The Attributed Pi Calculus

Mathias John[1], Cédric Lhoussaine[2,4],
Joachim Niehren[3,4], and Adelinde M. Uhrmacher[1]

[1] University of Rostock
Institute of Computer Science, Modeling and Simulation Group
[2] University of Lille 1, LIFL, CNRS UMR8022
[3] INRIA, Lille, Mostrare project
[4] BioComputing project, LIFL, Lille

Abstract. The attributed pi calculus ($\pi(\mathcal{L})$) forms an extension of the pi calculus with attributed processes and attribute dependent synchronization. To ensure flexibility, the calculus is parametrized with the language $\mathcal{L}$ which defines possible values of attributes. $\pi(\mathcal{L})$ can express polyadic synchronization as in pi@ and thus diverse compartment organizations. A non-deterministic and a stochastic semantics, where rates may depend on attribute values, is introduced. The stochastic semantics is based on continuous time Markov chains. A simulation algorithm is developed which is firmly rooted in this stochastic semantics. Two examples underline the applicability of $\pi(\mathcal{L})$ to systems biology: Euglena's movement in phototaxis, and cooperative protein binding in gene regulation of bacteriophage lambda.

1 Introduction

A plethora of formal concurrent modeling languages has been proposed for systems biology since the seminal work of Regev and Shapiro [2,1] on the π-calculus. These languages subscribe to two main paradigms: In the object-centered paradigm, interaction capacities are attached to concurrent actors, as for instance in the π-calculus [7,6,5,4,3]. Rule-based languages focus on chemical reactions and pathways being composed thereof [8,10,9]. Deterministic models in terms of differential equations can be obtained by averaging over possible behaviors of non-deterministic models in concurrent languages [11,12,8].

In the following, we introduce the attributed π-calculus, an extension of the π-calculus by attributed processes and attribute dependent synchronization. Attribute values are useful in order to define diverse properties of biological processes. Two examples on different levels of abstraction illustrate the basic idea: First, a cell `Cell(coord,vol)` is attributed by coordinates `coord` $\in \mathbb{R}^3$ and a volume `vol` $\in \mathbb{R}^+$. Second, protein `Prot(comp)` is attributed by the cellular compartment `comp` $\in \{$`'nucleus'`$, \ldots\}$ in which it is located.

Whether two processes of the attributed π-calculus are allowed to interact depends on constraints on their attribute values. Consider e.g. the binding action of a protein `Prot(x)` of type `x` $\in \{$`'A'`, `'B'`$\}$ to an operator `Op(y)` able to bind

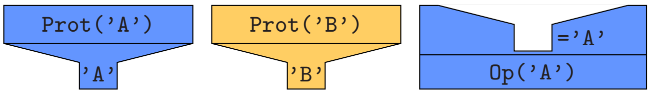

Fig. 1. Only the protein of type `'A'`is permitted to bind to the operator of type `'A'`, but not the protein of type `'B'`. Type equality is tested by the operator, once it sees the type of the protein, by applying the test function λ`x.x='A'`.

proteins of type `y` $\in$ {`'A'`, `'B'`}, as for instance in the system in Fig. 1. For the binding of a protein of type `x` to an operator of type `y`, equality of their types `x=y` is required. In the attributed π-calculus this can be expressed by the following definitions, which are valid for all possible values of the attributes `x` and `y`:

$$\mathsf{Prot}(x) \triangleq \mathsf{bind}[x]!().0$$
$$\mathsf{Op}(y) \triangleq \mathsf{bind}[\lambda x.x{=}y]?().\mathsf{OpBound}(y)$$

This says, that before enabling a binding action, `Prot(x)` needs to provide its type, which is the value of attribute `x`, as specified by `bind[x]`. The operator receives the value of `x` and tests it for equality with its own type, the value of attribute `y`, as specified by `bind[`λ`x.x=y]`; the equality constraint is expressed by the Boolean valued function λ`x.x=y`. Consider for instance a system with two actors `Prot('A')` and `Op('A')`:

$$\mathsf{Prot}('A') \mid \mathsf{Op}('A') \rightarrow \mathsf{OpBound}('A')$$

The interaction constraint for these two actors is composed by function application `(`λ`x.x='A')'A'`. It evaluates to the truth value of `'A'='A'` which is true. This enables the above binding action, by which the operator turns into its bound state. The slightly more complex system `Prot('A') | Prot('B') | Op('A')` where only the first protein is permitted to bind to the operator is illustrated in Fig. 1. Assuming constraints that evaluate to positive real numbers, e.g. stochastic rates not only Booleans, also specific affinities of interaction partners can be expressed.

Attribute values are defined in a language $\mathcal{L}$. They subsume possible reaction rates and constraints as in higher order logic. The language $\mathcal{L}$ forms a parameter of the attributed π-calculus, which we call therefore $\pi(\mathcal{L})$, similarly to constraint logic programming CLP($\mathcal{L}$) [13] or concurrent constraint programming cc($\mathcal{L}$) [14]. This avoids inventing completely independent calculi for the many reasonable choices of attribute values and constraints. E.g. encoding the π-calculus with polyadic synchronization [15] requires a constraint language $\mathcal{L}_{\wedge,=}$ with equality tests for channel names and conjunctions thereof.

The paper is structured as follows. We start with the syntax of $\pi(\mathcal{L})$. Thereafter, we present a non-deterministic and a stochastic semantics for $\pi(\mathcal{L})$, independently a concrete choice of attribute language. Two examples provide deeper insight into the modeling with $\pi(\mathcal{L})$. The first one describes the light dependent movement of Euglena, a single celled organism living in inland water. The second example regards cooperative binding, a phenomenon, which is often observed in gene regulatory systems [17,16]. Compared to previous variants of the

π-calculus, the ability to make reaction rates dependent on the attribute values of interaction partners facilitates modeling for both examples.

Afterward, we present a stochastic simulator for $\pi(\mathcal{L})$ independently of the attribute language. In contrast to previous simulators for the stochastic π-calculus [5,4,3], we show how to infer the simulator directly from the definition of the stochastic semantics in terms of continuous time Markov chains.

We complete the paper with encoding π@ [15] in $\pi(\mathcal{L}_{\wedge,=})$ with respect to the non-deterministic semantics[1]. The language π@ was proposed as a unifying extension of the π-calculus, in which to express different compartment organizations as in BioAmbients [20] or Brane Calculi [21]. By encoding π@ in $\pi(\mathcal{L}_{\wedge,=})$, this feature can be inherited, while supporting richer classes of constraints and values.

2 Extending the π-Calculus

We present the attributed π-calculus $\pi(\mathcal{L})$ which extends the π-calculus with respect to some attribute language $\mathcal{L}$. This sublanguage provides expressions for describing values of process attributes. Values subsume constraints, for specifying communication abilities, and stochastic rates.

Keeping the attribute language $\mathcal{L}$ parametric avoids us reinventing independent calculi for the many useful choices in practice. We present two semantics for $\pi(\mathcal{L})$, a non-deterministic semantics and a stochastic semantics, such that the latter refines the former (for error-free programs). Both semantics are defined independently of the concrete choice of $\mathcal{L}$. The stochastic semantics leads to a stochastic simulator which also is independent of the concrete choice of $\mathcal{L}$, see Section 4. Fixing the attribute language is an orthogonal issue, which is easy in practice, for instance by using the implementation language of the simulator.

2.1 Languages of Attribute Values

Let $\mathbb{B} = \{0, 1\}$ be the set of Booleans, $\mathbb{N}$ the set of natural numbers starting from 1, and $\mathbb{R}^+$ the set of non-negative real numbers.

An attribute language is a functional programing language that provides expressions by which to compute values. Expressions are built from constants for numbers, functions, relations, such as 0, +, and $\geq$, and channel names, such as x and y. Whenever there exists any ambiguity in concrete syntax, we write constants with quotes such as `'val'` in order to distinguish them from channel names without quotes such as `val`.

Besides defining communication abilities, channel names behave like variables, whose values are defined by the environment. The value of a channel name x can be accessed by applying the function constant `'val'` to x. Constraints can be understood as expressions of type Boolean, as for instance (`'val'` x)+(`'val'` y) ≥ 0.

