HAL
open science

A Spatial Extension to the Pi-Calculus

Mathias John, Roland Ewald, Adelinde Uhrmacher

To cite this version:

Mathias John, Roland Ewald, Adelinde Uhrmacher. A Spatial Extension to the Pi-Calculus. From Biology To Concurrency and back, Sep 2007, Lissabon, Portugal. 10.1016/j.entcs.2007.12.010 . hal-00656206

HAL Id: hal-00656206

https://inria.hal.science/hal-00656206v1

Submitted on 3 Jan 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Spatial Extension to the π Calculus

Mathias John[1,2] Roland Ewald[3] Adelinde M. Uhrmacher[4]

Department of Computer Science
University of Rostock
D-18051 Rostock, Germany

Abstract

Spatial dynamics receive increasing attention in Systems Biology and require suitable modeling and simulation approaches. So far, modeling formalisms have focused on population-based approaches or place and move individuals relative to each other in space. SpacePi extends the π calculus by time and space. π processes are embedded into a vector space and move individually. Only processes that are sufficiently close can communicate. The operational semantics of SpacePi defines the interplay between movement, communication, and time-triggered events. A model describing the phototaxis of the Euglena micro-organism is presented as a practical example. The formalism's use and generality is discussed with respect to the modeling of molecular biological processes like diffusion, active transportation in cell signaling, and spatial structures.

Keywords: pi calculus, spatial modeling, systems biology.

1 Introduction

The majority of modeling and simulation approaches that are currently exploited in Systems Biology do not support an explicit representation and evaluation of spatial phenomena. With the progress of wet-lab techniques, spatial information becomes available that emphasizes the central role space plays in inter- and intracellular dynamics: e.g., crucial phases in the mitosis are distinguished by the spatial distribution of key actors in the cell, and signaling pathways depend on the location and movement of their components [27,26]. The relay of information in cells depends on diffusion and active transportation processes [10].

Consequently, a realistic modeling of many cellular phenomena requires that space is taken into account [10]. The modeling formalisms that have been exploited in Systems Biology can be distinguished along two different dimensions. One is whether the approach focuses on a population-based or individual-based perception

[1] This research has been funded by the DFG. We thank Céline Kuttler and Mathias Röhl for their helpful comments on earlier versions of this work.

[2] Email: mathias.john@uni-rostock.de

[3] Email: roland.ewald@uni-rostock.de

[4] Email: adelinde.uhrmacher@uni-rostock.de

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

of biological systems. The other dimension is whether a formalism allows an explicit representation of absolute space, or uses the components, their inter-relations, and their relative placement as a starting point to define space implicitly. One focus of research has been on combining absolute space and population-based approaches. E.g., partial differential equations have been used to represent signaling processes in membranes assuming a homogeneous distribution of receptors [8], and in [5] cellular automata describe the population growth of cells in space. Delays are introduced into differential equations that capture spatial phenomena implicitly, e.g. in [24] discrete time delays are used to describe transcription, translation, and nuclear transport. Thus, a population-based approach is combined with an indirect representation of space. Population-based approaches assume homogeneous distributions, which constraints their applicability, e.g. they are not suitable for realistic representations of membrane micro-domains [12]. Small numbers of actors, different geometries of components, and tracing of components causes additional problems. Furthermore, population-based approaches suffer from state space explosion [27]. This is the motivation to develop and use spatial individual-based approaches in Systems Biology. Those have so far focused on indirect, relative space. Examples are Beta Binders [20], Membrane Calculi [4] or Bio Ambients [21], which address the need to structure space into compartments and confine processes and their interactions spatially. Given the experimental set-ups, e.g. confocal microscopy, biologists are particularly interested in a combination of the individual-based approach with absolute space. Whereas some simulation systems exist that support this combination, e.g. [25,16,1], little work has so far been done with respect to modeling formalisms. We will exploit process algebras to develop a modeling formalism that allows to explicitly model location and movement of individuals in absolute space, as their use has been demonstrated in Systems Biology [18].

2 Extending the π Calculus with Space and Time

The π calculus [13] is a model of concurrent computation and is based on the notion of naming. Names represent both interconnection links between active entities, called processes, and the data that these entities exchange through communication. We extend the π calculus to allow a free movement of parallel π processes. Adding a spatial notion to a process definition is straightforward, as each process P can be associated with a certain position $\overrightarrow{p} \in V$, $V = \mathbb{R}^d$ being a vector space with a norm $\| \|_V$.

The introduction of movement also requires a notion of time, since the speed of motions could not be expressed otherwise. Adding time is not trivial, since we expect all communication to occur while processes are moving. Usually, a timed process algebra is built by distinguishing two phases: one in which all processes perform their actions, and one for the time to proceed [14].

Our approach expresses time by intervals and allows communication in a discrete-event manner during each interval. Then, the same principle as described in [17,14] is applied: for defining the semantics, we use a single rule that ensures the progress from time interval to time interval, and others to define the activities taking place in each interval. It is assumed that all processes move uniformly

during each interval, i.e. their velocity is constant and they move along a straight line. The length of each interval is δ_t time units, and non-uniform motion can be approximated by choosing a suitably small δ_t.

