Representing Biomolecular Processes with Computer Process Algebra: π-Calculus programs of Signal Transduction pathways

Aviv Regev[1,2], **William Silverman**[2] and **Ehud Shapiro**[2]
[1]Department of Cell Research and Immunology, Life Sciences Faculty
Tel Aviv University, Tel Aviv 69978, Israel
[2]Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel
Tel: 972-8-9344506 Fax: 972-4-8348799 e-mail: {aviv, udi, bill}@wisdom.weizmann.ac.il

Abstract

While biomolecular sequence and structure information has an objective representation that is amenable to critical evaluation and computer manipulation, knowledge of biomolecular processes has so far eluded such representation and study. Recent years have shown massive accumulation of experimental information on various biomolecular processes, including gene expression, metabolism and signal transduction (ST) pathways. However, this information is often diverse, disparate and fraught with detail, and a unifying approach to its formal representation and analysis is yet to emerge. In this paper, we propose to represent biomolecular processes using languages developed for specifying computational processes. In particular, we represent the complex RTK-MAPK signal transduction pathway in the π-calculus, a process algebra originally designed for specifying mobile communication systems. π-calculus representations of ST unify dynamic behavior and function of pathways with the molecular details that underly their behavior. They provide a comprehensive model of ST, which is both mathematically well-defined and biologically visible. π-calculus programs are amenable to computer analysis and execution, analogous to mutational manipulation and experimentation, as well as to formal comparison and verification. Adopting such a common representation language may facilitate our understanding of the function and evolution of biomolecular processes.

Key words: Signal Transduction, pathways, concurrency, π-calculus, process algebra, modeling, simulation, verification, homology of process, logic programming, FCP, Logix

Introduction

Integrated molecular systems play a key role in the normal and pathological function of cells, tissues and organisms (Gilbert 1997). These include mechanisms for the regulation of gene expression, metabolic pathways, and signal transduction (ST) networks. However, in spite of the rapid accumulation of experimental data on these systems, analysis of this data is hampered since it is diverse, disparate and fraught with molecular details (Bray 1997). The dynamic nature of these systems, their complexity, high connectivity and modularity (Heldin & Purton 1996) further complicate this task. Thus, while appropriate approaches and tools are now available for the study of biomolecular structures (e.g. sequences or 3D structures), it seems that we are lacking comparable ones for the analysis of biomolecular processes.

The need for novel approaches to study molecular processes has been recognized and discussed (Bray 1997; Wilson 1998) , and the importance of a formal language, which will allow us to represent and analyze dynamic systems and processes is evident (Smith 1998). An appropriate formal approach for molecular processes should fulfill certain goals. **First**, it should provide a unifying view of both the molecular data and the dynamic behavior it underlies. Preferably, this representation will also be biologically "visible", corresponding well to the informal concepts and ideas of molecular biology. **Second**, the formally represented data should be amenable to computer execution and analysis. Thus, the dynamic behavior of the system could be followed by simulation studies, including mutational analysis and simulated evolution. Alternatively, the system's behavior may be formally verified (Francez 1992). **Third**, the formal approach should facilitate comparative studies of a system's structure, dynamics, and function within and between species. **Finally**, the formalism should be scalable and modularized to higher levels of organization, thus providing a bridge between biochemistry and physiology, similar to the one provided by molecular systems in living organisms.

Previous attempts to formalize knowledge of expression regulation, metabolism or ST have only partially met these goals. Kinetic models, based on sets of differential equations (Barkai & Leibler 1997; McAdams & Arkin 1997; Bray, Levin, & Morton-Firth 1998; Alon *et al.* 1999) represent accurate dynamic behavior but lack the lower-level details as well as the modularity typical of a molecular system (Fontana & Buss 1996). Importantly, since higher-order characteristics are not a natural part of the model, comparative studies and scaling up are not supported by it. Logical models of boolean networks (Kauffman 1993; Szallasi & Liang 1998) provide a biologically visible functional view. However, as in the kinetic models, molecular entities are atomic and lower-level data is not incorporated.

Furthermore, the details of molecular interactions are generalized yielding a much simplified dynamic behavior. Thus, although the theory is scalable, comparative studies of real systems are greatly compromised. Object-oriented databases (Igarashi & Kaminuma 1997; Karp *et al.* 1997; Kolpakov *et al.* 1998; Salamonsen *et al.* 1999; Eilbeck *et al.* 1999) are highly successful in representing functional and molecular information and are scalable. However, such practical approaches lack the formal operational semantics necessary for modeling, studying and comparing dynamic behavior. In conclusion, these approaches represent either accurate dynamic behavior or detailed molecular information, but not both.

In this paper, we propose to unify the molecular and dynamic aspects of molecular systems, such as gene expression regulation, metabolic pathways and ST networks, within a single theoretical framework. This can be achieved by treating such systems as concurrent computational processes, and employing approaches and tools adopted from computer science (CS) studies of concurrency.

In order to show the power of such approaches, we develop a formal method for representing and analyzing ST networks, which merges molecular information with the resulting dynamic behavior. We view ST as a mobile communication system, a particular CS model of concurrent computation (Milner 1989; 1999), and treat molecules as computational processes and molecular interaction as communication. Based on this view we utilize the π-calculus, a process algebra (Milner 1999), to represent ST networks. A particular complex example, the pathway leading from a receptor tyrosine kinase, through Ras and into the MAPK cascade is represented in detail. We present designated software tools developed for the simulation of ST pathways, formally represented in the calculus, and show that such representations are amenable to various modes of simulation and formal analysis, enhancing our understanding of the function and evolution of complex molecular processes.

Concurrency, process algebras and the π-calculus

Concurrent systems are composed of a community of coexisting computational processes that can communicate with each other, in a synchronous or asynchronous fashion (Milner 1989). Various approaches have been developed for the study of concurrent systems, including Petri nets, process algebras, and modeling languages (Baeten & Weijland 1990; Milner 1999). These approaches provide graphic, algebraic or programming languages to *specify* concurrent systems, and analyze their behavior.