[1] It remains unclear how to add reaction rates to π@ in a general manner as needed for defining stochastic semantics. This has only be done for particular adaptations of π@ [18,19].

A constraint is successful in environments in which it evaluates to the successful Boolean value 1. In the stochastic setting, successful values are relaxed to stochastic rates in $\mathbb{R}^+ \cup \{\infty\}$.

More formally, we start from an infinite set *Chans*, whose elements are called channel names and ranged over by x, y. An attribute language over *Chans* is a triple $\mathcal{L} = (Consts, Succ, \Downarrow)$. The first component is a finite set *Consts*, whose elements are called constants and ranged over by c. The set *Exprs* of $\mathcal{L}$ is defined as the set of all λ expressions with constants in *Consts* and variables in *Chans*. The set *Vals* of $\mathcal{L}$ is defined as the set of all values, i.e. channel names, constants, and λ abstractions.

$$\begin{aligned} e \in Exprs &::= x \mid c \mid \lambda x.e \mid e_1 e_2 \\ v \in Vals &::= x \mid c \mid \lambda x.e \end{aligned}$$

The second component of $\mathcal{L}$ is a finite set $Succ \subseteq Vals$, whose elements are called successful values and ranged over by r. In examples, *Succ* will cover the Boolean 1 only or all stochastic rates $r \in \mathbb{R}^+ \cup \{\infty\}$.

The third component of $\mathcal{L}$ is a big-step evaluator $\Downarrow$, a black box algorithm, which evaluates expressions to values. Its behavior depends on environments $\rho : Chans \rightarrow Vals$ mapping channel names to values (including stochastic rates). Let *Env* be the set of all such environments. The big-step evaluator is a partial function of type: $\Downarrow : Env \times Exprs \rightharpoonup Vals$. Instead of $\Downarrow(\rho, e) = v$ we write $\rho \vdash e \Downarrow v$ and call v the value of e wrt. ρ. If the special constant `'val'` belongs to *Consts* then we assume that it satisfies for all channel names $x \in Chans$:

$$\rho \vdash \texttt{'val'}\ x \Downarrow \rho(x)$$

When considering stochastic semantics, we have to choose an attribute language in which *Succ* is a subset of stochastic rates, i.e., $Succ \subseteq \mathbb{R}^+ \cup \{\infty\}$.

Values of $\Downarrow(\rho, e)$ may be undefined for two possible reasons: program errors or non-termination. Typical examples for program errors are type errors, e.g. applying constants to real numbers instead of functions. Non-termination is notorious, when the big-step semantics is defined by a small-step evaluator, as usual. We treat non-termination of expressions in processes as program errors. In this case execution blocks, since it is running into an unproductive infinite loop. The assumption of termination is essential for stochastic simulation, where we have to evaluate all expressions in redexes (i.e. pairs of sender and receiver on the same channel) before performing any communication step. One could try to exclude non-termination statically by imposing a simple type system on the level of processes. Instead, we prefer to treat attempts to access undefined values $\Downarrow(\rho, e)$ as program errors dynamically.

An example for an attribute language is $\mathcal{L}_{\wedge,=}$, which is used in Section 5 to encode the π-calculus with polyadic synchronization and a non-deterministic semantics. It provides constraints by which to test a conjunction of equalities between channel names $\wedge_{i=1}^{n} x_i{=}y_i$. We use the following set of constants: $Consts = \mathbb{B} \cup \{\texttt{'and'}, \texttt{'equal'}, 0, 1\}$ of which there is a single successful value in $Succ = \{1\}$. The types of the function constants are $\texttt{'and'} : \mathbb{B} \rightarrow \mathbb{B} \rightarrow \mathbb{B}$

Processes	$P ::= A(\tilde{e})$	defined process
	$\mid\ P_1 \mid P_2$	parallel composition
	$\mid\ (\nu x{:}v)\ P$	channel creation
	$\mid\ \pi_1.P_1 + \ldots + \pi_n.P_n$	summation of alternative choices
	$\mid\ \mathbf{0}$	empty solution
Prefixes	$\pi ::= v[e]?\tilde{y}$	receiver
	$\mid\ v[e]!\tilde{v}$	sender
Definitions	$D ::= A(\tilde{x}) \triangleq P$	parametric process definition

Fig. 2. Syntax of $\pi(\mathcal{L})$ where $v, \tilde{v} \in \mathit{Vals}$, $x, \tilde{y} \in \mathit{Chans}$, and $e \in \mathit{Exprs}$

$$\begin{array}{ll}
fn(\pi_1.P_1 + \ldots + \pi_n.P_n) = \bigcup_{i \in \{1,\ldots,n\}} fn(\pi_i.P_i) & fn(\mathbf{0}) \ = \ \emptyset \\
fn(v[e]?\tilde{y}.P) = fn(v) \cup fn(e) \cup (fn(P) \setminus \{\tilde{y}\}) & fn(P_1 \mid P_2) = fn(P_1) \cup fn(P_2) \\
fn(v[e]!\tilde{v}.P) = fn(v) \cup fn(\tilde{v}) \cup fn(e) \cup fn(P) & fn(A(\tilde{e})) = fn(e) \\
fn((\nu x{:}v)\ P) = (fn(P) \cup fn(v)) \setminus \{x\} & fn(A(\tilde{x}) \triangleq P) = fn(P) \setminus \{x\}
\end{array}$$

Fig. 3. Free channel names

and `'equal'` : $\mathit{Chans} \to \mathit{Chans} \to \mathbb{B}$. Therefore, the evaluator is defined for all $x, y \in \mathit{Chans}$ and $c_1, c_2 \in \mathbb{B}$ and undefined for all other values:

$$\rho \vdash \texttt{'equal'}\ x\ y \Downarrow \begin{cases} 1 \text{ if } x = y \\ 0 \text{ otherwise} \end{cases} \qquad \rho \vdash \texttt{'and'}\ c_1\ c_2 \Downarrow \begin{cases} 1 \text{ if } c_1 = c_2 = 1 \\ 0 \text{ otherwise} \end{cases}$$

2.2 Syntax of $\pi(\mathcal{L})$

Let $\mathcal{L}$ be an attribute language over some infinite set of channel names $x \in \mathit{Chans}$. The vocabulary for building attributed processes in $\pi(\mathcal{L})$ consists of an infinite set of process names $A \in \mathit{Proc}$, each of which comes with a fixed arity in $\mathbb{N} \cup \{0\}$, and the set of expressions $e \in \mathit{Exprs}$ and values $v \in \mathit{Vals}$ of $\mathcal{L}$.

We use tuple notation in many places. We write $\tilde{e}$ for tuples of expressions, $\tilde{v}$ for tuples of values, and $\tilde{x}$ for tuples of channel names. Substitutions $[\tilde{v}/\tilde{x}]$ apply to tuples of the same length. The same holds for sequences of expressions and values in tuple evaluation $\rho \vdash \tilde{e} \Downarrow \tilde{v}$.

The syntax of the attributed π-calculus $\pi(\mathcal{L})$ is defined in Fig. 2. There are three syntactic categories, (attributed) processes P, prefixes π, and definitions D. The expressions of these categories are as usual, except that we permit λ expressions e in places, where previously, only channel names x or rates r could be found.

A defined process is a term $A(\tilde{e})$ where the arity of tuple $\tilde{e}$ is equal to the arity of A.

Channel creation $(\nu x{:}v)\ P$ creates a new channel name x and assigns a value v to it. The scope of x ranges over v and P. The assignment of v to x is stored in the current environment ρ, such that it can be accessed later on by using the function constant `'val'`. This mechanism is useful in order to assign stochastic rates to channels, as in the usual stochastic π-calculus. More generally, it makes channel names behave like variables or memory blocks which refer to values.