Having defined space and time, it is now possible to associate a *movement function* $m : V \times X \to V$ with each process. It takes the current position of a process into account, as well as an element from the set of additional parameters, X, that could be used to represent forces influencing the processes' movement. As the additional parameters depend on the abstraction level and the application area, X and a function to select parameters $\chi \in X$ have to be provided with the model. The movement function is used to generate a *target vector* of the process. Adding a target vector to the position $\overrightarrow{p}$ of a process at the beginning of the time interval will result in the *target position* $\overrightarrow{t} = \overrightarrow{p} + m(\overrightarrow{p}, \chi)$, which will be reached after the time interval is over, i.e. δ_t time units have passed.

As each process P is now associated with its current position $\overrightarrow{p}$ and a movement function m, it can be written as $P_m^{\overrightarrow{p}}$. The position vector is superscripted and, since it is a vector, marked with an arrow. The movement function of the process is subscripted. This notation is used throughout this paper.

We define the set of all SpacePi expressions rather similarly to the π calculus, except that we emulate replication, i.e. it is not part of the basic definition. An empty process, *nil*, is introduced, which does not move and has its position at V's point of origin. In other words, *nil* is a short hand for $nil^{\overrightarrow{0}}_{m(\overrightarrow{0},\chi)=\overrightarrow{0}}$. The formal definition of a SpacePi expression is as follows:

$$P_m^{\overrightarrow{p}} ::= \sum_{i \in I} \pi_i . P_{i\ m_i}^{\overrightarrow{p}_i} \mid P_{1\ m_1}^{\overrightarrow{p}_1} | P_{2\ m_2}^{\overrightarrow{p}_2} \mid (new\ x) P_m^{\overrightarrow{p}} \mid nil$$

This strongly resembles to the π calculus, even more so when regarding the position and movement function of each process as its parameters (see [13]). Let $\mathcal{P}_S$ be the set of all SpacePi processes. Furthermore, let $\mathcal{C}h$ be the some set of channel names. The action π can either be:

$$\begin{aligned} \overline{ch}\langle x, r\rangle,\ & \textit{i.e. sending } x \textit{ over channel } ch \textit{ with radius } r \\ ch\langle x, r\rangle,\ & \textit{i.e. receiving } x \textit{ over channel } ch \textit{ with radius } r \\ wait\langle t\rangle,\ & \textit{i.e. waiting until time } t \end{aligned}$$

with $r \in \mathbb{R}_0^+ \cup \{\infty\}$, $t \in \mathbb{R}_0^+$ and $ch, x \in \mathcal{C}h$. Two processes may only communicate over a channel if the sum of their corresponding send/receive action radii exceeds the distance between them (see Fig. 1). A communication action with $r = \infty$ has the same semantics as in the π calculus, i.e. the process sends/receives without any spatial restrictions. The *wait* action is an addition to the π calculus, but its semantics is similar to the *idle* action d from [14]: this action occurs after time t, so that a process $P = wait(0.5).Q + \pi.Q'$ will be transformed either to Q at time $t = 0.5$ or to Q' if π executes before $t = 0.5$ (cf. weak choice in [14]). Although we regard t as absolute time, relative times could be easily integrated by

substitution at 'runtime'. We omit a polyadic version of the calculus for the sake of clarity. It would be quite similar to the polyadic π calculus.

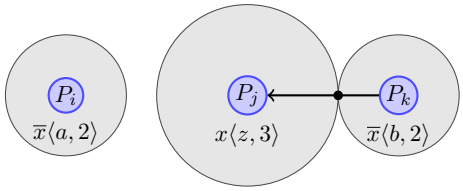

Fig. 1. Communication of two processes: At this moment, P_i and P_j may not communicate over channel x, since the sum of their action's radii, 2 and 3 (denoted by gray circles), does not exceed the distance between them. Instead, P_j and P_k are able to communicate over x. The movement of the processes is left out in this figure.

SpacePi expressions use the same operators as π processes, so it can be presupposed that a normal form (or standard form, see [13, p. 90]) can be found for each expression. A normal form is an expression $(new\ \widetilde{a})(M_1|\ldots|M_n)$, with $\widetilde{a}$ being a set of restricted names [5] and each process M_i is a sum.

2.1 Semantics

At the beginning of an interval, the current position and movement function of each concurrently running process (i.e., a sum M_i of the normal form) are used to determine the *target position* to be reached at the end of the interval. Processes may act or interact during an interval. They act when they reach a time-out as defined by a *wait* action, and they interact using communication. Since all processes are moving uniformly and their initial positions as well as target positions are known, it is possible to select the next pair of processes that is able to communicate (see Fig. 2). Likewise, it is possible to select the next time-triggered action that occurs.

As already mentioned, we use two kinds of rules to express progress in time: Rules that are applied *during* a time interval and specify the ability of the processes to act and interact, and a rule that defines the transition between one time interval and another (see sec. 2.2). These rules are used in alternation. The transition between one interval and the next one is not triggered until there are no further actions or interactions to be executed.