A recent advance in the theory of concurrent computation concerns processes that change their interconnection structure when they execute, termed *mobile* processes (Milner 1999). A mathematical theory for such mobility has been developed in the π-calculus (Milner, Parrow, & Walker 1992; Milner 1993; 1999), a concise process algebra, (Figure , a detailed description is given in (Milner 1993; 1999)). A program in this calculus specifies a community of interacting processes. Intuitively, each computational process is defined by its *potential* communication activities and may be composed in sequence or in parallel with other processes. Communication occurs via channels, denoted by their names, that represent atomic access capabilities. Computation is modeled solely as synchronous binary communication between processes over these channels. The only content of messages transmitted in communication is channel names or tuples of channel names, which may be used for further communication.

Formally, the π calculus consists of three components:

- A simple *syntax* for writing formal descriptions of a concurrent system's state (specifications)
- A set of *congruence laws* that determine when two syntactic expressions are equivalent
- An *operational semantics* consisting of *reduction rules*, which delineate the potential changes in the state of the system induced by communication.

There are two basic entities in the calculus: the *channels*, named $x, y, \cdots$, and the *processes* $P, Q, \cdots$ that communicate through them. Processes are constructed using channel names according to the syntax (Figure 1). There are two types of atomic actions, denoted as π, and called *prefix* or *guard*): The input $x(y)$, to receive a name for y along the channel named x, and the output $\overline{x}\langle z\rangle$, to send the name z along the channel named x. Complex processes can be constructed from atomic actions and simpler processes in several ways: The simplest process is the empty process, 0. A process P may be prefixed with an input or output action π, yielding a *normal process*. In addition, a match operator may precede a process. In all these cases the process P will occur only after the π action is performed or the match succeeds. Processes may also be combined to create more complex ones. The *mutual exclusive choice* operator " + " allows summation of normal processes and denotes a complex process able to perform exactly one of several alternative actions. The *choice* between the alternatives is non-deterministic and depends on the actual communication that will occur (see below). Two processes P and Q can also be *composed in parallel* by the PAR $\mid$ operator. In this case both are concurrently active. They can act independently, but can also communicate with each other. Similarly, a process P may be *replicated* by the *Bang* operator. Intuitively, $!P$ means $P \mid P \mid P \mid \cdots$, standing for an unlimited number of copies of P able to run concurrently. Finally, the use (*scope*) of a certain channel name may be *restricted* to a particular process P, by the new operator. Several additional derived constructs conveniently encode polyadic communication of a tuple of names (rather than single names) and recursive parametric definitions of processes (Milner

Syntax

$$\begin{array}{rcll} \pi & ::= & x(y) & \text{receive } y \text{ along } x \\ & & \overline{x}\langle y\rangle & \text{send } y \text{ along } x \end{array}$$

$$P \; ::= \; 0 \mid \textstyle\sum_{i\in I} \pi_i.P_i \mid [x=y]\; P \mid P_1|P_2 \mid (\mathsf{new}\ x)P \mid !P$$

Structural congruence

Renaming of bound variables

$$\begin{array}{rcll} x(y).P & = & x(z).(\{z/y\}\, P) & \text{if } z \notin FN(P) \\ (\mathsf{new}\ y).P & = & (\mathsf{new}\ z).(\{z/y\}\, P) & \text{if } z \notin FN(P) \end{array}$$

Structural congruence laws

$$\begin{array}{rclll} P|Q & \equiv & Q|P & & \text{commutativity of parallel composition} \\ (P|Q)|R & \equiv & P|(Q|R) & & \text{associativity of parallel composition} \\ P+Q & \equiv & Q+P & & \text{commutativity of summation} \\ (P+Q)+R & \equiv & P+(Q+R) & & \text{associativity of summation} \\ (\mathsf{new}\ x)0 & \equiv & 0 & & \text{restriction of inert processes} \\ (\mathsf{new}\ x)(\mathsf{new}\ y)P & \equiv & (\mathsf{new}\ y)(\mathsf{new}\ x)P & & \text{polyadic restriction} \\ ((\mathsf{new}\ x)P)|Q) & \equiv & (\mathsf{new}\ x)(P|Q) & \text{if } x \notin FN(Q) & \text{scope extrusion} \\ !P & \equiv & P|!P & & \text{replication} \end{array}$$

Reaction rules

$$(\cdots + \overline{x}\langle z\rangle.Q)|(\cdots + x(y).P) \rightarrow Q|P\,\{z/y\} \quad \text{communication (COMM)}$$

$$\frac{P \rightarrow P'}{P|Q \rightarrow P'|Q} \quad \text{reaction under parallel composition (PAR)}$$

$$\frac{P \rightarrow P'}{(\mathsf{new}\ x)P \rightarrow (\mathsf{new}\ x)P'} \quad \text{reaction under restriction (RES)}$$

$$\frac{Q \equiv P \;\; P \rightarrow P' \;\; P' \equiv Q'}{Q \rightarrow Q'} \quad \text{structural congruence (STRUCT)}$$

Figure 1: The π-calculus

1999). There are also several alternative definitions and uses of the match operator (Parrow & Sangiorgi 1995)

Since a given system may be written in different ways there are several laws that allow us to identify such *structurally congruent* expressions. These include, for example, the associativity and commutativity of the choice and PAR operators. The syntax is thus made to reflect the non-sequential nature of the processes which it represents.

The *reaction rules* define how different concurrent components communicate and the effects of this communication. In order for two concurrent processes to communicate they must share a common channel name on which the sender outputs a message (another channel name) which the receiver receives. As a result, the communication possibility is consumed, and the remaining portions of the sender and receiver are released to proceed. The receiver may now use the received channel name in further communication. This transition is formally written the COMM axiom. Note, that any other processes in the "$\cdots$" of the sum are discarded. Since several alternative communications may be possible, the actual communication that will be realized is unknown in advance and the behavior is nondeterministic. This basic transition is extended to more complex terms via derived reaction rules, that serve to rearrange the order of expression according to the structural congruence laws (STRUCT), and allow communication under parallel composition (PAR) and channel restriction (RES), but not under a prefix, a sum, or a replication (Milner 1999).

Representation of ST pathways in the π-calculus

We propose that the concurrency paradigm and the π-calculus theory are uniquely suited to model and study biomolecular processes in general (Table 1) and ST in particular (Table 2). Intuitively, the community of interacting molecules which comprises a pathway is analogous to concurrent computational processes. Hence, the structural and biochemical modifications which occur as a result of molecular interaction (transcription, synthesis, phosphorylation and activation of enzyme) correspond to the change of channel names (mobility) in the communication system. In both cases, these changes serve to enable or disable further interactions. For example, in a system (pathway) which consists of three processes (proteins), A, B, and C, communication (interaction) may occur between A and B, the message would be a change (phosphorylation) of a channel (tyrosine residue), and this would enable further communication (binding) of processes (proteins) B and C.

