Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

DISCRETE EVENT SYSTEMS SPECIFICATION IN SYSTEMS BIOLOGY - A DISCUSSION OF STOCHASTIC PI CALCULUS AND DEVS

Adelinde M. Uhrmacher

University of Rostock
Albert Einstein Str. 21
Rostock 18059, GERMANY

Corrado Priami

University of Trento
Via Sommariva 14
Povo, TN 38050, ITALY

ABSTRACT

The goal of Systems Biology is to analyze the behavior and interrelationships between entities of entire functional biological systems. Discrete event approaches are of particular interest if small numbers of entities, like DNA molecules, shall be modeled. Two general approaches toward discrete event modeling and simulation are presented. They provide rather different perspectives on the system to be modeled, as is illustrated based on a model of the Trypophan Operon. Whereas in `Devs` distinctions are emphasized, e.g. between system and its environment, between structural and non structural changes, between properties attributed to a system and the system itself, these distinctions become fluent in the compact description of the π-`Calculus`. However, both share the problem that in order to support a comfortable modeling, adaptations and extensions according to the concrete requirements of this challenging application area are needed.

1 INTRODUCTION

The goal of Systems Biology is to analyze the behavior and interrelationships between entities of entire functional biological systems (Wolkenhauer 2001), a research area that has received increasingly attention over the last years. Particularly, based on the availability of more reliable and more fine grained data the research has gained momentum. Diverse modeling and simulation methods are being applied in the area of Systems Biology. Although continuous systems models are the dominant type of models being used in Systems Biology (de Jong 2002), stochastic discrete event models are recently gaining ground as well. They address specific constraints of continuous, deterministic models: concentrations do not necessarily change continuously, particularly if the dynamics of a small amount of entities, like DNA molecules and plasmids, shall be modeled (Kuo and Keasling 1996). In addition, the dynamics of some biological systems can be best approached in a stochastic manner, e.g. if the gene regulation is to be described, where stochastic fluctuations are abundant (Cowan 2003). However, so far most work has concentrated on a stochastic, discrete event execution of reaction equations, rather than on an explicit modeling in a discrete event modeling formalism. Many simulation systems, e.g. (Ramsey et al. 2005), offer already to execute reaction networks by numerical integration or by stochastic discrete event simulation on demand. For the latter the reaction rates are turned into reaction probabilities per unit time following the approach suggested by Gillespie (Gillespie 1977). However, whereas stochastic discrete event executions are widely employed and only critically inspected due to their lack of performance, discrete event modeling formalisms have a harder time to get accepted in the Systems Biology community. Unlike the continuous systems modeling and simulation realm, the discrete modeling and simulation realm lacks a common denominator for model description, even though general approaches exist. E.g. `Devs` (Zeigler, Praehofer, and Kim 2000), stochastic `Petri Nets` (Haas 2002), and stochastic π-`Calculus` (Priami 1995) (in the following also `Stochastic` π) are formal and generally applicable approaches toward discrete event systems modeling. Each has been developed with a rather different objective in mind. E.g. the goal of `Devs` has been to combine the functional, network, and hierarchical perspective in describing systems, and thus stands in the tradition of general systems theory. In contrast `Petri Nets` and π-`Calculus` have been developed for describing concurrent processes and are best known in the context of Computer Science. Whereas the relation between `Devs` and `Petri Nets` has been the subject of research before, e.g. recently (Bobeanu et al. 2004), the relation between `Devs` and stochastic π-`Calculus`, is largely unexplored. Our goal is to approach filling this gap by particularly focusing on gene regulation processes as part of Systems Biology.

2 STOCHASTIC π-`Calculus`

The stochastic π-`Calculus` (Priami 1995) is an extension of the π-calculus (Milner 1999) that copes with quantitative information to support temporal simulation of complex systems. The calculus is based on the notion of name that

represent both interconnection links between active entities called processes and the data that these entities exchange through communication. Communication channels are associated with probabilistic distributions to describe the quantitative evolution of the overall system.

The abstraction principles that drive the modeling of molecular interactions are as follows. Processes model molecules and domains (subcomponents of the molecules that can have active roles in chemical interactions). Global channel names and co-names represent complementary domains and newly declared private channels define complexes and cellular compartments. Communication and channel transmission model chemical interaction and subsequent modifications. The actual rate of a reaction between two proteins is determined according to a constant *basal rate* empirically-determined and the concentrations or quantities of the reactants . If two different reactant molecules, P and Q, are involved, the reaction rate is given by $r_b \times |P| \times |Q|$, where r_b is the reaction's basal rate, and $|P|$ and $|Q|$ are the numbers of P and Q in the chemical solution computed via the two auxiliary functions, In_x, Out_x that inductively count the number of receive and send operations on a channel x enabled in a process.

The syntax of the calculus follows

$$P ::= \mathbf{0} \mid (\pi, r_b).P \mid (\nu x)P \mid [x = y]P \mid P|P \mid P + P \mid A(y_1, \ldots, y_n) \qquad (1)$$

where π may be either $x(y)$ for *input*, or $\overline{x}y$ for *output* (where x is the *subject* and y is the *object*). $(\nu x)P$ means that x is a new name declared in P. The parameter r_b corresponds to the basal rate of a biochemical reaction and it is an exponential distribution associated to the channel occurring in π. The order of *precedence* among the operators is the order (from left to right) listed above.