During a time interval, each activity will be associated with a certain point in time, defined by $\lambda \in [0, 1)$, which can be regarded as the current time offset. If a rule is applied at time offset λ, it actually occurs at time $(ti + \lambda) \cdot \delta_t$, with $ti \in \mathbb{N}_0$ being the number of the current time interval. Therefore, λ can be regarded as the relative time in a given interval. A rule may only be applied at time λ if there is no rule applicable at $\lambda' \in [0, \lambda)$. By this restriction, we ensure a rule application that is ordered with respect to time. If more than one rule is applicable at the same point in time, it is not determined which rule is applied. All in all, we use a discrete-event approach to account for all actions that may occur in one interval, and a discrete-time approach to model movement, since the target positions are generally updated every δ_t time units.

[5] We use the notation of $\widetilde{a}$ instead of $\overrightarrow{a}$ (as in [13]) to distinguish between sets of restricted names and (spatial) vectors.

However, if a process acts or interacts, its target position $\overrightarrow{t}$ may change. In other words, a process may also change its direction and velocity *during* an interval (see Fig. 2). A new target vector is generated for each new parallel process. Every new vector $\overrightarrow{t}_{new}$, generated by the new movement function, has to be scaled with $1-\lambda$ to account for the fact that the process only moves into that direction from time $(ti+\lambda)\cdot\delta_t$ to $(ti+1)\cdot\delta_t$. During the time interval, the process would thus move from its initial position $\overrightarrow{p}$ to $\overrightarrow{p}' = (\overrightarrow{p}+\lambda\cdot(\overrightarrow{t}_{old}-\overrightarrow{p}))$, and after a communication at time λ it would move from $\overrightarrow{p}'$ to $(\overrightarrow{p}' + (1-\lambda)\cdot(\overrightarrow{t}_{new}-\overrightarrow{p}'))$. A process might communicate multiple times during an interval, but since uniform motion is assumed, its movement can be expressed as a sum of its original position and scaled vectors to its target positions. If a parallel process is spawned at a new position in V, the vector to the target position is scaled in the same way, as there is still only $(1-\lambda)\cdot\delta_t$ time left to move in the current time interval.

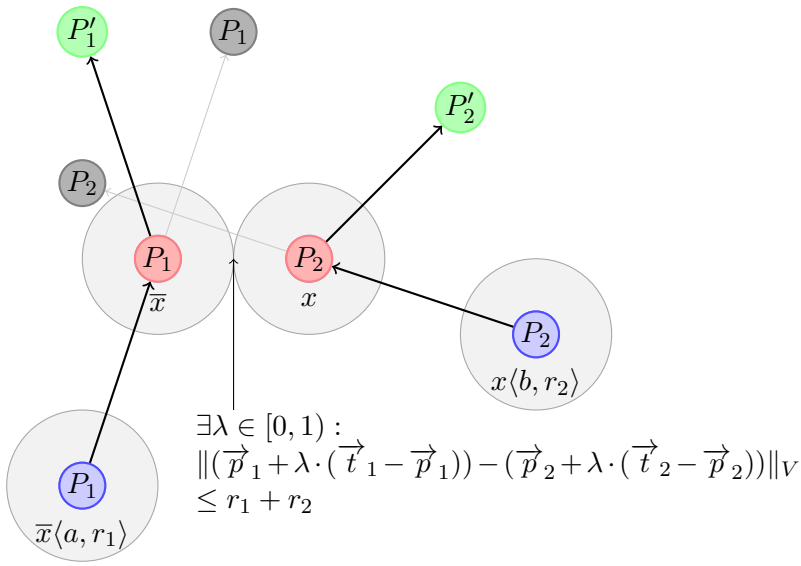

Fig. 2. Communication of two processes: P_1 may send a message to P_2 over x if there is a point λ in the current time interval at which the sum of their radii regarding x is larger than or equal to their distance at this moment. The vectors $\overrightarrow{p_i}$ define the initial positions and the vectors $\overrightarrow{t_i}$ the target positions.

2.2 Formal semantics

We define the operational semantics as rules on a tuple, but with the help of some predicates. A tuple

$$\langle \widetilde{a}, CP, T, ti, \lambda, \delta_t\rangle$$

defines the current configuration, with $CP = (P_{1\ m_1}^{\overrightarrow{p}_1}, \ldots, P_{k\ m_k}^{\overrightarrow{p}_k})$ being the tuple of parallel processes in the normal form $(new\ \widetilde{a})(P_{1\ m_1}^{\overrightarrow{p}_1}|\ldots|P_{k\ m_k}^{\overrightarrow{p}_k})$, i.e. each process is a sum of processes guarded by some action π. $T = (\overrightarrow{t}_1, \ldots, \overrightarrow{t}_k)$ is the tuple of target positions for each of the k parallel processes, i.e. $\overrightarrow{t}_1 = \overrightarrow{p}_1 + m_1(\overrightarrow{p}_1, \chi)$ is the destination of $P_{1m_1}^{\overrightarrow{p}_1}$ and so on. To make the following expressions clearer, the position vector and the movement function of a process are omitted whenever irrelevant.