Prefixes $v[e_1]!\tilde{v}$ for receiving inputs and $v[e_1]?\tilde{x}$ for sending outputs over channels are generalized in two aspects. First of all, we add expressions $[e_1]$ and $[e_2]$ to senders resp. receivers, which impose constraints on attribute values to be satisfied before communication. Second, we permit values in non-binding positions, where usually only channel names can be used. This way, arbitrary values can be send and received over channels. Prefixes in which v is not a channel name are considered to be erroneous. Such errors are difficult to exclude statically since channel names can be substituted by values dynamically. Even more permissive would be to allow expressions $\tilde{e}$ instead of values $\tilde{v}$. This does not raise any problems as long as they are evaluated before communication. For sake of simplicity, we stick to the slightly more restrictive setting.

The free channel names $\mathit{fn}(P)$ are defined as usual in Fig. 3. They account for free channel names in lambda expressions e denoted by $\mathit{fn}(e)$, i.e., those occurring out of the scope of all λ binders in e. Bound channel names $\mathit{bn}(P)$ are defined as before, except that λ-binders in expressions $e \in \mathit{Exprs}$ are included too. The structural congruence on processes $\equiv$ remains the least congruence satisfying α conversion, the axioms of commutative monoids w.r.t. $(|, \mathbf{0})$, associativity and commutativity of summation $+$, and the usual scoping rules of ν-binders:

$$\begin{array}{ll} (\nu x{:}v)\ (P_1 \mid P_2) \equiv (\nu x{:}v)\ P_1 \mid P_2 & \text{if } x \notin \mathit{fn}(P_2) \\ (\nu x{:}v)\ (\nu y{:}u)\ P \equiv (\nu y{:}u)\ (\nu x{:}v)\ P & \text{if } x \notin \mathit{fn}(u) \text{ and } y \notin \mathit{fn}(v) \end{array}$$

2.3 Non-deterministic Semantics

The non-deterministic operational semantics of the attributed π-calculus is given in Fig. 4. It is defined in terms of a small step reduction relation between processes w.r.t. to an environment $\rho : \mathit{Chans} \to \mathit{Vals}$:

$$\rho \vdash P \to P'$$

Rule ($\textsc{def}_{nd}$) applies the definition of a process in a call-by-value manner once all parameters have been evaluated. This will always succeed for error free programs, but may run into an infinite loop or raise an exception otherwise. The communication rule ($\textsc{com}_{nd}$) applies to two parallel sums, which contain matching communication parts, i.e. a receiver $x[e_1]?\tilde{y}.P_1$ and a sender on the same channel $x[e_2]!\tilde{v}.P_2$ such that the constraint $e_1 e_2$ evaluates to some successful value. Reduction cancels all other alternatives of the communicating sums. Furthermore, it substitutes $\tilde{y}$ by $\tilde{v}$ in the continuation P_1 of the receiver, and keeps the continuation P_2 of the sender. The structural rule ($\textsc{new}_{nd}$) eliminates a binder $(\nu x{:}v)\ P$ and updates the value of x in the environment to v.

The usual π-calculus as used by BioSpi or Spim [4,3] (without stochastics) can be encoded in $\pi(\mathcal{L}_1)$ with the successful Boolean value as unique constant, i.e., $\mathit{Consts} = \mathit{Succ} = \{1\}$. The translation introduces dummy constraints that are

Application of definitions

$$(\text{DEF}_{nd})\ \frac{\rho \vdash \tilde{e} \Downarrow \tilde{v} \qquad A(\tilde{x}) \triangleq P}{\rho \vdash A(\tilde{e}) \to P[\tilde{v}/\tilde{x}]}$$

Communication

$$(\text{COM}_{nd})\ \frac{\rho \vdash e_1 e_2 \Downarrow r \in \mathit{Succ}}{\rho \vdash x[e_1]?\tilde{y}.P_1 + \ldots \mid x[e_2]!\tilde{v}.P_2 + \ldots \to P_1[\tilde{v}/\tilde{y}] \mid P_2}$$

Structural rules

$$(\text{PAR}_{nd})\ \frac{\rho \vdash P_1 \to P_1'}{\rho \vdash P_1 \mid P_2 \to P_1' \mid P_2} \qquad (\text{NEW}_{nd})\ \frac{\rho \cup \{x : v\} \vdash P \to P'}{\rho \vdash (\nu x{:}v)\ P \to (\nu x{:}v)\ P'}$$

$$(\text{STRUC}_{nd})\ \frac{P \equiv P_1 \qquad \rho \vdash P_1 \to P_2 \qquad P_2 \equiv P'}{\rho \vdash P \to P'}$$

Fig. 4. Non-deterministic semantics of $\pi(\mathcal{L})$

Solutions	$S ::=$	$A(\tilde{e})$	defined molecule
	$\mid$	$S_1 \mid S_2$	parallel composition
	$\mid$	$(\nu x{:}v)\ S$	channel creation
	$\mid$	$\mathbf{0}$	empty solution
Molecules	$M ::=$	$\pi_1.S_1 + \ldots + \pi_n.S_n$	sum of alternative choices
	$\mid$	$(\nu x{:}v)\ M$	channel creation
Prefixes	$\pi ::=$	$v[e]?\tilde{y}$	receiver
	$\mid$	$v[e]!\tilde{v}$	sender
Definitions	$D ::=$	$A(\tilde{x}) \triangleq M$	molecule definition

Fig. 5. Biochemical forms where $v, \tilde{v} \in \mathit{Vals}$, $x, \tilde{y} \in \mathit{Chans}$, and $e \in \mathit{Exprs}$

always true. In order to do so, let $\_$ be an arbitrary channel. We rewrite all receivers $x!\tilde{y}.P$ to $x[\_]!\tilde{y}.P$ and all senders $x?\tilde{y}.P$ to $x[\lambda\_.1]?\tilde{y}.P$. Channel declarations $(\nu x)P$ are translated to $(\nu x{:}\_)\ P$, i.e.; they introduce dummy values too.

2.4 Biochemical Forms

Every process of $\pi(\mathcal{L})$ can be normalized into biochemical forms [4]. These forms are well-suited for defining the stochastic semantics and implementing simulators.

The idea is to introduce parametric process definitions for all sums in processes. What remains is a parallel composition of defined processes possibly with some local channel names. A systematic transformation replaces all nested sums by newly defined processes. The parameters of the new process definitions are the free names of the nested sums, while the sums itself are moved into the new process definitions.

Biochemical forms as defined in Fig. 5 have four categories of terms. The previous category of processes is split into two levels, *solutions* ranged over by S and *molecules* ranged over by M. Solutions can be identified with multisets of defined molecules $A(\tilde{e})$ up to some local channels. Molecule definitions are sums possibly in the scope of ν-binders. The alternatives of the sums are prefixed solutions, so that no nested sums can be produced by communication.

2.5 Stochastic Semantics

The stochastic semantics of $\pi(\mathcal{L})$ is given in Fig. 6. It defines Markov chains for solutions of $\pi(\mathcal{L})$. The states of such Markov chains are the congruence classes of solutions with respect to structural congruence $\equiv$. The stochastic rates, associated to state transitions, are obtained by evaluating constraints to successful values. Rather than saying "that" a constraint is successful, this expresses "how" successful it is. We use product notation for parallel compositions, by writing $\prod_{i=1}^{n} S_i$ instead of $S_1 \mid \ldots \mid S_n$. We use meta variables N to range over quantifier prefixes $(\nu\widetilde{y{:}v})$, write $N(S)$ instead of $(\nu\widetilde{y{:}v})\ (S)$, and $N(M)$ instead of $(\nu\widetilde{y{:}v})\ (M)$. We are looking for a stochastic semantics expressing the *Chemical Law of Mass Action* according to which the speed of a chemical reaction in a solution is proportional to the number of possible interactions of its reactants in the solution. This can be translated in $\pi(\mathcal{L})$ as follows: given a source state S, the semantics defines how many distinct interactions allow a transition to a common (i.e. structurally equivalent) target state S'. For instance, suppose that $S = A \mid A \mid B$, with definitions associated to A and B such that $A \mid B \to C$ with rate r, then one expects $S \to A \mid C$ with rate $2r$. Indeed, two distinct interactions lead to $A \mid C$ involving either the first or the second occurrence of A. To this end, we use an intermediate reduction relation $\xrightarrow[\ell]{r}$ using labels $\ell \in \mathbb{N}^4$ and rates $r \in \mathbb{R}^+ \cup \{\infty\}$, from which we build a Markov chain $\xrightarrow{r}$ by summation over labels.