The intuitive semantics of the operators follows. The prefix π is the first atomic action that the process $\pi.P$ can perform. The input prefix binds the name y in the prefixed process. Intuitively, some name y is received along the link named x. The output prefix does not bind the name y which is sent along x. Summation denotes nondeterministic choice. The operator | describes parallel composition of processes. The operator (νx) acts as a static binder for the name x in the process P that it prefixes. In other words, x is a unique name in P which is different from all the external names. Matching $[x = y]P$ is an `if-then` operator: process P is activated if $x = y$. Finally, each agent identifier A has a unique defining equation of the form $A(y_1, \ldots, y_n) = P$, where the y_i's are the only free names (see below) of P and $y_i \neq y_j$ if $i \neq j$. The parameter r associated with prefixes defines an exponential distribution according to which the probabilistic behavior is determined. The semantics of the calculus thereby defines the dynamic behavior of the modeled system driven by a *race condition*, yielding a probabilistic model of computation. All the activities enabled in a state compete and the fastest one succeeds. The continuity of exponential distributions ensures that the probability that two activities end simultaneously is zero.

The reduction semantics of the biochemical stochastic π-`Calculus` is

$$(\ldots + (\overline{x}\langle z\rangle, r_b).Q)|((x(y), r_b).P + \ldots) \xrightarrow{x, r_b \cdot 1 \cdot 1} Q|P\{z/y\}$$

$$\frac{P \xrightarrow{x, r_b \cdot r_0 \cdot r_1} P'}{P|Q \xrightarrow{x, r_b \cdot r_0' \cdot r_1'} P'|Q}, \begin{cases} r_0' = r_0 + In_x(Q) \\ r_1' = r_1 + Out_x(Q) \end{cases}$$

$$\frac{P \xrightarrow{x, r_b \cdot r_0 \cdot r_1} P'}{(\nu\, x)P \xrightarrow{x, r_b \cdot r_0 \cdot r_1} (\nu\, x)P'} \qquad \frac{Q \equiv P, P \xrightarrow{x, r_b \cdot r_0 \cdot r_1} P', P' \equiv Q'}{Q \xrightarrow{x, r_b \cdot r_0 \cdot r_1} Q'} \quad . \qquad (2)$$

A reaction is implemented by the three parameters r_b, r_0 and r_1, where r_b represents the basal rate, and r_0 and r_1 denote the quantities of interacting molecules, and are computed compositionally by In_x and Out_x. For instance the first rule describes how communication takes place. When two complementary actions are enabled (they can be executed, one is willing to send a message on a channel and another is willing to receive something on the same channel), the two prefixes are consumed. The resulting system is made up of the continuation after the process having discarded the alternatives.

3 DEVS

`Devs` has been developed as a general approach toward modeling and simulation discrete event systems (Zeigler, Praehofer, and Kim 2000). `Devs` distinguishes between atomic models and coupled models. Whereas atomic models describe the behavior in terms of state transitions that might be triggered by external events or the flow of time, coupled models define how their components, which might be atomic or coupled, interact with each other. Thereby, a hierarchical, modular construction of models is supported. Both atomic and coupled models communicate with their environment via input and output sets which are typically structured into ports. Therefore, Zeigler introduced so called structured sets. An abstract simulator defines the execution semantics of typical `Devs` models. In the following we will take the `PDevs` variant of `Devs` as a base (see (Zeigler et al. 2000) for further discussions). An atomic model is defined by

$$DEVS := \langle X^b, Y^b, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle \qquad (3)$$

where X^b, Y^b are structured multi sets of input and output respectively, S is the structured set of states, $\delta_{int} : S \to S$ describes the state transition triggered by the flow of time, i.e. the occurrence of an internal event, $\delta_{ext} : Q \times X^b \to S$ with $Q := \{(s, e) | s \in S \wedge 0 \leq e \leq ta(s)\}$ defines state transitions triggered by the arrival of "external" events, the confluent transition $\delta_{con} : S \times X^b \to S$ defines the transition