As already defined in section 2.1, $ti \in \mathbb{N}_0$ is the number of the current time interval, $\lambda \in [0,1)$ is the time offset for time interval ti and δ_t is the timespan of each interval, so that $(ti + \lambda) \cdot \delta_t$ defines the current time of the system.

Two predicates are defined on this configuration: $Comm$ is true if a communication is possible and $Wait$ is true if the end time of a wait action has been reached. If neither is true, no parallel process may act or interact at any remaining moment of the current time interval. This results in the transition from one interval to another. The predicates are used to express the conditions of the four rules that constitute the semantics of a SpacePi execution.

The predicate $Comm$ defines under which circumstances process $P_{1\ m_1}^{\overrightarrow{p}_1}$ is able to send y over channel ch to $P_{2\ m_2}^{\overrightarrow{p}_2}$, where y then substitutes x:

$$
\begin{aligned}
(P_1, P_2, Q_1, Q_2, ch, x, y, CP, T, o) \in Comm \iff \\
P_{1\ m_1}^{\overrightarrow{p}_1}, P_{2\ m_2}^{\overrightarrow{p}_2} \in CP \wedge \overrightarrow{t}_1, \overrightarrow{t}_2 \in T \wedge ch, x, y \in \mathcal{C}h \wedge o \in [0, 1-\lambda) \wedge \\
(P_1 = \ldots + \overline{ch}\langle y, r_1\rangle.Q_1 + \ldots) \wedge (P_2 = \ldots + ch\langle x, r_2\rangle.Q_2 + \ldots) \wedge \\
\|(\overrightarrow{p}_1 + o \cdot (\overrightarrow{t}_1 - \overrightarrow{p}_1)) - (\overrightarrow{p}_2 + o \cdot (\overrightarrow{t}_2 - \overrightarrow{p}_2))\|_V \leq r_1 + r_2
\end{aligned}
$$

The first terms of the predicate simply ensure that all variables are from the correct domain, and that P_1 and P_2 are currently able to send/receive over the same channel. Fulfilling the last term requires that the sum of the communication action's radii, r_1 and r_2, has to be greater than or equal to the distance of both processes at a certain time offset o (see Fig. 2). Note that there are two offsets, λ and o. λ is the *current* time offset, while o denotes the offset to the next moment at which an interaction may happen. Since communication ability is only defined within the current interval, the sum of λ and o must be smaller than 1, which means that o has to be in $[0, 1-\lambda)$. The predicate $Wait$ is true if there is a process P that acts time-triggered due to a wait action:

$$
\begin{aligned}
(P, Q, CP, ti, \lambda, o) \in Wait \iff \\
P \in CP \wedge o \in [0, 1-\lambda) \wedge P = \ldots + wait\langle t\rangle.Q + \ldots \ \wedge t = (ti + \lambda + o) \cdot \delta_t
\end{aligned}
$$

One can now define the time offset to the next event (i.e., process action or interaction) as follows:

$$
\begin{aligned}
o_{min}(CP, T, ti, \lambda) = \\
\min_{o \in [0, 1-\lambda)} : \exists P, Q \in CP : Wait(P, Q, CP, ti, \lambda, o) \vee \\
\exists P_1, P_2 \in CP, ch, x, y \in \mathcal{C}h : Comm(P_1, P_2, Q_1, Q_2, ch, x, y, CP, T, \lambda, o)
\end{aligned}
$$

For a given configuration $\langle CP, T, ti, \lambda, \delta_t\rangle$, let o_{min} be $o_{min}(CP, T, ti, \lambda)$ if an o exists that satisfies either predicate, otherwise $o_{min} = \infty$. o_{min} is therefore the time offset to the next event. If it equals ∞, neither communication nor time-triggered

actions can be executed in the current time interval, and the transition to the next time interval is possible.

The definition of the operational semantics rules requires some auxiliary functions. *move* takes the tuple of current processes, CP, as well as the tuple of corresponding target vectors T and a time offset x as parameters. It updates the position of all $P_{i\ m_i}^{\overrightarrow{p}_i} \in CP$ to $\overrightarrow{p}_i + x \cdot \overrightarrow{t}_i$ and returns the updated tuple. In other words, it executes the movement of all processes for a timespan of $x \cdot \delta_t$ time units.

There are two additional functions, fn and *clear*, that take a normal form as a parameter. Since the normal form is divided into the restrictions $\widetilde{a}$ and a tuple CP of parallel processes, these are given separately. Similar to [13], $fn(\widetilde{a}, X)$ returns all free names of the processes in X. We introduce *clear* to filter restricted names that are no longer known to any process and can therefore be deleted from $\widetilde{a}$. To generate normal forms that conform to the free and restricted names of the normal form $(new\ \widetilde{a})CP$, a function $nf(P, \widetilde{a})$ is defined, which returns a tuple $(\widetilde{x}, CP_X)$ representing the normal form of P, while ensuring with appropriate α-conversion that all names from $\widetilde{a}$ are avoided. Finally, *ind* returns the index of a process in CP. The rules that define the operational semantics are given in figure 3.