Principles

Within the particular framework of the π-calculus, we set three principles for this correspondence (Table 2). First, as our primitive *process*, we choose the functional signaling *domain*. This captures the functional and structural independence of domains in signaling molecules, their modularity and the composition of entities of growing complexity from simpler ones: molecules from domains, complexes from molecules.

Second, we model the component *residues* of domains as communication *channels* that construct a process. Residue stretches are the interacting portions of the domain but are not independently functional, just as channel names are the communication ports of the process, but are not processes in their own right.

Finally, molecular interaction and modification is modeled as communication and the subsequent change of channel names. The complementary molecular determinants that allow the interaction are modeled as complementary sides of a channel (e.g. x and $\overline{x}$, for input and output). Different types of molecular changes, such as chemical modification, conformation changes, or binding, all affect further interaction in an analogous manner to the change in communication capabilities following channel name passing in mobile systems. This strong correspondence, including domains as processes, residues as channels and interaction as communication, yields a model of the molecular realm which is both highly visible and detailed.

Based on these three principles the π-calculus allows us to fully represent complex molecular structures and signaling events (Table 3). For example, the integrity of single molecules and of molecular complexes, and compartments, is represented by private restricted channels, such as backbone or compartment channels. Binding and formation of such complexes, is based on communication on complementary channels, in which private channel names are passed. Sequential events, such as the change from inactive to active conformation is given as a guarded (normal) process. Reconstitutive enzymatic activity is a recursive process. Importantly, the lower-level accurate representation that we have established and the concurrent nature of the model yields the entire scope of higher-level global behavior, without additional explicit modeling of global phenomena. Since all signaling components operate in parallel, accessible to mutual communication, multiple effects can be registered on a single component, which is the basis for signals merge, cross-talk, mutual exclusion or feedback.

A π-calculus representation of the RTK-MAPK pathway

In order to show the modeling power of the π-calculus we have represented in detail substantial portions of the ST pathway which leads from a receptor tyrosine kinase (RTK) to the ERK1 MAP kinase (MAPK) cascade via Ras (Heldin & Purton 1996; Brunet & Pouysséegur 1997; Lewis, Shapiro, & Ahn 1998; Widmann *et al.* 1999). Detailed representations were given for the dimeric growth factor ligand molecule, the receptor tyrosine kinase, the MP1 scaffold, the protein kinases Raf-1, MKK1, and ERK1, and the protein phosphatase PP2A (to be available from `www.wisdom.weizmann.ac.il/~aviv`). Modeled behaviors include ligand binding, ligand-induced receptor dimerization, processive and non-processive phosphorylation and de-phosphorylation, phosphorylation-induced conformation and activity changes, scaffolding and sequestration. Feedback, competition and integration of signals is an emergent behavior that does not require explicit modeling. For example, a portion of the π-calculus code for the interacting MEK1 and ERK1 molecules is given in Figure 2. Note, that the code has been slightly simplified for clarity. Fuller code is available in the Appendix.

The $ERK1$ molecule is composed of three processes, representing its three domains (Brunet & Pouysséegur 1997): Nt_LOBE, Ct_LOBE and $CATALYTIC\_CORE$ (1). Several intramolecular connections exist between its different portions, represented by the restricted `new` channels (1). The $CATALYTIC\_CORE$ is composed of four separate determinants (2): the $ATP\_BINDING\_SITE$, the $ACTIVE\_SITE$, and the regulatory T_LOOP, which induces conformational changes upon phosphorylation and de-phosphorylation (Canagarajah *et al.* 1997). The $MEK1$ molecule (5) has a $CATALYTIC\_CORE$ and may also interact outside the core with the scaffold $MP-1$ (not shown) through its $BINDING\_SITE$ (Schaeffer *et al.* 1998). Its $CATALYTIC\_CORE$ has similar components to $ERK1$ (6). We focus on the phosphorylation or de-phosphorylation of the $ERK1$ $T-loop$. An interaction with the $MEK1$ kinase or a phosphatase (not shown) is either the result of a $DIRECT$ interaction (4), or of previous binding in a

Concurrency	Molecular Biology	Transcriptional regulation	Metabolism	ST
Concurrent computational processes	Molecules	Transcription factors (TF) and DNA sites	Enzymes and metabolites	Interacting proteins
Synchronous communication	Molecular Interaction	Binding and transcriptional activation	Binding and catalysis	
Transition or mobility	Biochemical modification or relocation	TF and RNA polymerase binding	Metabolite synthesis	Protein binding, modification or sequestration

Table 1: Molecular processes as concurrent computations

		Example	
ST	π-calculus	ST	π-calculus
Domain	Process	Several receptor molecules each with three domains	$SYSTEM ::=$ $RECEPTOR \mid RECEPTOR \mid \cdots$ $RECEPTOR ::=$ $(\mathsf{new}\ internal\ channels)(EC \mid TM \mid CYT)$
Residues	Channel names and co-names	A phosphorylation site with a phosphorylatable tyrosine residue accessible to a kinase	$PHOSPH\_SITE(tyr) ::=$ $\overline{tyr}.PHOSPH\_SITE\ +$ $kinase(tyr).PHOSPH\_SITE$
Molecular interaction and modification	Communication and change of channel names	Phosphorylation of a tyrosine residue by an active kinase	$\cdots +\ kinase(tyr).PHOSPH\_SITE \mid$ $\overline{kinase}\langle p-tyr\rangle.KINASE\_ACTIVE\_SITE$ $\longrightarrow$ $PHOSPH\_SITE\{p-tyr/tyr\} \mid$ $KINASE\_ACTIVE\_SITE$

Table 2: Principles for representing ST in the π-calculus

separate site ($INDIRECT$, not shown). For simplicity, only $DIRECT\ NON-PROCESSIVE$ modification is shown (4 and 9). Interaction depends on the presence of an appropriate residue (tyr or thr for kinase; p-tyr or p-thr for phosphatase). In this interaction, a closed scope is determined by reciprocating private channels ($cross\_enzyme$, corresponding to the binding event) and the residue channel is then changed (corresponding to the enzymatic event). The conformation, dependent on the state of the residues (phosphorylated or not) is propagated internally on the restricted *backbone* channel (4). The $MEK-1\ ACTIVE\_SITE$ (7) is regulated by the its LIP (not shown). A conformation change, propagated through the molecular backbone (as in ERK1 above) precedes the enzyme active conformation. The active enzyme (8) may bind the target residue (only direct binding is shown) and modify it (described above) followed by reconstitution of the enzyme's activity (modeled as recursion).