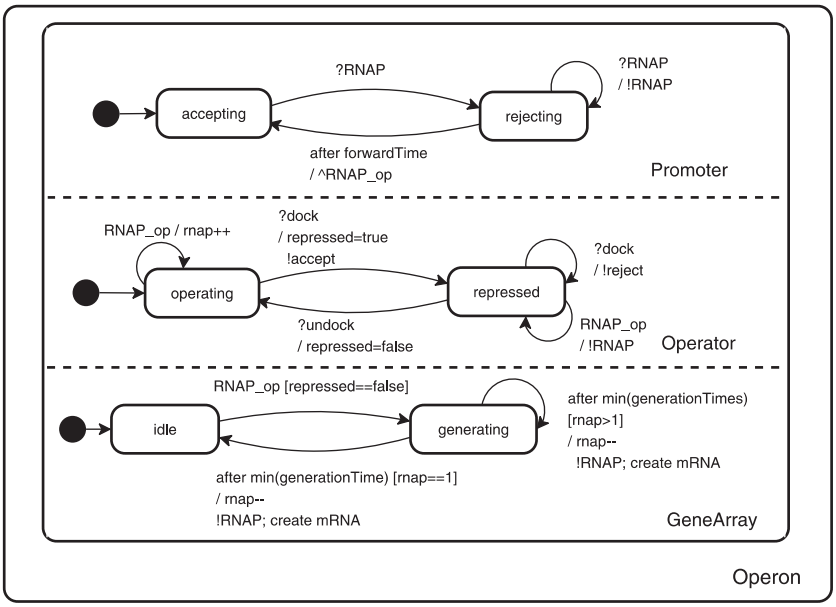

Figure 1: The Operon Model

in case internal and external events coincide (note for simplicity, the confluent transition function will in our examples be neglected assuming to be defined by successively calling the internal and thereafter the external transition function), the output function $\lambda : S \rightarrow Y$ determines the output given a state, and the time advance function $ta : S \rightarrow R_0^+$ associates with each state a time span after which an internal event is triggered, which implies calling the lambda function and thereafter executing the internal transition function. Stochastic aspects are included by using random variables of arbitrary random distributions in the different functions, e.g. in the time advance function to describe an exponential distributed basal rate.

$$CoupDEVS := \langle X^b, Y^b, D, Couplings \rangle \qquad (4)$$

where X^b, Y^b denote again the inputs and outputs, D the set of components which have to be again `Devs` models, $Couplings$ define the interactions that exist among the components and between the inputs and outputs of the coupled models and the components. As far as the domain of Systems Biology is concerned, the atomic models describe the active species of the cellular systems and the coupled models support grouping species and defining the interaction between them. Thus, a similar approach as in applying `Statecharts` in Systems Biology (Efroni et al. 2003) is taken. For a comparison of `Statecharts` and `Devs` in general see e.g. (Borland and Vangheluwe 2003).

4 APPLICATION

The Tryptophan (Trp) Operon is one of the most extensively studied systems for the examination of the prokaryotic gene regulation including the spatio-temporally progressing phenomena of gene repression, transcriptional attenuation and post translational enzyme feedback inhibition. The Trp operon comprises a promoter and an operator region and genes responsible for coding of the 5 enzymes that are needed to synthesize Trp. The promoter region has a binding site where the RNA polymerase can bind. This is the enzyme responsible for the transcription of genes. As long as the repressor is inactive it cannot bind to the operator of the Trp Operon. However, if the corepressor Trp binds to the inactive repressor protein, the allosteric repressor protein will change its shape causing it to become activated. In this state the repressor protein can bind to the operator region of the operon. With the active repressor protein bound to the operator region, RNA polymerase is unable to bind to the promoter region of the operon. Only if no repressor is bound to the operon, the transcription of the five genes into mRNA will be enabled. In the following we will only show parts of the model. For a detailed discussion of the behavior of the Tryptophan Operon as a continuous model based on delayed ODEs, see e.g. (Santill'an and Mackey 2001), as a discrete event model based on `Devs`, see e.g. (Degenring et al. 2004, Degenring et al. 2005), and a model focussing on the role of the attenuation based on π-`Calculus`, see e.g. (Kuttler to appear).

4.1 Operon in DEVS

The operon model can be represented as a `StateChart` as shown in Figure 1. (Note inputs are identified by the prefix "?", and outputs by the prefix "!"). In `Devs` the model is realized as a coupled model with three interacting atomic models (Figure 2).

```
Operon := < X, Y, D, Couplings >

  X := { (RnapIn, RepIn) |
           RnapIn ∈ { rnap, nil },
           RepIn ∈ { dock?, nil } },

  Y := { (RnapOut, RepOut) |
           RnapOut ∈ { rnap, nil }
           RepOut ∈ { accept!, reject!, nil } },

  D := { Promoter, Operator, Genearray },

  Couplings =
       { (self.RnapIn, Promoter.RnapIn),
          (Promoter.RnapOut, self.RnapOut),
          (Operator.RnapOut, self.RnapOut),
          (GeneArray.RnapOut, self.RnapOut),
          (self.RepIn, Operator.RepIn),
          (Operator.RepOut, self.RepOut),
          (Promoter.RnapOutOp, Operator.RnapIn),
          (Operator.RnapOutGe, GeneArray.RnapIn) }
```

Figure 2: Operon Model in Devs

After a certain time, the promoter forwards the incoming rnap to the Operator. If the Operator is repressed it will release the rnap molecule into the cytoplasm, if not it will forward the incoming rnap to the GeneArray. The operator is being repressed if a Repressor is coupled to the Operon, to be more precise to its Operator. Please note that the model components Promoter, Operator and Genearray are meant to work also for other types of operons, like e.g. the Lac Operon. Therefore, the model allows an rnap to temporarily bind to the Promoter even if a Repressor is docked to the Operator. In the presence of a repressor the Operator will turn it away. The GeneArray is able to process a series of rnap molecules concurrently, and thus to generate a series of mRNA models. These models will be newly generated and are themselves responsible for generating the enzyme models responsible for the production of the Tryptophan (Degenring et al. 2004). With each transcription of an individual mRNA, the molecules of type RNA polymerase are released to the cytoplasm. Please note that in the description below we did not include explicitly the δ_{con} function, by default δ_{con} is defined by a successive execution of δ_{ext} and δ_{int}.