The first rule, *Communication*, describes how a communication reaction is executed. If the possibility of a communication exists and the time of this event is minimal, P_1 and P_2 communicate over channel *ch*. Both are reduced to the processes A and B, respectively, which are then integrated into CP' and T'. Note that there is a name substitution in all processes of B's normal form (denoted by $CP_B\{x/y\}$), just as it is done in the π calculus. All traces of P_1 and P_2 have to be removed from the configuration. Note that *all* processes are moved using the *move* function. The *Wait* rule works similarly.

Finally, the *Interval transition* rule re-sets the current time offset λ to 0 while incrementing the number of the current interval, *ti*, and overwrites all process positions with the corresponding target position from T. The *Nil process* rule can be used to reduce *nil* processes.

2.3 Notation

The following notations and specifications have been found to make SpacePi models much easier to grasp and less cluttered, while retaining the full expressive power of SpacePi.

Non-spatial processes: If processes are defined so that their position is $\overrightarrow{0}$, their movement function is $m_0(\overrightarrow{0}, \chi) = \overrightarrow{0}$, and their send/receive actions all have the radius ∞, one can regard them as 'non-spatial' processes. By doing so, a combination of non-spatial and spatial model aspects is possible.

Replication: Replication can be emulated by defining a parametrized process $Q_P(r, \overrightarrow{p}, m) = wait\langle r\rangle.(P_m^{\overrightarrow{p}} | Q_P(r, \overrightarrow{p}, m))$. One could also pass a function that generates a random position for each newly created process.

Omitting standard transitions: It is quite usual to spawn a new process at the same point where the preceding process has been before. To facilitate this, the modeler may signal the use of the same position or movement function by leaving any of the definitions out. Hence, $P_m^{\overrightarrow{p}} = x\langle y, 0.1\rangle.Q$ means that Q will eventually

Communication
if $o_{min} < \infty \wedge Comm(P_1, P_2, A, B, ch, x, y, CP, T, \lambda, o_{min})$:
$\langle \widetilde{a}, CP, T, ti, \lambda, \delta_t \rangle \rightarrow \langle \widetilde{a}', CP', T', ti, \lambda', \delta_t \rangle$ with:
$(\widetilde{a}_A, CP_A) = nf(A, \widetilde{a} \cup fn(\widetilde{a}, CP))$, $(\widetilde{a}_B, CP_B) = nf(B, \widetilde{a} \cup \widetilde{a}_A \cup fn(\widetilde{a} \cup \widetilde{a}_A, CP + CP_A))$
$CP' = (move(CP, T, o_{min}) - (P_1, P_2)) + CP_A + CP_B\{x/y\}$
$\lambda' = \lambda + o_{min}$, $\widetilde{a}' = clear(\widetilde{a} \cup \widetilde{a}_A \cup \widetilde{a}_B, CP')$
$T' = T - (\overrightarrow{t}_{ind(P_1)}, \overrightarrow{t}_{ind(P_2)}) + \bigcup_{Q_i \in CP_A \cup CP_B} (\overrightarrow{q}_i + (1 - \lambda') m_{qi}(\overrightarrow{q}_i, \chi))$

Wait
if $o_{min} < \infty \wedge Wait(P, Q, CP, ti, \lambda, o_{min})$:
$\langle \widetilde{a}, CP, T, ti, \lambda, \delta_t \rangle \rightarrow \langle \widetilde{a}', CP', T', ti, \lambda', \delta_t \rangle$ with:
$(\widetilde{a}_Q, CP_Q) = nf(Q, \widetilde{a} \cup fn(\widetilde{a}, CP))$, $CP' = (move(CP, T, o_{min}) - P) + CP_Q$
$\lambda' = \lambda + o_{min}$, $\widetilde{a}' = clear(\widetilde{a} \cup \widetilde{a}_Q, CP')$
$T' = T - \overrightarrow{t}_{ind(P)} + \bigcup_{Q_i \in CP_Q} (\overrightarrow{q}_i + (1 - \lambda') m_{qi}(\overrightarrow{q}_i, \chi))$

Interval transition
if $o_{min} = \infty$: $\langle \widetilde{a}, CP, T, ti, \lambda, \delta_t \rangle \rightarrow \langle \widetilde{a}, CP', T', ti + 1, 0, \delta_t \rangle$ with:
$CP' = (P_{1\ m_1}^{\overrightarrow{t}_1}, \ldots, P_{k\ m_k}^{\overrightarrow{t}_k})$, with $\overrightarrow{t}_i \in T$, $P_i \in CP$
$T' = (\overrightarrow{t}_1 + m_1(\overrightarrow{t}_1, \chi), \ldots, \overrightarrow{t}_k + m_k(\overrightarrow{t}_k, \chi))$

Nil process
if $P = nil \wedge P \in CP$: $\langle \widetilde{a}, CP, T, ti, \lambda, \delta_t \rangle \rightarrow \langle \widetilde{a}', CP', T', ti, \lambda, \delta_t \rangle$ with:
$CP' = CP - P$, $\widetilde{a}' = clear(\widetilde{a}, CP')$, $T' = T - \overrightarrow{t}_{ind(P)}$

Fig. 3. Operational Semantics of SpacePi. The operators + and - hold up the order of the processes, i.e. they ensure that T and CP are ordered equally.

be initialized as $Q_m^{\overrightarrow{p} + o_{min} \cdot \overrightarrow{t}_{ind(P)}}$.