A label locates a potential redex in solutions. Pairs $(i_1, j_1) \in \mathbb{N}^2$ mark the position of the j_1'th alternative in the i_1'th molecule of a solution. Labels $\ell = (i_1, j_1, i_2, j_2) \in \mathbb{N}^4$ fix a pair of alternatives in a solution S uniquely up to α-renaming. Such a pair is a redex, if it is a receiver-sender-pair on the same channel from different defined molecules. The set $\text{redex}_\ell^\rho(S)$ defined by rule (REDEX) contains all the α-variants of the redex at ℓ, if there is any. The environment ρ matters here, since all arguments of the defined molecules with numbers i_1 and i_2 need to be evaluated. The redex contains the j_1'th alternative of molecule i_1 and the j_2'th alternative of molecule i_2 as selected by rule (CHOOSE). Only alpha congruence $\equiv_\alpha$ is permitted here. Using full structural congruence instead would spoil the meaning of labels, so that redexes could not be uniquely identified by them. This would falsify summation in rule (SUM) and counting in rule (COUNT).

Going back to our previous example, and assuming that A and B are defined with a single alternative consisting of complementary prefixes, we have $A \mid A \mid B \xrightarrow[(1,1,3,1)]{r} A \mid B$ and $A \mid A \mid B \xrightarrow[(2,1,3,1)]{r} A \mid B$.

Redexes $(1 \leq j \leq m,\ i_1, j_1, i_2, j_2 \in \mathbb{N})$

$$\text{(CHOOSE)}\ \frac{\rho \vdash \tilde{e} \Downarrow \tilde{v} \qquad A(\tilde{x}) \triangleq N(\pi_1.S_1 + \ldots + \pi_m.S_m)}{\text{choose}^{\rho}_{j}(A(\tilde{e})) = (N(\pi_j.S_j))[\tilde{v}/\tilde{x}]}$$

$$\text{(REDEX)}\ \frac{\begin{array}{l}\text{choose}^{\rho}_{j_1}(A_{i_1}(\tilde{e}_{i_1})) \equiv_\alpha (\nu \widetilde{y_1{:}v_1})\ (x[e'_1]?\tilde{y}.S_1) = S'_1 \\ \text{choose}^{\rho}_{j_2}(A_{i_2}(\tilde{e}_{i_2})) \equiv_\alpha (\nu \widetilde{y_2{:}v_2})\ (x[e'_2]!\tilde{v}.S_2) = S'_2 \qquad i_1 \neq i_2\end{array}}{(S'_1, S'_2) \in \text{redex}^{\rho}_{(i_1,j_1,i_2,j_2)}(\prod_{i=1}^{n} A_i(\tilde{e}_i))}$$

where $x \in \mathit{Chans}$, $x \notin \{\tilde{y_1}\} \cup \{\tilde{y_2}\}$ and $\{\tilde{y_1}\} \cap \{\tilde{y_2}\} = \emptyset$.

Labeled reduction $(r \in \mathbb{R}^\infty$ and $\ell = (i_1, j_1, i_2, j_2) \in \mathbb{N}^4)$

$$\text{(COM)}\ \frac{\begin{array}{l}(N_1(x[e'_1]?\tilde{y}.S_1), N_2(x[e'_2]!\tilde{v}.S_2)) \in \text{redex}^{\rho}_{\ell}(\prod_{i=1}^{n} A_i(\tilde{e}_i)) \\ \rho \vdash e'_1 e'_2 \Downarrow r \in \mathit{Succ}\end{array}}{\rho \vdash \prod_{i=1}^{n} A_i(\tilde{e}_i) \ \underset{\ell}{\xrightarrow{r}}\ \prod_{i=1, i \neq i_1, i_2}^{n} A_i(\tilde{e}_i) \mid N_1 N_2 (S_1[\tilde{v}/\tilde{y}] \mid S_2)}$$

$$\text{(NEW)}\ \frac{\rho \circ [v/x] \vdash S \underset{\ell}{\xrightarrow{r}} S'}{\rho \vdash (\nu x{:}v)\ S \underset{\ell}{\xrightarrow{r}} (\nu x{:}v)\ S'} \text{ where } x \notin \mathit{dom}(\rho).$$

Markov chain $(r, r' \in \mathbb{R}^+)$

$$\text{(CONV)}\ \frac{\forall \ell {\in} \mathbb{N}^4 \forall (N_1(x[e'_1]?\tilde{y}.S_1), N_2(x[e'_2]!\tilde{v}.S_2)) \in \text{redex}^{\rho}_{\ell}(S) \exists v {\in} \mathit{Vals} : \rho \vdash e'_1 e'_2 \Downarrow v}{\rho \vdash S \Downarrow}$$

$$\text{(SUM)}\ \frac{\rho \vdash S \Downarrow \quad S \equiv S_1 \quad \sum_{\{\ell \mid \rho \vdash S_1 \underset{\ell}{\xrightarrow{r'}} S_2 \equiv S'\}} r' = r \neq 0 \quad \nexists \ell. \rho \vdash S_1 \underset{\ell}{\xrightarrow{\infty}}}{\rho \vdash S \xrightarrow{r} S'}$$

$$\text{(COUNT)}\ \frac{\begin{array}{lll} & & n = \sharp\{\ell \mid \rho \vdash S_1 \underset{\ell}{\xrightarrow{\infty}} S_2 \equiv S'\} \neq 0 \\ \rho \vdash S \Downarrow & S \equiv S_1 & m = \sharp\{\ell \mid \rho \vdash S_1 \underset{\ell}{\xrightarrow{\infty}} S_2\}\end{array}}{\rho \vdash S \xrightarrow{\infty(n/m)} S'}$$

Fig. 6. Stochastic operational semantics of $\pi(\mathcal{L})$

Rule (COM) performs the interactions of the redex with label $\ell = (i_1, j_1, i_2, j_2)$, under the condition that the constraint of the redex is successful. Rule (NEW) states that reduction may occur under channel restriction with corresponding environment updating. We assume that restricted channels do not occur in the environment.

Markov chains are derived by summing up the rates of all labeled reductions leading from S to S' modulo structural congruence. Summation in rule (SUM) requires that all constraints in all redexes of S converge to some value, as defined by rule (CONV). It equally presupposes that no immediate reaction with rate ∞

is enabled. Rule (COUNT) does the same except that it assigns to each transition a probability. This probability represents the number of immediate located transitions leading to a common state over the total number of outgoing immediate transitions. Thanks to those probabilities, immediate transitions can be eliminated in order to obtain a true Continuous Time Markov chain preserving *probability transitions* and *sojourn times* (see [5] for details).

The usual stochastic π-calculus can be encoded in $\pi(\mathcal{L}_{\mathbb{R}^+\cup\{\infty\}})$ with $Consts = Succ = \mathbb{R}^+ \cup \{\infty\}$. The translation only needs to access the rate of the channel in the environment. One way to do so is to rewrite sender $x!\tilde{y}.P$ to $x[\texttt{'val'}\,x]!\tilde{y}.P$ and receiver $x?\tilde{y}.P$ to $x[\lambda r.r]?\tilde{y}.P$. Channel declarations with stochastic rates $(\nu x{:}r)\ P$ remain unchanged. Another encoding would be to rewrite output sender $x!\tilde{y}.P$ to $x[\_]!\tilde{y}.P$ and input receiver $x?\tilde{y}.P$ to $x[\lambda\_.\texttt{'val'}\ x]?\tilde{y}.P$. The only difference is whether sender or receiver accesses the rate of the communicating channel (which both of them know).

3 Modeling Techniques and Biological Examples

We present two examples for modeling biological systems with attribute dependent rates, in order to illustrate the advantages of $\pi(\mathcal{L})$ as a modeling language. The first example introduces a simplistic, discrete spatial model of Euglena's phototaxis [22], while the second more complex example shows how to deal with cooperative enhancement in gene regulation at the lambda switch [17].