Implementation and simulated mutational analysis

π-calculus representations, which formally describe the disparate knowledge on ST, can be treated as computer programs, amenable to execution and analysis. There are previous implementations of the π-calculus and similar models of concurrency. The closest to the π-calculus are Pict, which is directly based on the calculus (Pierce & Turner 1998) and the Join-Calculus language, based on the related reflexive chemical abstract machine (Fournet & Gonthier 1996). Unfortunately, both proved inadequate for our needs since they deal only with the asynchronous fragment of the π-calculus (Honda & Tokoro 1991; Boudol 1992). This fragment does not allow output guards, and by this circumvents the need for "handshaking" during communication. This renders both languages inherently inadequate for simulation of molecular interaction, which is synchronous in nature. Note, that the implementation of synchronous communication is strictly more difficult than that of asynchronous (Palamidessi 1997).

We have therefore developed an alternative implementation of the full, synchronous, π-calculus,

ST	Modeling analogy	π-calculus
Molecular integrity	Intramolecular interaction via internal channels accessible only within the scope of the molecule	The new operator $MOLECULE ::=$ new $\overrightarrow{channels}\ (DOMAIN_1 \mid DOMAIN_2 \mid \cdots)$
Binding and complex formation	Two-way complementarity of global channel names and two-way passing of private channel names	The mixed choice[1] operator (+) $DIMERIZATION\_DOMAIN ::=$ $BINDING\_MOTIF(dimer_1) +$ $BINDING\_MOTIF(dimer_2)$ $BINDING\_MOTIF(dimer) ::=$ $\overline{dimer}\langle private\ \ \overrightarrow{channels\_send}\rangle.$ $private(cross\_private\ \ \overrightarrow{channels\_get}).$ $\overline{backbone}\langle cross\_dimer\ \ \overrightarrow{channels\_get}\rangle.P$
Ordered regulation	One communication event strictly preceding another	The prefix operator (.) $INACTIVE\_ACTIVE\_SITE ::=$ $backbone(residue_1, residue_2).$ $[(residue1, residue2) = (p - ser, p - ser)].$ $ACTIVE\_SITE$
Enzymatic activity	A recurring communication sequence: binding, modification and reconstitution	A guarded recursive definition $ACTIVE\_SITE ::=$ $\overline{thr}\langle kinase\rangle.atp\_binding\_site.\overline{kinase}\langle p - thr\rangle.$ $ACTIVE\_SITE$

Table 3: π-calculus modeling of typical molecular structures involved in ST and key signaling events

based on an existing platform of Flat Concurrent Prolog (FCP), the Logix system (Silverman *et al.* 1987; Shapiro 1987), to be available from `www.wisdom.weizmann.ac.il/~aviv`. Unlike most concurrent languages, FCP supports synchronous interaction with input and output prefixes, due to its incorporation of guarded atomic unification as its basic operation.

An appropriate surface syntax, PiFCP, has been devised for the polyadic π-calculus syntax. In this syntax pure ASCII code is used and many of the notations were modified for convenience. Importantly, PiFCP has been devised in such a way that it is clearly insulated from general Logix procedures. Thus, a pure π-calculus representation is maintained in spite of the use of an FCP-based platform. This distinction has not been clearly kept in previous implementations of the π-calculus and related formalisms.

We built a compiler from PiFCP to FCP. Similarly to previous implementations (Pierce & Turner 1998) the channels are represented by message streams, on which messages are written and from which messages are read and consumed. However, since synchronous communication is emulated, the messages are transformed with identification tags that ensure synchronous release of input and output guards as well as their appropriate consumption in cases of mutually exclusive choice. Each PiFCP process is transformed to an FCP procedure, and its channel set (global channels, arguments, newly declared channels and bound input channels) is identified. This allows full use of channels as in the original calculus: for global communication, as parameters in recursive definitions, newly restricted channels, and bound input channels, to be instantiated only following communication.

Several debugging tools are available for tracing stepwise execution of PiFCP programs. These include tree traces and step-by-step execution mode, with the ability to set specific break points. The level of detail in which a system is traced (process, channels, messages, senders, etc.) can be determined dynamically throughout a session. Thus, not only the net outcome of a computation can be studied, but also the specific scenario that has led to this outcome. In this sense, a π-calculus representation of a pathway can be studied by the same tools used by programmers to evaluate the behavior of programs. The PiFCP system is therefore a unique tool for following pathways represented in the π-calculus.

An example for the possibilities offered by this tool is summarized in Table 4. Due to the strong correspondence between molecular biology and π-calculus representations, detailed above, simulation studies of pathways behavior can be performed in an analogous fashion to laboratory experiments. The visible nature of the representation lends itself naturally to the modeling of various mutations. For example, a representation (programs) of a deletion mutant (bug) is created from the wildtype one by removal of the corresponding process. The resulting behavior of different mutants can

(1) $ERK1 ::= (\mathsf{new}\ backbone\ atp\_binding\_site\ erk\_kinase)$
$(ERK1\_FREE\_Nt\_LOBE \mid ERK1\_CATALYTIC\_CORE \mid ERK1\_FREE\_Ct\_LOBE)$

(2) $ERK1\_CATALYTIC\_CORE ::=$
$(ERK1\_ATP\_BINDING\_SITE \mid ERK1\_FREE\_ACTIVE\_SITE \mid ERK1\_T\_LOOP)$

(3) $ERK1\_T\_LOOP(thr, tyr) ::= DIRECT + INDIRECT$

(4) $DIRECT ::=$
$thr(cros\_enzyme).cross\_enzyme(thr).\overline{backbone}\langle thr, tyr\rangle.ERK1\_T\_LOOP +$
$tyr(cros\_enzyme).cross\_enzyme(tyr).\overline{backbone}\langle thr, tyr\rangle.ERK1\_T\_LOOP$

(5) $MEK1 ::= (\mathsf{new}\ backbone\ atp\_binding\_site\ mek\_kinase)$
$(MEK1\_FREE\_MP1\_BINDING\_SITE \mid MEK1\_CATALYTIC\_CORE)$