The above definition of a promoter in Devs (Figure 3) illustrates one of the problems in defining a cell biological model in this formalism. The promoter might not be in the phase to accept an arriving RNA Polymerase, so it will turn it away. The arrival of the rnap triggers the external transition function being invoked. Intuitively one would like to remain in the phase rejecting and simply turn the rnap away. This turning away requires producing an output. However, in DEVS only at the time of an internal event an output can be produced (Zeigler et al. 2000). Thus, in order to reject the rnap the model has shortly to enforce an internal

```
Promoter := <X, Y, S,  delta_int, delta_ext, λ , ta>

 X := { RnapIn | RnapIn ∈ { rnap, nil } },
 Y := { (RnapOut, RnapOutOp) |
          RnapOut ∈ { rnap, nil },
          RnapOutOp ∈ { rnap, nil } },
 S :=  { (phase, remainTime) |
           phase ∈ { accepting, rejecting, dummy }
           remainTime ∈ R ∪ ∞  }

 ta (phase, remainTime) :=
      if phase = dummy then 0 else remainTime

 δ_int  (phase, remainTime) :=
      if phase = rejecting
           then (accepting,  ∞  )
           else (rejecting, remainTime)

 δ_ext  ((phase, remainTime), elapsedT, rnap) :=
         if phase = accepting
           then (rejecting, expRandom(forwardT))
           else (dummy, remainTime - elapsedT)

 λ (phase,remainTime) :=
         if phase = dummy
           then (rnap, nil) else (nil, rnap)
```

Figure 3: Promoter Model in Devs

transition to the dummy phase and afterward will return to the state (rejecting, remainTime) to continue its work. Therefore, the remaining time to finish its job when it was being interrupted, i.e. remainTime is calculated remainTime-elapsedT, to be used afterward when the model via the internal transition switches from dummy to rejecting. The Devs formalism helps rescheduling events by explicitly including in the invocation of the external transition function the elapsed time, i.e. the time that has passed since the last event, and thus the time to its next internal event. As a further example how in Devs cell biological models can be specified, we will give the description of the GeneArray (Figure 4).

This reveals another problem in applying the original Devs formalism as traditional Devs does not support variable structures. Neither exists a direct or visual support in Statecharts, they are integrated based on functional calls as kind of side effects. In the above Devs specification we integrated them just as a kind of special port mRnaOut that is actually not connected to any other model but allows us to signalize this kind of structural event. The events that are leaving via this port are of type VariableStructureRequest.

4.2 Operon in Stochastic π-Calculus

The operon model is represented by the parallel processes Promoter and *Operator* (Figure 5). The component GeneArray which was responsible for successively generating the mRNA is now represented by the

```
GeneArray := <X, Y, S,  delta_int, delta_ext, λ , ta>

 X :={ RnapIn | RnapIn  ∈  { rnap, nil }},
 Y :={ (RnapOut, mRnaOut) |
         RnapOut  ∈  { rnap, nil},
         mRnaOut ofType VariableStructureRequest},
 S :={ (phase, remainTime) |
         phase  ∈   { idle, generating }
         remainTime  ∈  2^R  }

 ta (phase, remainTime) :=
    if phase = idle
    then  ∞ else  minimum(remainTime)

 δ_int (phase, remainTime) :=
    if |remainT| = 1 then (idle, {∞})
    else (generating,
          remainTime \ minimum(remainTime))

 δ_ext ((phase, remainTime), elapsedT, rnap) :=
    if phase = idle
       then (generating, {expRandom(generationT)})
       else (generating,
             Subtract(remainTime, elapsedT) ∪
              { expRandom(generationT) })

λ(generating, remainTime) :=
    (rnap, VariableStructureRequest
                (Create,mRna,toFrom(Cytoplasm))
```

Figure 4: Genearray Model in `Devs`

```
Operon ::= Promoter | Operator

Promoter ::= (rnap, rnapT) .
    (rnapOp,opForwardT). Promoter

Operator ::=
     ((rnapOp, opForwardT).
         (Transcribing | (generate,generationT) |
                         Operator)) +
     ((dock(d),reprT). (d,unbindingT) . Operator)

Transcribing ::= (generate,generationT) . mRNA

mRNA ::= ...
RnaPolymerase ::= ...
```

Figure 5: Operon Model in `Stochastic` π

`Transcribing` process. After activation on the `rnap` channel, the `Promoter` interacts with the `Operator` on the channel `rnapOp` and then resumes its normal behavior. The `Operator` can now follow two distinct alternative paths: `((rnapOp ...)` – it can activate a copy of the `Transcribing` process, a new copy of itself and a control process of transcription that communicate with `Transcribing` via the `generate` channel (Note that the processes of type `Transcribing` are generated on demand.), or `((dock ...)` – it can receive a channel over which to synchronize to unbind and resume its behavior. The role of the promoter, i.e. forwarding RNA Polymerase molecules, to the operator region becomes evident.

4.3 Repressor in DEVS

Gene repression takes place, if an active form of the repressor docks onto free controlling sections of the DNA molecules, occupies them and prevent the mRNA polymerase from binding there and initiating the transcription. The repressor itself is activated by binding two Trp molecules per each repressor molecule, so that the process of repression is directly regulated by the Trp concentration. For simplicity in the model we assume that one `trp` corresponds to two Tryptophan molecules. The more often the repressor receives the two Trp molecules, i.e. `?Trp` (Figure 6, which means the higher the concentration of `Trp` in the cytoplasm, the more likely it is that the repressor will try to dock at the operon model.