3 Example: Euglena

Euglena is a micro-organism living in inland water. They perform photosynthesis and thus are dependent on solar irradiation. Therefore, they show a certain behavior which is commonly known as phototaxis. Phototaxis describes motion phenomena that are influenced by sun irradiation. In case of Euglena, the movement is directed toward the sun (water surface) when too little light is perceived, and away from the sun (downward) if it is too much [6]. Our model focuses on this kind of motion. For simplification, we assume a two dimensional top-down slice through a lake of certain width and depth.

To describe motion, we use a vector $\boldsymbol{o}$, a range r, and an angle α. The vector $\boldsymbol{o}$ defines the basic motion orientation relative to the global coordinate system, with its origin being the bottom left corner of the rectangle representing the lake. The range r and the angle α describe the limits of possible variations in speed and direction associated with two probability distributions (see Fig. 4). An expression $P[mov]$ denotes that the movement function mov is assigned to process P.

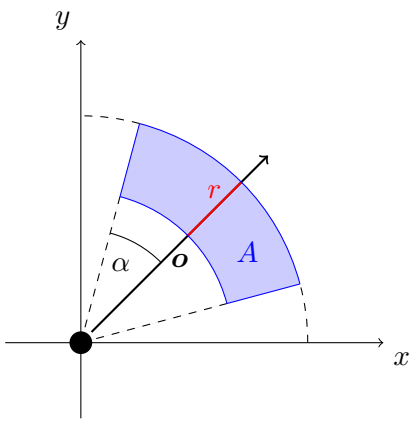

Fig. 4. Stochastic motion - our motion notation is based on an orientation vector $\boldsymbol{o}$ denoting the basic direction and speed of P. A scalar range r shows variations in speed and an angle α defines direction deviations. Assuming that speed is given on a time basis in seconds, A is the area that covers all of P's next possible positions in one second. The probability of P being at a certain point in A is determined by two distributions, one for speed and one for direction variations.

Our model is built as follows. A process named *Sun* generates photons (Fig. 5) which move from the surface to the bottom of the lake (not necessarily reaching it). *Sun* is a non-spatial process, i.e. there is neither a position nor a motion associated with it. *Sun* first waits for a certain amount of time (*prod*). It proceeds by invoking a *Photon* process (producing a photon) and restarting in parallel. Photons perform their motion for a certain time ($wait\langle scatter\rangle$) with *scatter* being equally distributed, which mimics different wavelengths of light. The expression $\overline{phot}\langle\sim, r_p\rangle + wait\langle scatter\rangle$ denotes that *Photon* terminates when its motion time is up or when communication takes place on the spatial channel *phot* due to absorption by Euglena ($\sim$ denotes the empty message).
Euglena's behavior is mapped to three different states denoted by the processes *Euglena*, *EuglenaUp*, and *EuglenaDown*. The first describes the organism when it is staying at a certain depth level waiting for light. Therefore, the process is associated with a non-directional slow motion (*rand*). Two parallel subprocesses, *EuglenaLight* and *EuglenaColl*, share the restricted channel *end*. The first process defines Euglena's reaction on arriving photons, the second detects collisions with other cells. The processes are declared to run in parallel, because neither the inherently random motion nor Euglena's reaction on incoming light is influenced by cell collision. However, *EuglenaColl* is needed for the model to represent collision of randomly moving cells and those that perform directed motion. *EuglenaLight* starts with the expression $wait\langle expect\rangle$. If a photon arrives before *expect* expires, Euglena intends to stay at the current level of depth. Therefore, it proceeds with $EuglenaLight'$ which is defined as $wait\langle deny\rangle.\ EuglenaLight + phot\langle\sim, r_p\rangle.\ \overline{end}.\ EuglenaDown$. It denotes that the arrival of a photon in the period *deny* signalizes a too high light intensity. In this case Euglena would perform a downward motion, i.e. *EuglenaColl* terminates by receiving a signal on *end* and $EuglenaLight'$ proceeds with *EuglenaDown*. *EuglenaColl* has to terminate because in case of *EuglenaDown* and of *EuglenaUp*, collision of Euglena cells is differently processed. By communicating on the restricted channel *end*, the simultaneous termination of $EuglenaLight'$ and *EuglenaColl* is guaranteed. If no photon arrives before *expect* expires, the expression $phot\langle\sim, r_e\rangle.\ end.\ EuglenaUp$ follows: a cell