3.1 Space

Spatial aspects of molecular systems gain increasing interest in systems biology [23]. We illustrate the use of attribute dependent rates for spatial modeling with a simple example, which is the modeling of Euglena's light dependent motion (phototaxis). More complex compartment structures can be modeled in the attributed π-calculus as well, as we will show by encoding π@ in Section 5.

Euglena is a single cell organism that lives in inland water and performs photosynthesis. Depending on the brightness, it swims up and down in order to reach a zone with just the right amount of light, [24]: if the amount of light decreases it moves towards a lighter zone, and vice verca. This behavior is specified in the attributed π-calculus model in Fig. 7. The model assumes a light source positioned over water, of which it distinguishes five discrete zones of water, of depths $\texttt{d} \in \{0, 1, 2, 3, 4\}$.

Euglenas at depth `d` are defined by processes `Euglena`(`d`). There are two channels `up` and `down` for upward respectively downward motions. The speed of upward motions is `d*(1-i)` where $\texttt{i} \in [0, 1]$ is the current light intensity, and the speed of the downwards movement is `(4-d)*i`. For $\texttt{i} = 0.5$, Euglenas are expected to concentrate at depth level `d` =2. Note that `Euglena`(0) cannot climb further, since the value of $0 * (1 - \texttt{i})$ is 0 and thus not successful for all `i`. For the same reason, `Euglena`(4) cannot descend. The interaction partners are processes `Light`(`i`) modeling light sources of intensity $\texttt{i} \in [0, 1]$. In our example, we use two light sources, with different intensities `Light`(0.5) and `Light`(0.6). `Euglena`(`d`)

Process definitions

```
Euglena(d) ≜ up[λi.d*(1−i)]?().Euglena(d−1)
           + down[λi.(4−d)*i]?().Euglena(d+1)
Light(i) ≜ up[i]!().Light(i)
         + down[i]!().Light(i)
```

Example solution

```
Euglena(2) | ...| Euglena(2) | Light(0.5) | Light(0.6)
```

Fig. 7. $\pi(\mathcal{L})$ model of Euglena's light-dependent motion: the rates for climbing and falling depend on the organism's current depth level and the light intensity

can adapt to different intensities of light. For this we use λ`i` abstractions in its definition.

The same system can be modeled in the stochastic π-calculus, since all parameters are finitely valued: `d` $\in \{0,\ldots,4\}$ and `i` $\in \{0.5, 0.6\}$. The idea is to duplicate channels for all possible reaction rates. In the above example, we need channels $\mathtt{up}_{\mathtt{d,i}}$ with rates `d*(i-1)`, and channels $\mathtt{down}_{\mathtt{d,i}}$ with rates `(4-d)*i`, for all possible values of `i` and `d` that yield nonzero rates. This leads to independent definitions of Euglenas for all possible depths, see Fig. 8.

$$
\begin{array}{lcl}
\mathsf{Euglena}_0() & \triangleq & \mathsf{down}_{0,0.5}?().\mathsf{Euglena}_1() + \mathsf{down}_{0,0.6}?().\mathsf{Euglena}_1() \\
\mathsf{Euglena}_1() & \triangleq & \mathsf{up}_{1,0.5}?().\mathsf{Euglena}_0() + \mathsf{up}_{1,0.6}?().\mathsf{Euglena}_0() \\
 & & + \mathsf{down}_{1,0.5}?().\mathsf{Euglena}_2() + \mathsf{down}_{1,0.6}?().\mathsf{Euglena}_2() \\
\ldots & & \\
\mathsf{Euglena}_4() & \triangleq & \mathsf{up}_{4,0.5}?().\mathsf{Euglena}_3() + \mathsf{up}_{4,0.6}?().\mathsf{Euglena}_3()
\end{array}
$$

Fig. 8. A model of Euglena's in the stochastic π-calculus. Distinct definitions for $\mathtt{Euglena}_d$ are used for all depth levels $d \in \{0, 4\}$.

Whether arbitrary processes of $\pi(\mathcal{L})$ are expressible in the stochastic π-calculus is open, particularly for infinitely valued parameters. Even if it is possible, the size of the the stochastic π-calculus definitions may often become prohibitively larger.

3.2 Cooperative Enhancement

Cooperative binding is a frequent and often decisive aspect in gene regulatory networks, where proteins stabilize each other's binding to neighboring DNA sites by adhesive contacts. In quantitative terms, the decay rate of one DNA-protein complex decreases by the existence of another. This is an instance of cooperative enhancement of reaction rates by third partners. As shown in [16,17], cooperative enhancement can be modeled in the stochastic π-calculus. It however requires nontrivial encodings, that can be alleviated within $\pi(\mathcal{L})$.

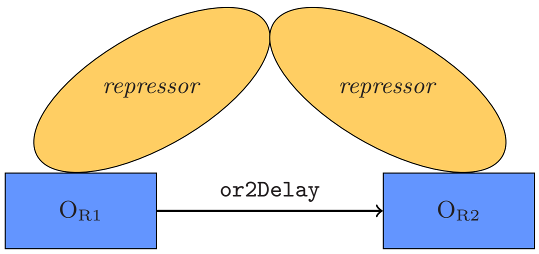

Fig. 9. The decay of the *repressor*-O_{R2} complex: in order to make the decay rate of the *repressor*-O_{R2} complex dependent on O_{R1}'s state, the two sites communicate on `or2Delay` before O_{R2} unbinds

A well understood instance of cooperative binding occurs during transcription initiation control at the λ switch. The λ switch is a segment of the DNA of bacteriophage λ. It contains two binding sites O_{R1} and O_{R2}, where *repressor* and *cro* proteins can bind. An unstable binding of a *repressor* molecule to O_{R2} is stabilized by the simultaneous presence of another *repressor* at the neighboring site O_{R1}. As illustrated in Fig. 9, the two proteins actually touch each other.

A $\pi(\mathcal{L})$ model of cooperative binding at O_{R2} is presented in Fig. 10. It contains the parametric definition `Prot(type)`, which emulates the behavior of the proteins. The parameter `type` can be instantiated by either `'rep'` or `'cro'`, for modeling *repressor* or *cro* proteins respectively. Proteins can bind to both sites O_{R1} and O_{R2}. Free sites are defined by processes $\mathtt{O_{R1}()}$ and $\mathtt{O_{R2}()}$, where proteins can attach via channel `bind`. As this occurs the channel `release` is created, and henceforth connects the protein to the site (complexation). Later communication on `release` breaks the complex. The reaction rate of complexation is fixed to `0.098`. For decomplexation the rate is determined by the sender, i.e. the binding site, the receiving protein accepts it by applying the identity function λ`r.r`.

Now consider the models for the protein bound DNA sites. $\mathtt{O_{Ri}Bound(type,release)}$ describes the unbinding from the occupied site O_{Ri}, where `type` indicates the type of the bound protein. For $i = 1$ the rate of the unbinding reaction merely depends on the protein type. Using the global channel `or1Delay` it is specified as `['val' or1Delay type]`. Recall that `'val'` simply access the channel's value from the environment.

For the second site ($i = 2$) decomplexation is influenced by cooperative binding. To model this, O_{R1} and O_{R2} are linked via the channel `or2Delay`, illustrated in Fig. 9. Additionally, the release operation is decomposed into an interaction on channel `or2Delay`, with a reaction rate defining the actual unbinding delay, and an immediate communication on `release`. As stated in the definition of the global channel `or2Delay` the unbinding delay depends not only on the type of the bound protein, but also on the state of O_{R1}, which can be either `'free'`, bound to `'rep'` or bound to `'cro'`. The reaction rate is computed by applying the function λ `state1.'val' or2Delay state1 type` to the state of O_{R1}.