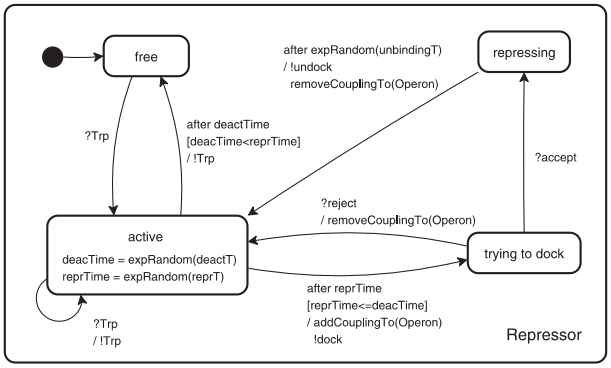

Figure 6: A State Chart of the Repressor

Therefore, the `Repressor` model installs an explicit coupling to `Operon`. If the docking request is confirmed by the operon model, the `Repressor` stays in phase repressing for a certain time. Afterward, the `Repressor` will dissolve its coupling to the `Operon` and will release the two Tryptophan molecules. Lets have a look how the backward and forward reaction from the state `active` to the state `free` or `trying to dock` is modeled (Figure 7), as forward and backward reactions are quite common in Systems Biology. With each state `Devs` associates a time span how long this state will persist per se. After this time it will call the output function to produce an output and the internal transition function to determine the next state. So intuitively the time advance function would be generating two exponentially distributed random variable based on `deacT` and `reprT` respectively and choose the smallest one. However, when the internal transition function would be invoked, it would have no idea whether to change to the state `free` or the state `trying to dock` as the only information available is the former state, i.e. active. Therefore, both generated random numbers `deacTime, reacTime` have to become part of the state to be accessible by the inter-

nal state transition. So the time advance function will return `minimum(deacTime, reprTime)`, and the internal transition function will determine based on the defined guards to which state to move. In installing an explicit docking to the `Operon`, the `Repressor` has to know the name of the model it wants to dock to. When initialized each repressor is given the name of the one and only operaton, i.e. `Operon` in the cell model. Otherwise these information has to be communicated similarly as in stochastic π.

```
Repressor :=  ...

 S :=  { (phase, deacTime, reprTime) |
           phase ∈ { free, active, dummy,
                     tryingToDock,
                     repressing, releasing }
           deacTime, reprTime  ∈    R    }
 ...
 ta (phase, deacTime, reprT) :=
    if phase = active then min(deacTime, reprTime)
 ...
 δint (phase, deacTime, reacTime) :=
    if phase = active then
        if deacTime < reacTime then (free, ...)
        else (tryingToDock, ...)
 ...
 δext ((phase,deacTime,reacTime),elapsedT,trp) :=
    if phase = free then
       (active, expRandom(deacT), expRandom(reacT))
 ...
 λ (phase, deacTime, reacTime) :=
     if phase = active then
        if deacTime < reacTime then (trp, nil)
        else (nil, VariableStructureRequest
                  (AddCoupling, toFrom(Operon))
```

Figure 7: Repressor Model in `Devs`

4.4 Repressor in Stochastic π-Calculus

Tryptophan molecules, repressors and operon are all processes. In the beginning the only channel known is the `trp` port and the `dock` port. Please note that the interaction occurs randomly between repressors and tryptophan processes. The interaction is synchronous, which implies identifying a matching pair of repressor and tryptophan processes and triggering the reduction rule that will be fired with a certain delay. The delay is calculated based on an exponentially distribution taking the rate of the corresponding channel as its mean. Interestingly also here the passing of names is crucial: not the names of the processes involved but the names of the channels. Those channels are generated dynamically and frequently. Channel creations are the means for distinguishing between different phases of the repressor, i.e. `free`, `active`, `repressing`, and `releasing` and associating with these different phases certain time spans. So instead of a time advance function that associates with a state a time span, now a synchronous interaction is introduced which naturally will put both processes on hold until completed. Distinguishing a phase `trying to dock` is no longer needed. As the interaction will only occur if the repressor's request $\overline{dock}$ (its output) is met by the operator's request for an input $dock$, it is guaranteed that the operon is in the correct state for the docking operation to occur.

```
Repressor ::= (trp(c),reactionT) .
   ((c̄,releaseT). Repressor) +
   ((ν freeRepr)(dock̄(freeRepr),reprT).
         (freeRepr̄,unbindingT) .
         (c̄,releaseT). Repressor))

Tryptophan ::=
   ((ν release) (trp̄(release),reactionT) .
        (release,releaseT) . Tryptophan
```

Figure 8: Repressor Model in `Stochastic` π

4.5 Structure of the Overall Model

Figure 9 shows the structure of the Tryptophan Operon model as a multi-level model. The `Cytoplasm` is a coupled model that describes at macro level, i.e. at concentration levels, species like Trp, Rnap, Indole, IGP, Serine, G3P, and `Ribosom`, which are exchanged between the models. Thus, it launches molecules to the respective models at times that are calculated based on the reaction rate and the number of reactands, following Gillespie (Gillespie 1977). As the events of certain reactions to occur are calculated in advance similarly to the approach presented in (Gibson and Bruck 2000), the events have to be rescheduled. In this context again the availability of the elapsed time is crucial. The coupling between cytoplasm and the individual repressor models, enzyme models and mRNA models are realized by a special type of coupling: multi-couplings which randomly select a coupling each time an output is generated by the `cytoplasm`. It should be noted, that `Cytoplasm, Enzyme, Operon`, and `mRNA` are all coupled models, so the model is hierarchically structured. The structure of the overall model in π-`Calculus`, if depicted, would show itself as a set of processes of different types, signalizing at their interfaces their interest in certain communications, some of them randomly and spontaneously involved in communication. The structure would be overall flat. This however would change at the moment extensions of the stochastic π-`Calculus`, e.g. `Ambients` (Cardelli and Gordon 2000) or `Beta Binders` (Priami and Quaglia 2005) would come into play.