moves upward because light intensity is too low. Each individual Euglena cell gains the information about upward direction by absorbing light [7], thus *EuglenaLight* waits for a photon to arrive before moving.
EuglenaUp and *EuglenaDown* perform fast directed motions toward and away from the surface. The lasting of these motions is parameterized with a normally distributed value. However, both processes can be interrupted by either a collision with another cell (denoted by the activation of the channel *coll*) or the arrival of a photon. By colliding, their orientation gets lost, so that the cell remains at the current depth level, i.e. they proceed with *Euglena*. The reaction to a photon arrival differs between *EuglenaUp* and *EuglenaDown*. *EuglenaUp* proceeds with *Euglena*, i.e. it tests the current level of depth. *EuglenaDown* restarts such that it moves downward for an additional period.
In SpacePi, each action at a channel might be associated with different radii, in our case we assume $r_e = 0.15$ for actions on *coll* and on *phot* by *Euglena* (representing the approximated size of Euglena) and $r_p = 0.05$ for actions on *phot* by *Photon*.
The movement functions of photons and Euglena cells are described by a vector $\boldsymbol{o}$, a range r, and an angle α. E.g. the upward movement of an Euglena cell *up* is specified by the orientation $\boldsymbol{o} = (0, 0.15)^T$, the speed indicating range $r = [2, 3]$, i.e. Euglena moves with speed twice or thrice its size, and the direction deviation angle $\alpha = 40°$.

As seen above, collision plays a major role in our model. It provides the basis for realizing the absorption of light - micro-organisms can only use photons within in their area of perception. The collision of Euglena cells has an impact on their motion and behavior. For a collision to occur, two processes need to communicate on a spatial channel, thereby the name of the channel, the radius of the complementary channel actions, and the location of the processes are taken into account.

4 SpacePi for modeling biomolecular systems

As our interests preferentially regard the modeling of biomolecular systems, we below discuss the applicability of SpacePi to this field.

4.1 Bindings and decays

SpacePi uses abstractions for binding sites and bonds similar to those described in [22], i.e. common channels for binding sites and restricted channels for bonds. Complex forming processes share one movement function and its results. Also, more sophisticated sorts of complex motion can be considered (see below). In contrast to stochastic π [19], channels in SpacePi are not associated with rates, due to its explicit time basis. To integrate experimental reaction rates in SpacePi, bindings and decays are treated differently. The rates of bindings are determined by defining appropriate channel radii. The derivation of radii from kinetic rates is presented, e.g., in [1]. Decay rates are expressed by *wait* processes that delay decays for probability distributed time amounts. Hence, decays are associated with stochasticity and so are bindings, where stochasticity evolves from process motion with random variations (see Fig. 4).

Process definitions

$Sun = wait\langle prod\rangle.\ (Photon \mid Sun)$
$Photon[downPho] = \overline{phot}\langle\sim, r_p\rangle\ +\ wait\langle scatter\rangle$
$Euglena[rand] = new\ end\ (EuglenaLight \mid EuglenaColl)$
$EuglenaLight = wait\langle expect\rangle.\ phot\langle\sim, r_e\rangle.\ \overline{end}\langle\sim,\infty\rangle.\ EuglenaUp\ +$
$\quad phot\langle\sim, r_e\rangle.\ EuglenaLight'$
$EuglenaLight' = wait\langle deny\rangle.\ EuglenaLight\ +\ phot\langle\sim, r_e\rangle.$
$\quad \overline{end}\langle\sim,\infty\rangle.\ EuglenaDown$
$EuglenaColl = \overline{coll}\langle\sim, r_e\rangle.\ EuglenaColl\ +\ end\langle\sim,\infty\rangle$
$EuglenaUp[up] = (wait\langle moveUp\rangle + \overline{coll}\langle\sim, r_e\rangle + coll\langle\sim, r_e\rangle + phot\langle\sim, r_e\rangle).\ Euglena$
$EuglenaDown[down] = (wait\langle moveDown\rangle + \overline{coll}\langle\sim, r_e\rangle + coll\langle\sim, r_e\rangle).\ Euglena +$
$\quad phot\langle\sim, r_e\rangle.\ EuglenaDown$

Radius declarations

$r_e = 0.15$ (approximated size of Euglena)
$r_p = 0.05$ (size of photon abstractions)

Motion declarations

downPho: $\boldsymbol{o} = (0, -1)^T$, $r = [5, 10]$ (norm), $\alpha = 20°$ (norm)
rand : $\boldsymbol{o} = (0, 0.15)^T$, $r = [0, 1]$ (equal), $\alpha = 360°$ (equal)
up: $\boldsymbol{o} = (0, 0.15)^T$, $r = [2, 3]$ (norm), $\alpha = 40°$ (norm)
down: $\boldsymbol{o} = (0, -0.15)^T$, $r = [2, 3]$ (norm), $\alpha = 40°$ (norm)

Fig. 5. Euglena example - distances in millimeters, time in seconds. Concrete positions are left out, movement functions are written in square brackets behind process names. An expression $c\langle\sim, r\rangle$ denotes that an empty message is to be received on channel c with radius r.

