrete mathematics instead of the (usual) partial differential equations. The transfer mechanisms are described step by step. Moreover, we can use some software tools of verification developed for the π-calculus. This means that it would be possible to verify properties of the described systems by using a computer program. The main consequence of this approach is the use of the verification software as a substitute for expensive lab experiments.

Biologists are of course interested in more detail. There are various ways to extend the approach based on the π-calculus. We have used in [5] a version of the π-calculus called stochastic π-calculus to describe the efficiency of the Na–K pump. In the stochastic π-calculus, the prefix $\pi.P$ is replaced by $(\pi, d).P$, where d is a probability distribution that characterizes the stochastic behaviour of the activity corresponding to the prefix π. This version of the π-calculus allows us to describe the complexity of the molecular interactions involving the dynamic efficiency of the pump and other quantitative aspects (e.g., kinetics rates, energy, pump failures).

Another attempt is presented in [6]. The idea is to descend from the π-calculus to a lower level of abstraction given by the communicating automata. This step allows us to add more details to the description of the system. Moreover, it is possible to get new results regarding the computational power of the

communicating automata, and to obtain a different software implementation able to model and simulate molecular networks.

Acknowledgments

Thanks to my former students Vlad Ciubotariu and Bogdan Tanasă for their collaboration. This work was partially supported by an academic research grant of the National University of Singapore.

References

1. G. Ciobanu. Formal Description of the Molecular Processes. In C. Martin-Vide, Gh. Păun (Eds.): *Recent Topics in Mathematical and Computational Linguistics*, 82-96, Ed. Academiei, Bucharest, 2000.
2. G. Ciobanu, M. Rotaru. A π-calculus machine. *Journal of Universal Computer Science*, vol.6(1), 39-59, 2000.
3. G. Ciobanu. Molecular Structures. In C. Martin-Vide, V. Mitrana (Eds.): *Words, Sequences, Languages*, 299-317, Kluwer, Dordrecht, 2001.
4. G. Ciobanu, M. Rotaru. Molecular Interaction. *Journal of Theoretical Computer Science*, vol.289(1), 801-827, 2002.
5. G. Ciobanu, V. Ciubotariu, B. Tanasă. A pi-calculus model of the Na pump, *Genome Informatics*, 469-472, Universal Academy Press, Tokyo, 2002.
6. G. Ciobanu, B. Tanasă, D. Dumitriu, D. Huzum, G. Moruz. Molecular networks as communicating membranes. In Gh. Păun, C. Zandron (Eds.), Proceedings of Workshop On Membrane Computing, MolCoNet vol.1, 163-175, 2002.
7. E. Clarke, O. Grumberg, D.Peled. *Model Checking*. MIT Press, Cambridge, MA, 1999.
8. M. Dam. Model Checking Mobile Processes. *Information and Computation*, vol.129(1), 35-51, 1996.
9. R. Milner. Elements of Interaction. Turing Award Lecture, *CACM*, vol.36(1), 78-89, 1993.
10. R. Milner. *Communicating and mobile systems : the π-calculus*. Cambridge University Press, Cambridge, 1999.
11. R. Milner, J. Parrow, D. Walker. A calculus of mobile processes. *Journal of Information and Computation*, vol.100, 1-77, 1992.
12. K.L. McMillan. *Symbolic Model Checking*, Kluwer, Dordrecht, 1993.
13. F. Orava, J. Parrow. An Algebraic Verification of a Mobile Network. *Formal Aspects of Computing*, vol.4, 497-543, 1992.
14. J. Parrow. An Introduction to the π-calculus, J.Bergstra, A.Ponse, S.Smolka (Eds.): *Handbook of Process Algebra*, Elsevier, Amsterdam, 2000.
15. A. Regev, E. Shapiro. Cellular abstractions. *Nature*, vol. 419, 343, 2002.
16. A. Regev, W. Silverman, E. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Pacific Symposium on Biocomputing*, vol.6, 459-470, World Scientific Press, Singapore, 2001.
17. B. Victor. A Verification Tool for the Polyadic π-Calculus, Licentiate Thesis, Uppsala University, 1994.
18. B. Victor, F. Moeller. The Mobility Workbench: A tool for the π-calculus, In *CAV'94: Computer-Aided Verification*, LNCS vol.818, 428-440, Springer-Verlag, Berlin Heidelberg New York, 1994.