Values of global channel names

```
bind: 0.098
or1Delay: λ type.
  if type='rep' then 0.155 else
  if type='cro' then 2.45
or2Delay: λ state1. λ type2.
  if state1='rep' then
     if type2='rep' then 0.155 else % big delay (cooperative)
     if type2='cro' then 3.99       % small delay
  else 2.45 % 'cro' or 'free'
```

Process definitions

```
Prot(type) ≜ (ν release:_)bind[_]!(type,release).
              release[λr. r]?().Prot(type)
O_R1() ≜
   bind[λ_. 'val' bind]?(type,release).O_R1Bound(type,release)
 + or2Delay['free']!().O_R1()
O_R1Bound(type,release) ≜
   release['val' or1Delay type]!().O_R1()
 + or2Delay[type]!().O_R1Bound(type,release)
O_R2() ≜
  bind[λ_. 'val' bind]?(type,release).O_R2Bound(type,release)
O_R2Bound(type,release) ≜
  or2delay[λ state1. 'val' or2Delay state1 type]?().
     release[∞]!().O_R2()
```

Example Solution

$\mathsf{O_{R1}}() \mid \mathsf{O_{R2}}() \mid \prod_{i=1}^{28} \mathsf{Prot}(\text{'rep'}) \mid \prod_{i=1}^{67} \mathsf{Prot}(\text{'cro'})$

Fig. 10. $\pi(\mathcal{L})$ model of cooperative binding between $\mathrm{O_{R1}}$ and $\mathrm{O_{R2}}$ at the λ switch

A previous model [17] in the the stochastic π-calculus calculus [2] requires to keep $\mathrm{O_{R2}}$ constantly informed about state changes of $\mathrm{O_{R1}}$, which is implemented by immediate communication steps. Keeping state information consistent in this manner is error-prone, it may easily lead to deadlocks. A subsequent model [16] in SPiCO [5], the the stochastic π-calculus calculus with concurrent objects, requires significantly fewer updates. In $\pi(\mathcal{L})$, reaction rates directly depend on the attribute values of the interaction partners. State changes are propagated without additional communication steps, preventing deadlocks.

Comments on the syntax. Biochemical forms are sometimes cumbersome for modeling, such that we prefer using the full language, as introduced in Section 2.2. In the definition of `Prot(type)` and `O_R2Bound(type,release)`, for instance, we use sequences of communication prefixes like $\pi_1.\pi_2.P$, which are a special case of nested sums and thus excluded by biochemical forms. The usual unnesting algorithm, which replaces nested sums by newly defined processes, provides the required transformation.

We freely use `if-then-else` statements in λ-terms of the attribute language, for instance in the values of channel names `or1Delay` and `or2Delay`. This is syntactic sugar increasing the readability of the model, which can be directly replaced by λ calculus expressions[2]. In practice, however, further syntactic sugar and extensions of the lambda calculus might be useful, in particular the addition of pattern matching `case` statements. These are advantageous for typeful programming, in contrast to pure `if-then-else` expressions.

4 Stochastic Simulator

The development of the simulator, as presented in this section, closely follows the introduced stochastic semantics in terms of continuous time Markov chains (Section 2.5). This shows that a simulator for $\pi(()\ \mathcal{L})$ can be obtained independently of the choice of $\mathcal{L}$, by extending of previous simulators for the stochastic π-calculus or SPiCO [4,5,3]. Our presentation differs previous ones though. The main advantage is to firmly root the simulator in the operational semantics. Algorithmic aspects remain mostly unchanged, but become applicable in greater generality.

The stochastic semantics for $\pi(\mathcal{L})$ induces the naive stochastic simulator given in Fig. 11. A simulator's input comprises a solution S, an environment ρ, and a time point $t \in \mathbb{R}$. The next reduction step for S is chosen in a memory less stochastic manner. The sojourn time $\Delta \geq 0$ in S is inferred, and the simulator proceeds with the resulting solution at time point $t + \Delta$.

Central to this simulator is to determine the set of labeled reactions in S with respect to the current environment ρ:

$$\texttt{Reacts} = \{(\ell, r) \mid \rho \vdash S \xrightarrow[\ell]{r} S'\}$$

Labeled reactions with rate $r = \infty$ are executed with priority and without time consumption. If no reaction with rate $r = \infty$ exists, we apply Gillespie's algorithm [25] to select a reaction $(\ell, r) \in \texttt{Reacts}(S)$ with probability r/s where $s = \sum_{(\ell,r') \in \texttt{Reacts}(S)} r'$. The sojourn time in S is $\Delta = -ln(1/U)/s$ for some uniformly distributed random number $0 < U \leq 1$.

The set `Reacts` can be computed as follows. First, all expressions in defined molecules $A(\tilde{e})$ of S are evaluated, and then replaced by their definitions. Next, all pairs of alternatives are enumerated and filtered for those that use the same channel and satisfy the communication constraint. The constraint is tested by applying the evaluation algorithm for the attribute language $\mathcal{L}$ in environment ρ. It terminates by assumption for error-free processes, so that the computation of `Reacts` terminates in that case too.

Most fortunately, the Markov chain itself does not need to be computed by the simulation algorithm. This would be largely unfeasible, since the number of possible outcomes of non-deterministic interactions may grow exponentially.

[2] When assuming $\rho \vdash \texttt{'true'}\ v_1\ v_2 \Downarrow v_1$ and $\rho \vdash \texttt{'false'}\ v_1\ v_2 \Downarrow v_2$ we can replace conditionals $\texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2$ by applications $e\ e_1\ e_2$.

```
Simulate−naive($S, \rho, t$)
  // $solution\ S,\ environment\ \rho : fn(S) \to Vals,\ time\ point\ t \in \mathbb{R}$
  case $S$
  of 0 then skip // termination
  of $\prod_{i=1}^{n} A_i(\tilde{e_i})$ then // no $\nu$ binders in $S$
     let Reacts = $\{(\ell, r) \mid \exists S'.\ \rho \vdash S \xrightarrow[\ell]{r} S', r \in Succ, \ell \in \mathbb{N}^4\}$

     if $\{(\ell, r) \in$Reacts$\mid r = \infty\} = \emptyset$
     then
        let $((\ell, r), \Delta)$ = Gillespie(Reacts) // (SUM)
        let $S'$ such that $S \xrightarrow[\ell]{r} S'$ // (COM)
        Simulate−naive($S', \rho, t + \Delta$)
     else
        select $(\ell, \infty)$ in Reacts with equal probability //(COUNT)
        let $S'$ such that $\rho \vdash S \xrightarrow[\ell]{\infty} S'$ // (COM)
        Simulate−naive($S', \rho, t$)
  else // some $\nu$ binder in $S$
     let $S', x, v$ such that $S \equiv (\nu x{:}v)\ S'$ with $x \notin dom(\rho)$
     let $\rho' = \rho \circ [v/x]$
     Simulate−naive($S', \rho', t$) // (NEW)
```

Fig. 11. Naive simulator interpreting the stochastic semantics

Furthermore, it would require to decide whether two solutions are structurally congruent (rules SUM and COUNT), which is a graph isomorphism complete problem [26].

In order to increase efficiency of the naive simulation algorithm, we apply an idea hidden already in the implementations BioSpi. The objective is to avoid the enumeration of all pairs of alternatives (and thus redexes), since there may be quadratically many in the size of S. The strategy is to *group* all reactions, using the same channel x and the same constraints $e_2 e_1$, modulo evaluation of e_1 and e_2, i.e. all reactions with $x[v_1]!\ldots$ and $x[v_2]?\ldots$ for some v_1 and v_2. We then apply the Gillespie algorithm to such *grouped* reactions.