5 DISCUSSION

Both modeling formalisms provide quite different perceptions on the system to be modeled.

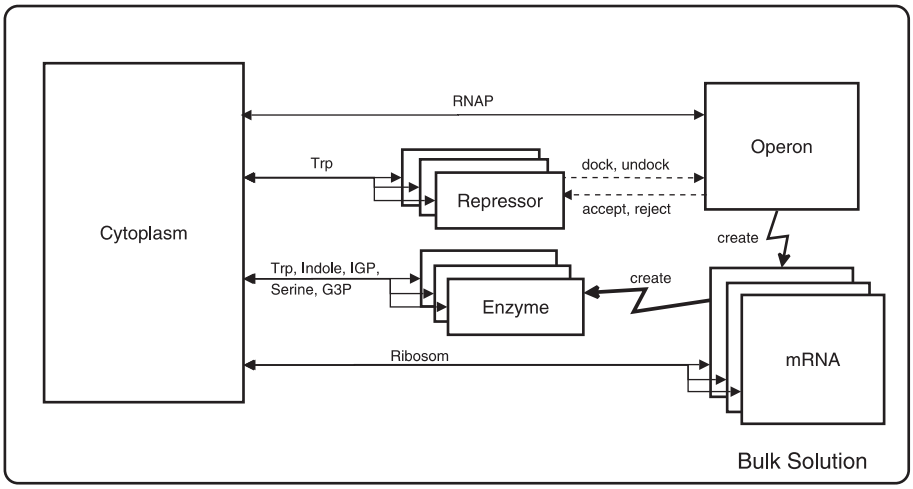

Figure 9: Overall Structure of the Devs Model

5.1 Distinction: System and Environment

Prominent in `Devs` is the clear distinction between system and environment by encapsulating the attributes and methods of a model and constraining the interaction between system and environment to ports. The knowledge of a system ends at its ports, it does not even know with whom it is interacting. Only the superior coupled model is privy to that information. On the other side from outside nothing is known about the internal state, if the model has not communicated it before. Already in the given example it becomes evident that this information hidden in `Devs` causes an undesirable overhead in modeling and simulating cell biological reactions. On the other side the encapsulation and modular design facilitates the step wise understanding of the system. Encapsulation, hierarchical design, together with the notion of describing the species as a kind of entity being in discrete states are very intuitive, particularly if visualized by `Statecharts`, as has been also emphasized in (Efroni et al. 2005).

In `Stochastic` π, processes rather than systems are described. A system is obtained as composition of elementary processes. This notion of compositionality is a key feature of the formalism because it allows the user to incrementally define the whole set of systems of interest by adding information to the part already developed.

As each process is able to generate new channels and new processes, the interface to the environment is frequently changing. Most of what happens within a `Devs` model becomes in `Stochastic` π an interaction directed to the environment. State transitions in `Devs` manifest themselves as communications in `Stochastic` π. As the interface is frequently changing the distinction between system and environment is fluent, in the sense that the interface changes during simulation, so what does belong and does not belong to a system changes over time. Whereas thereby, the distinction between system and environment is not facilitated, phenomena of changing interfaces that are particularly of interest in Systems Biology can easily be described.

The interface determines which reaction can occur between two species. This is even getting more pronounced by another extension of π-`Calculus`, i.e. `Beta Binders`. The new formalism allows to encapsulate pi-processes within boxes equipped with types that drive potential interactions based on the notion of affinity. This releases the exact complementarity required in `Stochastic` π that does not mimick what really happen in a biological setting. In addition `Beta Binders` provide additional structure within `Stochastic`-π models.

5.2 Structural and non Structural Changes

Whereas structural changes as changing interaction patterns and compositions are the exception in traditional modeling formalisms - if they are supported at all - in π-`Calculus` they are the rule. State transitions turn into the generation of channels. The successive processing of a number of RNA polymerase is modeled by a successive generation of transcription processes.

Traditional `Devs` does not support variable interaction, composition, and behavior pattern. Different extensions of the formalisms exist, e.g. (Barros 1997, Uhrmacher 2001), which support the change of structure as specific events, and likely even more implementations do exist. Surely, `Devs` was one of the first modeling and simulation formalisms, in which the need to support variable structures was stressed (Zeigler 1986). However, none of these realizations does allow a similarly seamless integration of structural and non structural changes as the `Stochastic` π-`Calculus`.

5.3 Micro-, Macro-, and Multilevel Modeling

`Devs` does not only clearly distinguish between structural and non structural events, but also between "active" entities like enzymes, repressors, mRNA and operon and the "pas-

sive" entities like the tryptophan molecules, serine etc. In the stochastic π-Calculus all entities are processes. Possibly processes could be distinguished though: e.g. whether processes have an own thread of control which is expressed in their continuation, e.g. Repressor, Operator, Genearray, or whether they are simply invoked for a task to be completed, e.g. Transcribing.