(6) $MEK1\_CATALYTIC\_CORE ::=$
$(MEK1\_ATP\_BINDING\_SITE \mid MEK1\_LIP\_REG\_KINASE \mid MEK1\_ACTIVATION\_LIP)$

(7) $MEK1\_LIP\_REG\_KINASE ::=$
$backbone(residue_1, residue_2).[(residue1, residue2) = (p - ser, p - ser)].$
$MEK1\_ACTIVE\_KINASE$

(8) $MEK1\_ACTIVE\_KINASE ::=$
$atp\_binding\_site.(NON\_PROCESSIVE\_KINASE + PROCESSIVE\_KINASE)$

(9) $NON\_PROCESSIVE\_KINASE ::= \overline{thr}\langle mek\_kinase\rangle.\overline{mek\_kinase}\langle p - thr\rangle.MEK1\_ACTIVE\_KINASE +$
$\overline{tyr}\langle mek\_kinase\rangle.\overline{mek\_kinase}\langle p - tyr\rangle.MEK1\_ACTIVE\_KINASE +$
$backbone(residue_1, residue_2).[(residue1, residue2) \neq (p - ser, p - ser)].$
$MEK1\_LIP\_REG\_KINASE$

Figure 2: The interaction between the ERK-1 T-loop and the MEK-1 catalytic site: A fragment of a π calculus representation of the RTK-MAPK ST pathway

then be studied by simulation and debugging. For example, as shown in Table 4 the effect of a dominant negative mutant process can be followed by tracing the subsequent behavior of receptor processes. The quantitative dependency on the amount of mutant ligand can be compared with previous experimental results. Similar quantitative and qualitative effects can be followed for other types of mutations, to determine critical steps and dynamics of a pathway. We are currently actively following such "mutants" (unpublished data). Note, that the robustness and of the syntax could also potentially allow automatic introduction of such "mutations" (bugs) into representations (programs), for simulated evolution studies.

Formal analysis of algebraic models of ST pathways

In addition to simulation studies, a major advantage of a specification-based approach is that the behavioral properties of represented systems, can also be analyzed, using methods and tools for formal verification (Francez 1992; Milner 1999). Thus, desirable outcomes and properties of an ST system, which is represented in the π-calculus can be formally proven.

Furthermore, the π-calculus behavioral theory allows to formally compare two programs, in order to determine the degree of mutual similarity of their behavior, termed *bisimulation*. Different levels of similarity, of weakening strength, have been defined (Milner 1999).

This opens up completely novel possibilities in the study of ST pathways. First, since formal π-calculus representations include both molecular details (domains, residues, etc.) and the behavior that these underly, comparison of similar pathways is the first step in studying the evolution not only at the level of a single structure, but at the level of the process (Gilbert, Opitz, & Raff 1996), establishing a *homology of processes*. Such comparative studies may be extendible to processes at higher levels of organization. Second, while in this paper we have represented pathways at the molecular levels, an ST pathway may also be viewed at a higher, functional level. Formal comparison methods can also be used in order to prove that the molecular information indeed supports the behavior we expect and specify.

ST	π-calculus	Example		
		ST	π-calculus	Observation
Deletion (insertion) of domains or residues	Removal (addition) of processes or prefixes	A dimer ligand	$LIGAND ::=$ (new $ligand$) $(BINDING\_DOMAIN \mid BINDING\_DOMAIN)$	Signaling to receptor
		A single domain ligand	$LIGAND ::=$ (new $ligand$) $(BINDING\_DOMAIN)$	Reduction of signal with increase in mutant number (dominant negative)
Conversion of residues	Change of channel names	MEK-1 lip with two serines	$MEK1\_ACTIVATION\_LIP ::= LOOP(ser,\ ser, \cdots)$	Signaling (phosphorylation) to ERK, only when ligand is present
		MEK-1 lip with two glutamates	$MEK1\_ACTIVATION\_LIP ::= LOOP(ala,\ ala, \cdots)$	signaling (phosphorylation) to ERK, independent of presence of ligand
Chimeric combination of domains	Combination of two processes sharing common restricted channels	A chimera of the ERK-1 catalytic core and Ct lobe with the p38 Nt lobe	$ERK1\_P38\_CHIMERA ::=$ (new $backbone\ atp\_binding\_site\ erk1\_kinase$) $(P38\_FREE\_Nt\_LOBE \mid ERK1\_CATALYTIC\_CORE \mid ERK1\_FREE\_Ct\_LOBE)$	The growth factor ligand affects stress-related gene expression

Table 4: ST mutational analysis in the π-calculus

Currently developed formal methods and automated tools (Orava & Parrow 1992; Victor 1994; Victor & Moller 1994) to determine the behavioral equivalence of processes provide a starting point to formalizing these notions.

Conclusions and future prospects

Modeling biological processes with a process algebra, such as the representation of ST pathways in the π-calculus is unique among existing approaches in providing a comprehensive theory in which to formally describe, simulate, analyze, and compare processes. The strength of this approach stems from treating molecular entities as computational processes, which allows us to represent a process based on the behavior of the structural entities which participate in it. In this way, molecular detail and dynamic behavior are unified within a single description.

A related approach was recently taken when an older CS model of concurrency, Petri nets, was used for the modeling of metabolic pathways (e.g. Goss98). Unfortunately, Petri nets are still lacking in their representation of the structural details of processes (Milner 1999). Thus, they do not scale well to ST pathways, where complex molecules serve both as enzymes and as compounds. However, the successful representation of metabolic pathways with Petri nets highlights the potential of models of concurrency to deal with biomolecular processes.

The formal operational semantics of the calculus renders these descriptions amenable to execution and simulation as well as to formal verification and comparison. The use of formal and algorithmic approaches has greatly accelerated progress in the sequence and structure branches of biology. Thus, adopting a common representation language for biomolecular processes may similarly accelerate progress in understanding their function and evolution. However, the previously developed tools for simulation and analysis of π-calculus programs required further enhancement in order to serve the molecular realm. In this work, we presented the PiFCP system, for the simulation and tracing of π-calculus programs for ST networks. We are currently developing a stochastic version of this simulation tools, based on the stochastic π-calculus (Priami 1995). This would allow more quantitative accuracy in simulation studies.