A *label* for a grouped reaction in a solution S is a triple $L \in Chans(S) \times Vals^2(S)$. It represents the following set of reactions with respect to a solution S of the form $S = \prod_{i=1}^{n} A_i(\tilde{e_i})$ and an environment ρ:

$$\begin{aligned}\texttt{Reacts}(L) = \{(\ell, r) \in \texttt{Reacts} \mid\ & L = (x, v_1, v_2),\ \ell = (i_1, j_1, i_2, j_2),\\ & \text{choose}^{\rho}_{j_1}(A_{i_1}(\tilde{e_{i_1}})) = x[v_1]!\ldots, \text{choose}^{\rho}_{j_2}(A_{i_2}(\tilde{e_{i_2}})) = x[v_2]?\ldots\}\end{aligned}$$

The stochastic rate for a grouping label L is usually called propensity $\texttt{prop}(L) \in \mathbb{R}^+ \uplus \{\infty(n) \mid n \in \mathbb{N}\}$. It sums up all rates of the labeled reactions that it groups together, or counts the number of labels of infinite rate reactions if there are any:

$$\texttt{prop}(L) = \begin{cases} \infty(n) & \text{if } n = \#\{\ell \mid (\ell, \infty) \in \texttt{Reacts}(L)\} \geq 1 \\ \sum_{(\ell,r) \in \texttt{Reacts}(L)} r & \text{otherwise} \end{cases}$$

With this we can define the set of grouped reactions in S with respect to ρ. They will be used as input to Gillespie's algorithm:

$$\texttt{GReacts} = \{(L, \texttt{prop}(L)) \mid L \in \mathit{Chans}(S) \times \mathit{Vals}(S)^2\}$$

In practice, the cardinality of $\texttt{Reacts}(\mathrm{L})$ is often linear in the size of S. Fig. 12 gives a simulation algorithm based on grouped reactions with propensities. In contrast to the naive simulator, it first selects a label of a grouped reaction by Gillespie's algorithm, and then a label of a reaction in the group with equal distribution.

It remains to compute the propensities of all labels of grouped reactions in a solution S. These can be derived from the values below where $S = \prod_{i=1}^{n} A_i(\tilde{e}_i)$:

$$\begin{aligned} \mathit{out}(x, v_1) &= \#\{(i,j) \mid \mathrm{choose}_j^{\rho}(A_i(\tilde{e}_i)) = x[v_1]!\ldots\} \\ \mathit{in}(x, v_2) &= \#\{(i,j) \mid \mathrm{choose}_j^{\rho}(A_i(\tilde{e}_i)) = x[v_2]?\ldots\} \\ \mathit{mixin}(x, v_1, v_2) &= \#\{(i,j_1,j_2) \mid \mathrm{choose}_{j_1}^{\rho}(A_i(\tilde{e}_i)) = x[v_1]!\ldots, \\ &\qquad\qquad \mathrm{choose}_{j_2}^{\rho}(A_i(\tilde{e}_i)) = x[v_2]?\ldots\} \end{aligned}$$

Lemma 1. *$\texttt{prop}(x, v_1, v_2) = (\mathit{out}(x, v_1) * \mathit{in}(x, v_2) - \mathit{mixin}(x, v_1, v_2)) * r$ if the solution does not contain infinite rates and $\rho \vdash v_1 v_2 \Downarrow r$.*

The computation of mixins can still produce an output of quadratic size and thus need quadratic time. The square factor, however, is in the maximal number of alternatives in sums defining molecules of S, which will be small in practice. All other needed values can be computed in linear time in the size of S, when ignoring the time for evaluating expressions, which is justified in many practical cases.

The final step toward an efficient simulator consists in computing the propensities $\texttt{prop}(x, v_1, v_2)$ incrementally, so that they don't have to be recomputed from scratch in every reduction step. This can be based on Lemma 1, since the values of $\mathit{out}(x, v_1)$, $\mathit{in}(x, v_2)$, $\mathit{mixin}(x, v_1, v_2)$ can be updated incrementally, when adding new solutions or canceling alternative choices by communication.

5 Polyadic Synchronization and Compartments

In this section, we encode a non-deterministic variant of π@ [15] in $\pi(\mathcal{L}_{\wedge,=})$, where $\mathcal{L}_{\wedge,=}$ is the attribute language providing conjunctions of equalities between channel names, as defined in Section 2.1.

The calculus π@ is an extension of the π-calculus, able to express various spatial aspects of compartment organization. There are two main ingredients in π@, polyadic synchronization and different levels of priorities for different types of reduction. Adding priorities to the $\pi(\mathcal{L})$ is straightforward. They can be defined by a total order on the successful values *Succ* of the attribute language. The operational semantics has to be adapted such that reduction steps with greater rates are executed first. This can be done by labeling possible reduction steps with the rate values of their constraints. Indeed, the stochastic semantics of

```
Simulate ($S, \rho, t$)  //solution $S$, environment $\rho : fn(S) \to Vals$, time point $t \in \mathbb{R}$
  case $S$
  of $\mathbf{0}$ then skip  // termination
  of $\prod_{i=1}^{n} A_i(\tilde{e_i})$ then  // no $\nu$ binders in $S$
     let GReacts = $\{(L, \mathtt{prop}(L)) \mid L \in Chans(S) \times Vals(S)^2\}$
     if $\{(L, r) \in$GReacts$\mid r = \infty(n)\} = \emptyset$
     then
        let $((L, r), \Delta)$ = Gillespie(GReacts)
        select $(\ell, r)$ in GReacts($L$) equally distributed
        let $S'$ such that $S \xrightarrow[\ell]{r} S'$
        Simulate($S', \rho, t + \Delta$)
     else
        select $(L, \infty(n))$ in GReacts
           with probability $n/m$ where $m = \sum_{(L', \infty(n') \in \mathtt{GReacts}} n'$
        select $(\ell, \infty)$ in Reacts($L$) with equal probability
        let $S'$ such that $\rho \vdash S \xrightarrow[\ell]{\infty} S'$
        Simulate($S', \rho, t$)
  else // some $\nu$ binder in $S$
     let $S', x, v$ such that $S \equiv (\nu x{:}v)\ S'$ with $x \notin dom(\rho)$
     let $\rho' = \rho \circ [v/x]$
     Simulate($S', \rho', t$)  // (NEW)
```

Fig. 12. Stochastic simulator for $\pi(\mathcal{L})$ (to be implemented incrementally)

$\pi(\mathcal{L})$ is already defined such that it gives highest priority to immediate reactions (with rate ∞). For sake of simplicity, we ignore priorities and focus on how to express polyadic synchronization in the following.

The syntax of π@ is given in Figure 13. It is similar to the syntax of the π-calculus with two exceptions. Inputs have the form $\tilde{x}?\tilde{z}.P$ and outputs the form $\tilde{x}!\tilde{y}.P$. A communication is possible for inputs and outputs that use the same sequence of channels rather than a single channel only:

$$\ldots + \tilde{x}?\tilde{z}.P \mid \tilde{x}!\tilde{y}.Q + \ldots \;\to\; P[\tilde{y}/\tilde{z}] \mid Q$$

The only values of π@ are channels. Local binders $(\nu x)Q$ do not assign any values to channels. Therefore, the semantics of π@ does not consider environments.

The encoding $[\![\_]\!] : \pi@ \to \pi(\mathcal{L}_{\wedge,=})$ is compositional; only, the encoding of communication prefixes deserves special attention:

$$\begin{aligned}[\![(x_1, \ldots, x_n)!\tilde{z}.P]\!] &= x_1[\lambda y_2 \ldots \lambda y_n. \\ &\qquad \texttt{'and'} \ldots \texttt{'and'}(\texttt{'equal'}x_2y_2) \ldots (\texttt{'equal'}x_ny_n)]\tilde{z}.[\![P]\!] \\ [\![(y_1, \ldots, y_n)?\tilde{z}.P]\!] &= y_1[\lambda e.ey_2 \ldots y_n]?\tilde{z}.[\![P]\!]\end{aligned}$$

Tuple equality in π@ is translated as a conjunction of name equalities in the attribute language: an output with subject $(x_1, \ldots, x_n)$ is translated as an output with subject x_1 with a constraint that checks whether its first argument (bound to y_2) is equal to x_2 *and* its second argument (bound to y_3) is equal to

Processes	$P ::= A(\tilde{x})$	defined process
	$\mid\ P_1 \mid P_2$	parallel composition
	$\mid\ (\nu x)P$	channel creation
	$\mid\ \pi_1.P_1 + \ldots + \pi_n.P_n$	summation of alternative choices
	$\mid\ \mathbf{0}$	empty solution
Prefixes	$\pi ::= \tilde{x}?\tilde{y}$	receiver
	$\mid\ \tilde{x}!\tilde{y}$	sender
Definitions	$D ::= A(\tilde{x}) \triangleq P$	parametric process definition

Fig. 13. Syntax of π@ [15] where $x, \tilde{x}, \tilde{y} \in \mathit{Chans}$

x_3, etc. Dually, an input with subject $(y_1, \ldots, y_n)$ is translated as an input with subject y_1 with a constraint applying its argument to $y_2, \ldots, y_n$. We also have to translate the definitions of molecules in use

$$[\![A(\tilde{x}) \triangleq P]\!] = A(\tilde{x}) \triangleq [\![P]\!]$$

and consider reduction with respect to a set of such encoded definitions.