However, the π-Calculus does not lend itself to describing part of a system as interacting individuals and part of it as changing concentrations. Thus, no multi-level modeling is directly supported in π-Calculus like it is in Devs. Although one has to note that in Devs a multi-level model, that describes a system at different levels of organization, is not supported via the coupled models as one might assume. Coupled models do not have a behavior of their own. They are aimed at supporting a modular, hierarchical construction of models rather than the description of dynamics at different hierarchical levels of organization. Coupled models can be used to define compartments and locations within the cell, similar e.g. to the boxes in Beta Binders. Thus, they structure the cellular space in a discrete manner, defining regions of increased interaction. However, models which monitor and update concentrations of species at macro level are defined as atomic models, same as the individual micro models. For rescheduling events at macro level again the elapsed time information becomes essential.

One of the advantages of simulating entities individually is the possibility to track one individual if required. To track an individual is possible in Devs and Stochastic π. Both trace specific entities by relying on the names of processes and models respectively.

5.4 Communication

The communication of the π-Calculus reflects well how biological systems are assumed to interact. Indeed it can occur only if both interacting partners are willing to do so (synchronous). Furthermore, the quantitative information associated with channels drives the stochastic behavior through races.

In Devs, an event will arrive asynchronously. It is checked whether e.g. the operator is in the right phase to accept the repressor, if not the docking request will be denied. In π-Calculus, it is known from the outside whether or not the repressor is in the correct state. Also selecting randomly among those available, mimics the idea of randomly moving molecules quite well.

In Devs, the interaction between models are defined by couplings. If more than one model is connected to the output of a model the output will be cloned and reach all. This is similar to the idea of broadcasting events in Statecharts. If information is communicated this makes perfect sense, however, less so when consumable resources like Tryptophan molecules are communicated along couplings. To realize the Tryptophan Synthase and Tryptophan Operon model (Degenring et al. 2004, Degenring et al. 2005) a stochastic multi-coupling was introduced to facilitate this form of interaction. It constitutes a form of variable coupling that given a set of models realizes at the moment of output a coupling to only one of them. Via multi-couplings a trp is sent to a Repressor, a serine to an Enzyme, or a ribosome to a mRNA (Figure 9).

The traditional couplings as they are supported in Devs are of particular benefit if covalent structures like the Tryptophan Synthase enzyme shall be modeled. Here the alpha and beta subunit form a covalent structure and directly interact. Whereas the beta unit signalizes the availability of serine to alpha, the alpha subunit tunnels the produced Indole via the tunnel to beta. Not only this static channeling but also dynamic channeling, e.g. observed in the glycolysis, require a direct communication. This form of interaction is nicely reflected in Devs.

Covalent structures like the enzyme responsible for the Tryptophan production, show another important aspect: the ability to form more complex models by grouping models into containers. This is one of the strength of Devs, which has been adapted by many continuous and discrete modeling formalism. In π-Calculus hierarchies are modeled by relying on the notion of scope of names introduced through the new operator. Like in blocks of programming languages, we can define names into nested structure that implicitly represent hierarchies. Extensions like Ambients and particularly Bio-Ambients, and Beta Binders make these implicit composition hierarchies explicit (Cardelli and Gordon 2000, Regev et al. 2004) and support a grouping of processes within locations.

5.5 Execution

For both stochastic π-Calculus and Devs simulation engines, e.g. (Himmelspach and Uhrmacher 2004, Regev 2001), and abstract simulator specifications exist, e.g. (Phillips and Cardelli 2004, Zeigler et al. 2000). Both formalisms belong into the family of discrete-event formalisms, and thus can be interpreted by a discrete-event simulation engine. Devs simulators are typically general discrete-event simulation engines. Simulation engines for Stochastic π, e.g. SPIM (Phillips and Cardelli 2004), BioSpi (Regev 2001), are aimed at cell biological applications. For scheduling the next event they take the propensities of reactions according to Gillespie into account. They interpret the channels as reactions, and processes as possible reactands. To determine the time of next event and which reaction to execute at this time, the simulator calculates the propensities based on the number of reactands for each reaction and the mesoscopic reaction rate. Given a reaction to execute, it selects randomly among the available reactands. In the Devs model which is executed by a general discrete

event simulator, the cytoplasma keeps track of the number of models, i.e. reactands, and describes the collision probability explicitly. In realizing the random selection of possible reactands stochastic multi-couplings are used. Thus, additional effort in modeling is required, however the simulation engine supports a higher flexibility in describing biological systems.

6 CONCLUSION

Both modeling formalism support different perspectives on the system to be modeled. The strength of `Devs` lies in its modular, hierarchical design. In combination with the support of variable structures and multicouplings it provides a basis for designing complex cellular models, whose individual entities are easy to understand, if visualized e.g. by a `StateChart` variant, which given the closeness of both formalisms should be straightforward. However, keeping all information about state changes in the inside of the model puts unnecessary overhead on modeling and simulation. Enriching the interface, e.g. by a dynamic generation of typed ports would be a possibility to address this deficiency. In addition more flexible communication patterns than the usual couplings have to be supported, one step into this direction are the multicouplings already introduced.