The current theory of behavioral equivalence in the calculus (Milner 1999) may also be modified to fit our notion of homology. For example, a numeric measure of distance should be added to this qualitative theory. Note, that contrary to simulation tools, the full calculus is supported by an existing verification tool, the Mobility Workbench (Victor & Moller 1994).

Finally, this approach can be extended to other realms of biology in higher levels of organization, where the principles of concurrent entities which interact and modify each others can be applied. Such scaling, if successful, can form the theoretical foundation to bridge the gap between molecular biology and physiology, within the unifying concept of the biological process.

References

Alon, U.; Surette, M. G.; Barkai, N.; and Leibler, S. 1999. Robustness in bacterial chemotaxis. *Nature* 397(6715):168–171.

Baeten, J. C., and Weijland, W. P. 1990. *Process Algebra.* Cambridge: Cambridge University Press.

Barkai, N., and Leibler, S. 1997. Robustness in simple biochemical networks. *Nature* 387(6636):913–917.

Boudol, G. 1992. Asynchrony and the π-calculus (note). Rapport de Recherche 1702, INRIA Sophia-Antipolis.

Bray, D.; Levin, M. D.; and Morton-Firth, C. J. 1998. Receptor clustering as a cellular mechanism to control sensitivity. *Nature* 393(6680):85–88.

Bray, D. 1997. Reductionism for biochemists: how to survive the protein jungle. *Trends in Biochemical Sciences* 22(9):325–326.

Brunet, A., and Pouysséegur, J. 1997. Mammalian MAP kinase modules: how to transduce specific signals. *Essays in Biochemistry* 32:1–16.

Canagarajah, B. J.; Khokhlatchev, A.; Cobb, M. H.; and Goldsmith, E. J. 1997. Activation mechanism of the map kinase erk2 by dual phosphorylation. *Cell* 90:859–869.

Eilbeck, K.; Brass, A.; Paton, N.; and Hodgman, C. 1999. Interact: An object-oriented protein-protein interaction database. In Lengauer, T.; Schneider, R.; Bork, P.; Brutlag, D.; Glasgow, J.; Mewes, H.-W.; and Zimmer, R., eds., *Proccedings of Seventh International Conference on Intelligent Systems for Molecular Biology*, 87–94. Menlo Park, California: AAAI Press.

Fontana, W., and Buss, L. W. 1996. The barrier of objects: From dynamical systems to bounded organizations. In Casti, J., and Karlqvist, A., eds., *Boundaries and Barriers.* New York: Addison Wesley. 56–116.

Fournet, C., and Gonthier, G. 1996. The reflexive chemical abstract machine and the join-calculus. In Steele, G., ed., *23rd Annual Symposium on Principles of Programming Languages (POPL)*, 372–385.

Francez, N. 1992. *Formal Verification.* New York: Addison Wesley.

Gilbert, S. F.; Opitz, J. M.; and Raff, R. A. 1996. Resynthesizing evolutionary and developmental biology. *Developmental Biology* 173(2):357–372.

Gilbert, S. F. 1997. *Developmental Biology, 5^{th} edition.* Sunderland, Massachusetts: Sinauer Associates.

Heldin, C.-H., and Purton, M., eds. 1996. *Signal Transduction.* London: Chapman and Hall. Modular Texts in Molecular and Cell Biology 1.

Honda, K., and Tokoro, M. 1991. An object calculus for asynchronous communication. In America, P., ed., *European Conference on Object-Oriented Programming (ECOOP'91)*, volume 512 of *Lecture Notes in Computer Science*, 133–147. Berlin: Springer-Verlag.

Igarashi, T., and Kaminuma, T. 1997. Development of a cell signalling networks database. In Altman, R. B.; Dunker, A. K.; Hunter, L.; and Klein, T. E., eds., *Proccedings of the Pacific Symposium of Biocomputing '97*, 187–197. Singapore: World Scientific Press.

Karp, P. D.; Riley, M.; Paley, S. M.; Pellegrini-Toole, A.; and Krummenacker, M. 1997. EcoCyc: Encyclopedia of *Escherichia coli* genes and metabolism. *Nucleic Acids Research* 25(1):43–50.

Kauffman, S. A. 1993. *The Origins of Order: Self-Organization and Selection in Evolution.* New York: Oxford University Press.

Kolpakov, F. A.; Ananko, E. A.; Kolesov, G. B.; and Kolchanov, N. A. 1998. GeneNet: A gene network database and its automated visualization. *Bioinformatics* 14(6):529–537.

Lewis, T. S.; Shapiro, P. S.; and Ahn, N. G. 1998. Signal transduction through MAP kinase cascades. *Advances in Cancer Research* 74:49–139.

McAdams, H. H., and Arkin, A. 1997. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Sciences USA* 94(3):814–819.

Milner, R.; Parrow, J.; and Walker, D. 1992. A calculus of mobile processes, part I/II. *Journal of Information and Computation* 100:1–77.

Milner, R. 1989. *Communication and Concurrency.* New York: Prentice Hall. Prentice Hall International Series in Computer Science.

Milner, R. 1993. The polyadic π-calculus: A tutorial. In Bauer, F. L.; Brauer, W.; and Schwichtenberg, H., eds., *Logic and Algebra of Specification, Proceedings of International NATO Summer School (Marktoberdorf, Germany, 1991)*, volume 94 of *NATO ASI Series F*, 428–440. Berlin: Springer-Verlag. Available as Technical Report ECS-LFCS-91-180, University of Edinburgh, October 1991.

Milner, R. 1999. *Communicating and Mobile Systems: The π-Calculus.* Cambridge: Cambridge University Press.

Orava, F., and Parrow, J. 1992. An algebraic verification of a mobile network. *Formal Aspects of Computing* 4(6):497–543.

Palamidessi, C. 1997. Comparing the expressive power of the synchronous and the ssynchronous π-calculus. In *24th Annual Symposium on Principles of Programming Languages (POPL)*, 256–265. ACM Press.

Parrow, J., and Sangiorgi, D. 1995. Algebraic theories for name-passing calculi. *Journal of Information and Computation* 120(2):174–197.

Pierce, B. C., and Turner, D. N. 1998. Pict: A programming language based on the pi-calculus. In Plotkin, G.; Stirling, C.; and Tofte, M., eds., *Proof,* *Language and Interaction: Essays in Honour of Robin Milner*. Cambridge, Massachusetts: The MIT Press.