Theorem 1 (Operational correspondence)

(a) if $S_1 \to S_2$ w.r.t. $\mathcal{D}$, then $[\![S_1]\!] \to [\![S_2]\!]$ w.r.t. $[\![\mathcal{D}]\!]$
(b) if $[\![S]\!] \to S_1$ w.r.t $[\![\mathcal{D}]\!]$, then $\exists S_2$ s.t. $S_1 \equiv [\![S_2]\!]$, and $S \to S_2$ w.r.t $\mathcal{D}$.

The proof is mostly straightforward. It might be worth noticing, however, that all λ expressions terminate in linear time. This can be seen by inspecting the occurring terms.

6 Conclusion and Outlook

We presented the attributed π-calculus $\pi(\mathcal{L})$, which incorporates an attribute language $\mathcal{L}$ fixing the values and constraints. Since $\mathcal{L}$ is introduced as a parameter, a proliferation of domain specific calculi can be avoided. The central idea of our approach is to control communication by constraints on attribute values of processes. $\pi(\mathcal{L})$ can encode π@ and thus, also BioAmbients, and BraneCalculi. Furthermore, an encoding of a version of Beta-binders into π@ was proposed recently [27]. Beyond discrete spatial structures, e.g. a cell's compartments, spatial attributes like volume and position can influence reaction rates. Thus, $\pi(\mathcal{L})$ appears particularly promising for modeling spatial processes. We have illustrated the usefulness of $\pi(\mathcal{L})$ as a modeling language for systems biology by the examples of cooperative protein-DNA binding in gene regulation of bacteriophage λ, and the phototaxis of the single celled organism Euglena. Currently, a model of the Wnt-pathway with focus on membrane interactions and receptors is being developed. If constitutes a further test case for the modeling formalism.

We defined a non-deterministic and a stochastic semantics for $\pi(\mathcal{L})$. The stochastic semantics has been specified in terms of a continuous time Markov chain (CTMC) and then transferred to a simulator, independently of the choice of $\mathcal{L}$. This simulator permits to reuse all algorithmic tricks of state-of-the-art simulators for the stochastic π-calculus. An implementation of the simulator is currently under way. Identifying parameter values for the Wnt-pathway model and, in addition, performance studies along the lines of [28] will, both, challenge and help assessing the simulator's practical efficiency.

In the context of modeling, one shall explore the implication of $\pi(\mathcal{L})$'s central idea, i.e. to map attribute values to interaction affinity, on discrete modeling languages, e.g. rule-based ones. Another issue to be investigated is its potential for large scale modeling. Therefore, object-oriented language concepts such as inheritance, shall be lifted from SPiCO [5] to $\pi(\mathcal{L})$. With respect to language theory, another yet open question is whether one can encode $\pi(\mathcal{L})$ into the π-calculus.

Acknowledgements

We thank Céline Kuttler for suggesting our collaboration, and valuable comments on this paper. Part of the research was financed by the DFG in the context of the Research Training School "dIEM oSiRiS". The BioComputing activity of the LIFL in Lille is funded by the ANR Jeunes Chercheurs BioSpace.

References

1. Regev, A., Shapiro, E.: Cells as Computation. Nature 419, 343 (2002)
2. Regev, A.: Computational Systems Biology: A Calculus for Biomolecular Knowledge. Tel Aviv University, PhD thesis (2003)
3. Priami, C., Regev, A., Shapiro, E., Silverman, W.: Application of a Stochastic Name-Passing Calculus to Representation and Simulation of Molecular Processes. Information Processing Letters 80, 25–31 (2001)
4. Phillips, A., Cardelli, L.: Efficient, Correct Simulation of Biological Processes in the Stochastic Pi Calculus. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 184–199. Springer, Heidelberg (2007)
5. Kuttler, C., Lhoussaine, C., Niehren, J.: A Stochastic Pi Calculus for Concurrent Objects. In: Anai, H., Horimoto, K., Kutsia, T. (eds.) Ab 2007. LNCS, vol. 4545, pp. 232–246. Springer, Heidelberg (2007)
6. Dematté, L., Priami, C., Romanel, A.: Modelling and Simulation of Biological Processes in BlenX. SIGMETRICS Performance Evaluation Review 35(4), 32–39 (2008)
7. Ciocchetta, F., Hillston, J.: Bio-PEPA: An Extension of the Process Algebra PEPA for Biochemical Networks. ENTCS 194(3), 103–117 (2008)
8. Chabrier-Rivier, N., Fages, F., Soliman, S.: The Biochemical Abstract Machine BIOCHAM. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 172–191. Springer, Heidelberg (2005)
9. Faeder, J.R., Blinov, M.L., Goldstein, B., Hlavacek, W.S.: Rule-Based Modeling of Biochemical Networks. Complexity 10(4), 22–41 (2005)

10. Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J.: Rule-based modelling of cellular signalling. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 17–41. Springer, Heidelberg (2007)
11. Hillston, J.: Process Algebras for Quantitative Analysis. In: LICS 2005: Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science, pp. 239–248. IEEE Computer Society, Los Alamitos (2005)
12. Cardelli, L.: On Process Rate Semantics. TCS 391(3), 190–215 (2008)
13. Jaffar, J., Lassez, J.L.: Constraint Logic Programming. In: 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, pp. 111–119. ACM Press, New York (1987)
14. Saraswat, V.A., Rinard, M., Panangaden, P.: The Semantic Foundations of Concurrent Constraint Programming. In: 18th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 333–352. ACM Press, New York (1991)
15. Versari, C.: A Core Calculus for a Comparative Analysis of Bio-inspired Calculi. Programming Languages and Systems, pp. 411–425 (2007)
16. Kuttler, C.: Modeling Bacterial Gene Expression in a Stochastic Pi Calculus with Concurrent Objects. PhD thesis, Université des Sciences et Technologies de Lille - Lille 1 (2007)
17. Kuttler, C., Niehren, J.: Gene regulation in the pi calculus: Simulating cooperativity at the lambda switch. Transactions on Computational Systems Biology VII, 24–55 (2006)
18. Versari, C., Busi, N.: Stochastic Simulation of Biological Systems with Dynamical Compartment Structure. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 80–95. Springer, Heidelberg (2007)
19. Versari, C., Busi, N.: Efficient Stochastic Simulation of Biological Systems with Multiple Variable Volumes. ENTCS 194(3), 165–180 (2008)
20. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: BioAmbients: An Abstraction for Biological Compartments. TCS 325(1), 141–167 (2004)
21. Cardelli, L.: Brane calculi. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 257–278. Springer, Heidelberg (2005)
22. John, M., Ewald, R., Uhrmacher, A.M.: A Spatial Extension to the Pi Calculus. ENTCS 194(3), 133–148 (2008)
23. Kholodenko, B.N.: Cell-Signalling Dynamics in Time and Space. Nature Reviews Molecular Cell Biology 7(3), 165–176 (2006)
24. Grell, K.G.: Protozoologie. Springer, Heidelberg (1968)
25. Gillespie, D.T.: A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. Journal of Computational Physics 22(4), 403–434 (1976)
26. Khomenko, V., Meyer, R.: Checking Pi Calculus Structural Congruence is Graph Isomorphism Complete. Technical Report CS-TR: 1100, School of Computing Science, Newcastle University, 20 pages (2008)
27. Cappello, I., Quaglia, P.: A translation of beta-binders in a prioritized pi-calculus. In: From Biology to Concurrency and back, Workshop FBTC (2008)
28. Ewald, R., Jeschke, M.: Large-Scale Design Space Exploration of SSA. In: Computational Methods in Systems Biology, International Conference CMSB 2008. LNCS. Springer, Heidelberg (2008)