The `Stochastic` π is a linguistic framework to model the dynamic behavior of complex interacting systems. It is based on the notion of names used to represent both communication channels and data. This property allows to make the interconnection topology of the interacting processes vary over time mimicking a notion of mobility. The formalism include the notion of rate associated with transitions that allow to build stochastic models on which simulation can be carried out. The main advantage of the formalism that has been proved useful in modeling biological systems, e.g. (Lecca et al. 2004, Kuttler 2005), are its simple structure coupled with a strong theoretical foundation and the notion of compositionality. Current efforts are directed to provide additional structure in modeling, see e.g. `BioAmbients` and `Beta Binders`.

REFERENCES

Barros, F. 1997. Modeling formalisms for dynamic structure systems. *ACM Trans. Model. Comput. Simul.* 7 (4): 501–515.

Bobeanu, M., E. Kerckhoffs, and H. Van Landeghem. 2004. Modeling of discrete event systems: A holistic and incremental approach using petri nets. *ACM Trans. Model. Comput. Simul.* 14 (4): 389–423.

Borland, S., and H. Vangheluwe. 2003. Transforming statecharts to DEVS. In *Summer Computer Simulation Conference. Student Workshop*, ed. A. Bruzzone and M. Itmi, 154–159: SCS.

Cardelli, L., and A. Gordon. 2000. Mobile ambients. *Theor. Comput. Sci.* 240 (1): 177–213.

Cowan, R. 2003. Stochastic models for DNA replication. In *Stochastic Processes*, ed. D. Shanbhag and C. Rao, Handbook of Statistics, Chapter 4.

de Jong, H. 2002. Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. *Journal of Computational Biology* 9 (1): 67–103.

Degenring, D., J. Lemcke, M. Röhl, and A. Uhrmacher. 2005. A variable structure model - the tryptophan operon. In *CMSB*, ed. G. Plotkin.

Degenring, D., M. Röhl, and A. Uhrmacher. 2004. Discrete event, multi-level simulation of metabolite channeling. *BioSystems* 75 (1-3): 29–41.

Efroni, S., D. Harel, and I. Cohen. 2003. Towards rigorous comprehension of biological complexity: Modeling, execution and visualization of thymic t cell maturation. *Genome Research* (13): 2485–2497.

Efroni, S., D. Harel, and I. Cohen. 2005. Reactive animation: Realistic modeling of complex dynamic systems. *Computer IEEE*.

Gibson, M. A., and J. Bruck. 2000. EfficientExact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. *Journal of Physical Chemistry A* 104 (9): 1876–1889.

Gillespie, D. T. 1977. Exact Stochastic Simulation of Coupled Chemical Reactions. *The Journal of Physical Chemistry B* 81 (25): 2340–2361.

Haas, P. 2002. *Stochastic petri nets: Modelling, stability, simulation*. Springer.

Himmelspach, J., and A. Uhrmacher. 2004. A component-based simulation layer for JAMES. In *Proc. of the 18th Workshop on Parallel and Distributed Simulation (PADS), May 16-19, 2004, Kufstein, Austria*, 115–122.

Kuo, D., and J. D. Keasling. 1996. A Monte Carlo simulation of plasmid replication during the bacterial division cycle. *Biotechnology and Bioengineering* 52 (6): 633–647.

Kuttler, C. 2005. Bacterial transcription in the pi calculus. In *CMSB*, ed. G. Plotkin.

Kuttler, C. to appear. The role of the attenuation of the tryptophan operon in the pi calculus. in preparation.

Lecca, P., C. Priami, P. Quaglia, B. Rossi, C. Laudanna, and G. Constantin. 2004. Language Modelling and Simulation of Autoreactive Lymphocytes Recruitment in Inflamed Brain Vessels. *SCS Simulation* 80:273–288.

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

Phillips, A., and L. Cardelli. 2004. A correct abstract machine for the stochastic pi-calculus. In *BioConcur 2004*: Electronic Notes in Theoretical Computer Science.

Priami, C. 1995. Stochastic π-calculus. *The Computer Journal*:578–589.

Priami, C., and P. Quaglia. 2005. Beta binders for biological interactions. *Transactions on Computational Systems*

Biology.

Ramsey, S., D. Orell, and H. Bolouri. 2005. Dizzy: Stochastic simulation of large scale genetic regulatory networks. *Journal of Bioinformatics and Computational Biology*.

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

Regev, A., E. Panina, W. Silverman, L. Cardelli, and E. Shapiro. 2004. Bioambients: An abstraction for biological compartments. *Theor. Computer Science*.

Santill'an, M., and M. C. Mackey. 2001. Dynamic regulation of the tryptophan operon: A modeling study and comparison with experimental data. *Proc. of the Nat. Acad. of Sciences of the USA* 98 (4): 1364–1369.

Uhrmacher, A. 2001. Dynamic structures in modeling and simulation - a reflective approach. *ACM Transactions on Modeling and Simulation* 11 (2): 206–232.

Wolkenhauer, O. 2001. Systems biology: the reincarnation of systems theory applied in the biology? *Briefings in Bioinformatics* 2 (3): 258–270.

Zeigler, B. 1986. Toward A Simulation Methodology for Variable Structure Modeling. In *Modelling and Simulation Methodology in the Artificial Intelligence Era*, 195–210. Amsterdam: North Holland.

Zeigler, B., H. Praehofer, and T. Kim. 2000. *Theory of Modeling and Simulation*. London: Academic Press.