Priami, C. 1995. Stochastic π-calculus. *The Computer Journal* 38(6):578–589. Proceedings of PAPM '95, the 3rd Workshop on Process Algebras and Performance Modelling.

Salamonsen, W.; Mok, K. Y. C.; Kolatkar, P.; and Subbiah, S. 1999. BioJAKE: A tool for the creation, visualization and manipulation of metabolic pathways. In Altman, R. B.; Dunker, K. L. A. K.; Hunter, L.; and Klein, T. E., eds., *Proccedings of the Pacific Symposium of Biocomputing '99*, 392–400. Singapore: World Scientific Press.

Schaeffer, H. J.; Catling, A. D.; Eblen, S. T.; Collier, L. S.; Krauss, A.; and Weber, M. J. 1998. Mp1: A mek binding partner that enhances enzymatic activitation of the map kinase cascade. *Science* 281:1668–1671.

Shapiro, E. 1987. Concurrent prolog: a progress report. In Shapiro, E., ed., *Concurrent Prolog (vol. I)*. Cambridge, Massachusetts: MIT Press. 157–187.

Silverman, W.; Hirsch, M.; Houri, A.; and Shapiro, E. 1987. The Logix system user manual, version 1.21. In Shapiro, E., ed., *Concurrent Prolog (vol. II)*. Cambridge, Massachusetts: MIT Press. 46–78.

Smith, T. F. 1998. Functional genomics - bioinformatics is ready for the challenge. *Trends in Genetics* 14(7):291–293.

Szallasi, Z., and Liang, S. 1998. Modelling the normal and neoplastic cell cycle with "realistic boolean genetic networks": Their application for understanding carcinogenesis and assessing therapeutic strategies. In Altman, R. B.; Dunker, A. K.; Hunter, L.; and Klein, T. E., eds., *Proccedings of the Pacific Symposium of Biocomputing '98*, 66–76. Singapore: World Scientific Press.

Victor, B., and Moller, F. 1994. The Mobility Workbench — A tool for the π-calculus. In Dill, D., ed., *Computer Aided Verification (Proceeding of CAV'94)*, volume 818 of *Lecture Notes in Computer Science*, 428–440. Berlin: Springer-Verlag.

Victor, B. 1994. *A Verification Tool for the Polyadic π-Calculus*. Licentiate thesis, Department of Computer Systems, Uppsala University, Sweden. Available as report DoCS 94/50.

Widmann, C.; Gibson, S.; Jarpe, M. B.; and Johnson, G. L. 1999. Mitogen activated protein kinase: Conservation of a three-kinase module from yeast to human. *Physiological Reviews* 79(1):143–180.

Wilson, E. O. 1998. *Consilience: The unity of knowledge*. London: Abacus.

Appendix A: Detailed π-calculus programs for the MEK1 and ERK1 molecules

MEK1 dual specificity kinase

(10) $MEK1 ::= (\mathsf{new}\ mek\ backbone1\ backbone2\ atp\_binding\_site\ mek\_kinase)$
$(MEK1\_FREE\_MP1\_BINDING\_SITE \mid MEK1\_CATALYTIC\_CORE)$

(11) $\overline{MEK1\_FREE\_MP1\_BINDING\_SITE ::=}$
$mp1\_prs(cross\_mp1, cross\_mp2, cross\_mp3).\overline{cross_m p1}\langle mek\rangle.$
$MEK1\_BOUND\_MP1\_BINDING\_SITE$

(12) $MEK1\_BOUND\_MP1\_BINDING\_SITE ::= (\mathsf{new}\ a)$
$(RESTRICTED\_BINDING(a,\ cross\_mp2,\ cross\_mp3,\ mek\_kinase,\ tyr,\ thr,\ backbone3) \mid$
$a.backbone3.\overline{mek}.MEK1\_FREE\_MP1\_BINDING\_SITE)$

(13) $MEK1\_CATALYTIC\_CORE ::=$
$(MEK1\_ATP\_BINDING\_SITE \mid MEK1\_ACTIVE\_SITE \mid MEK1\_ACTIVATION\_LIP)$

(14) $MEK1\_ACTIVATION\_LIP(ser,\ ser,\ backbone1,\ backbone2) ::=$
$ACTIVATION_LOOP(ser,\ ser,\ backbone1,\ backbone2)$

(15) $MEK\_ATP\_BINDING\_SITE ::= ATP\_BS(atp,\ atp\_binding\_site)$

(16) $MEK1\_ACTIVE\_SITE ::= LIP\_REGULATED\_KINASE\_ACTIVE\_SITE(mek\_kinase,$
$atp\_binding\_site,\ p-ser,\ p-ser,\ ser,\ p-ser,\ thr,\ p-thr,\ backbone2,\ backbone3)$

ERK1 ser/thr kinase

(17) $ERK1 ::= (\mathsf{new}\ erk\ erk\_nt\ backbone1\ backbone2\ backbone3\ atp\_binding\_site\ erk\_kinase)$
$(ERK1\_FREE\_Nt\_LOBE \mid ERK1\_CATALYTIC\_CORE \mid ERK1\_FREE\_Ct\_LOBE)$

(18) $ERK1\_FREE\_Nt\_LOBE ::= mp1\_erk1(cross\_mp1, cross\_mp2, cross\_mp3).\overline{cross\_mp1}\langle erk1\rangle.$
$ERK1\_MP1\_BOUND\_Nt\_LOBE$

(19) $ERK1\_MP1\_BOUND\_Nt\_LOBE ::= (\mathsf{new}\ a)$
$(RESTRICTED\_BINDING(a,\ cross\_mp2,\ cross\_mp3,\ erk\_kinase,\ thr,\ ser,\ backbone1) \mid$
$a.backbone1.\overline{erk}.ERK1\_FREE\_Nt\_LOBE)$

(20) $ERK1\_CATALYTIC\_CORE ::=$
$(ERK1\_ATP\_BINDING\_SITE \mid ERK1\_FREE\_ACTIVE\_SITE \mid ERK1\_T\_LOOP)$

(21) $ERK1\_T\_LOOP(thr,\ tyr,\ backbone1,\ backbone2) ::=$
$ACTIVATION\_LOOP(thr,\ tyr,\ backbone1,\ backbone2)$

(22) $ERK1\_ATP\_BINDING\_SITE ::= ATP\_BS(atp, atp\_binding\_site)$

(23) $ERK1\_ACTIVE\_SITE ::= LIP\_REGULATED\_KINASE\_ACTIVE\_SITE(erk\_kinase,$
$atp\_binding\_site,\ p-thr,\ p-tyr,\ ser,\ p-ser,\ thr,\ p-thr,\ backbone2)$

(24) $ERK1\_FREE\_Ct\_LOBE ::= (\mathsf{new}\ a)$
$(BINDING(a, erk\_srs, srs\_erk, erk\_nt, erk\_kinase, thr, ser, backbone3)|$
$a.backbone3.ERK1\_FREE\_Ct\_LOBE)$

MP1 scaffold molecule

(25) $MP1 ::= (\mathsf{new}\ mp1\ mp2\ mp3\ mp4)$
$(FREE\_MEK\_BS|(FREE\_ERK\_BS + FREE\_RAF\_BS))$

(26) $FREE\_MEK\_BS ::=$
$\overline{mp1\_prs}\langle mp1, mp3, mp4\rangle.mp1(cross\_mol).cross_m ol.FREE\_MEK\_BS$

(27) $FREE\_ERK\_BS ::=$
$\overline{mp1\_erk}\langle mp2, mp4, mp3\rangle.mp2(cross\_mol).cross_m ol.FREE\_ERK\_BS + FREE\_RAF\_BS$

(28) $FREE\_RAF\_BS ::=$
$\overline{mp1\_raf}\langle mp2, mp4, mp3\rangle.mp2(cross\_mol).cross_m ol.FREE\_ERK\_BS + FREE\_RAF\_BS$

Generic binding processes

(29) $RESTRICTED\_BINDING(a,\ mot1,\ mot2,\ enzyme,\ res1,\ res2,\ bb) ::=$
$RES\_BIND1(mot1,\ mot2) + RES\_BIND2(mot2,\ mot1)$

(30) $RES\_BIND1(mot1,\ mot2) ::=$
$mot1(cross\_enzyme, cross\_res1, cross\_res2).\overline{mot2}\langle enzyme, res1, res2\rangle.$
$\overline{bb}\langle cross\_enzyme, cross\_res1, cross\_res2\rangle.\overline{a}$

(31) $RES\_BIND2(mot1,\ mot2) ::=$
$\overline{mot2}\langle enzyme, res1, res2\rangle.mot1(cross\_enzyme, cross\_res1, cross\_res2).$
$\overline{bb}\langle cross\_enzyme, cross\_res1, cross\_res2\rangle.\overline{a}$

(32) $BINDING(a,\ mot1,\ mot2,\ new\_mot,\ enzyme,\ res1,\ res2,\ bb) ::=$
$BINDING1 + BINDING2$

(33) $BINDING1 ::=$
$mot1(cross\_mot, cross\_enzyme, cross\_res1, cross\_res2).\overline{cross\_mot}\langle enzyme, res1, res2\rangle.$
$\overline{bb}\langle cross\_enzyme, cross\_res1, cross\_res2\rangle.\overline{a}$

(34) $BINDING2 ::=$
$\overline{mot2}\langle new\_mot, enzyme, res1, res2\rangle.new\_mot(cross\_enzyme, cross\_res1, cross\_res2).$
$\overline{bb}\langle cross\_enzyme, cross\_res1, cross\_res2\rangle.\overline{a}$

Generic enzymatic processes

(35) $ACTIVATION\_LOOP(res1,\ res2,\ bb1,\ bb2) ::=$
$DIRECT(res1,\ res2,\ bb1,\ bb2)\ +\ INDIRECT(res1,\ res2,\ bb1,\ bb2)$

(36) $DIRECT ::=$
$res1(cross\_enzyme).cross\_enzyme(res1').\overline{bb2}\langle res1', res2\rangle.$
$ACTIVATION\_LOOP(res1',\ res2,\ bb1,\ bb2)+$
$res2(cross\_enzyme).cross\_enzyme(res2').\overline{bb2}\langle res1, res2'\rangle.$
$ACTIVATION\_LOOP(res1,\ res2',\ bb1,\ bb2)$

(37) $INDIRECT ::=$
$backbone1(cross\_enzyme, cross\_motif1, cross\_motif2).$
$[cross_motif1 = res1].cross\_enzyme(res1').\overline{bb2}\langle res1', res2\rangle.INDIRECT(res1',\ res2,\ bb1,\ bb2)+$
$[cross_motif2 = res2].cross\_enzyme(res2').\overline{bb2}\langle res1, res2'\rangle.INDIRECT(res1,\ res2',\ bb1,\ bb2)+$
$[otherwise].\overline{bb1}.ACTIVATION\_LOOP$

(38) $ATP\_BINDING\_SITE(atp, atp\_bs) ::=$
$atp(adp).\overline{atp\_bs}.\overline{adp}.ATP_BS$

(39) $LIP\_REGULATED\_KINASE\_ACTIVE\_SITE(enzyme,\ atp\_bs,\ res1,\ res2,\ res3,$
$modres3,\ res4,\ modres4,\ bb1,\ bb2) ::=$
$bb1(res1', res2').[(res1', res2') = (res1, res2)].ACTIVE\_SITE$

(40) $ACTIVE\_SITE ::= atp\_bs.(DIRECT\_KINASE\ +\ INDIRECT\_KINASE)$

(41) $DIRECT\_KINASE ::=$
$\overline{res3}\langle enzyme\rangle.\overline{enzyme}\langle modres3\rangle.ACTIVE\_SITE+$
$\overline{res4}\langle enzyme\rangle.\overline{enzyme}\langle modres4\rangle.ACTIVE\_SITE+$
$bb1(res1', res2').[(res1', res2') \neq (res1, res2)].LIP\_REGULATED\_KINASE\_ACTIVE\_SITE$

(42) $INDIRECT\_KINASE ::= bb2(cross_enzyme, cross_res1, cross_res2).$
$(\overline{enzyme}\langle modres3\rangle.atp\_bs.INDIRECT\_KINASE+$
$\overline{enzyme}\langle modres4\rangle.atp\_bs.INDIRECT\_KINASE+$
$\overline{bb2}.ACTIVE\_SITE+$
$bb1(res1', res2').[(res1', res2') \neq (res1, res2)].\overline{bb2}.LIP\_REGULATED\_KINASE\_ACTIVE\_SITE)$