4.2 Molecular motion

Molecular motion is mostly seen as Brownian motion and abstracted as a random walk. Random motions of processes can be easily defined in SpacePi (see sec. 3). Additionally, more sophisticated motions can be described by defining movement functions that consider internal (in case of complexes) and external forces (see sec. 2). Moreover, given a collision channel, molecular crowding effects, which strongly affect signaling pathways (see [26]), can be modeled.
A fast and directed sort of molecular motion is active transportation, e.g. in neuronal cells (see [23]). Transport molecules move fast along intracellular structures (e.g. microtubules) by binding to successive structure parts under energy consumption. In SpacePi, active transportation can be modeled by using processes with fixed positions for intracellular structures (*rail processes*) and mobile processes for transport molecules (*train processes*). When communicating with a *rail process*, the movement function of a *train process* is modified, such that it quickly reaches the next *rail process*. With this model, molecular transportation problems that result from defects of transportation systems could be modeled, e.g. when structural parts

are missing or too little energy is available.

4.3 Membranes and compartments

Membranes are semi-permeable barriers that have major impact on the distribution of intracellular material and thus strongly influence cellular processes (e.g. gene expression). In SpacePi, membranes can be introduced by restricting the motion functions of material representing processes to certain areas and placing non-mobile processes at compartment borders to represent functional membrane parts (e.g. molecular pumps or channels). However, this approach could cause artifacts when modeling membranes varying in size and position. An alternative is to entirely build up membranes with non-mobile processes. Membrane processes share channels with processes that represent non-membrane-passing material. Communication on those channels is used to cause the respective material processes to stay on their current membrane side (semi-permeability). In this manner dynamic compartments with flexible semi-permeable membranes can be modeled, yet, at higher computational costs.

5 Related Work

Different approaches to explicitly integrate space into process algebra exist. CCSG [9] uses space to discriminate process interactions as being relevant or not. Depending on the location and the distance of the processes, it is decided which processes can communicate. This leads to a more efficient analysis of the model as the number of potential communication partners is reduced. No process motion is considered. The Real Space Process Algebra supports the movement of processes [2]. It is inspired by the testing of synchronization protocols in mobile networks. Hence, it is based on broadcasting messages and integrates asynchronous aspects in communication. Furthermore, Real Space Process Algebra assumes that messages propagate in space radially. Whereas this assumption makes sense in the proposed application area, in cell biology it does not seem fitting. An explicit notion of time is crucial in describing motions. Therefore, Real Space Process Algebra makes use of concepts developed in Real Time Process Algebra [29].

Many process algebras deal implicitly with time by introducing rates which define an exponential distribution according to which the probabilistic and dynamic behavior of processes is determined, e.g. stochastic π. However, this implicit concept of time is not sufficient for describing motion, nor is it for some other dynamic phenomena.

Quite a few process algebra have introduced explicit timing [3], concepts that are also exploited in our approach. Another project which is interesting in our context is Kiltera [17]. Following the argumentation line in [28], it merges the benefits of process algebras to deal with dynamic topologies of networks with an explicit representation of time and scheduling of events inspired by approaches like Devs [30]. However, those approaches only integrate time and not space.

Several simulation packages exist that support an individual-based view of the system with spatial dynamics [1,25,16]. Those systems do not distinguish between a formal modeling and the execution of the model. The SpacePi formalism has the

major advantage of being a process algebra: it can be analyzed mathematically and the modeler is free to choose any abstraction level that is deemed appropriate. For example, the movement functions of the processes could be expressed as complex functions, taking all kinds of external forces into account. On the other hand, it could be sufficient to assume random motion for certain models. All these assumptions have to be explicitly formulated when describing the model, which prevents misunderstandings and may lead to additional insights. In contrast, the cited simulation systems presuppose a certain level of abstraction, ranging from molecular dynamics approaches to certain approximations. Since SpacePi allows to switch easily between different underlying assumptions (e.g., by simply editing the definition of a movement function), it can be used to examine the effects that are imposed by simplifying abstractions regarding particle movement, and the artifacts that might stem from them.

6 Conclusion and Outlook

In this paper, we introduced an extension to the π calculus. It associates processes with coordinates in a vector space and individual movement functions. Communication between processes is based on channel names, the radii of the channel actions, and the positions of the processes. As illustrated by the exemplary model of Euglena's phototaxis, the formalism allows a concise and rigorous description of spatial phenomena. By combining an individual-based perception and absolute space, not only diffusion processes but also active transportation and membrane micro-domains can be modeled realistically at different resolutions: features that we exploit in a modeling project in cooperation with wet-lab partners. Models are being developed to analyze the role of Wnt signaling mediated changes in the cytoskeleton motor enzyme system and their implications for the differentiation of neural progenitor cells.

A central prerequisite for modeling individually moving processes in a vector space is a notion of explicit time. This makes our approach different from other π calculus extensions, like stochastic π. It also implies that existing simulation engines based on Gillespie's approach, e.g. [15], cannot be used.

A simulation engine for SpacePi has been developed. The validation of the program and its application to the research issues of our wet-lab partners are under way. The prototype is implemented along the lines of SpacePi's formal semantics (see sec. 2.2). However, there are several implementation aspects that deserve closer consideration.

Firstly, for each pair of processes that announce to communicate over the same channel it has to be determined if their channel radii overlap. Although we only deal with circular shapes in 2D space (or spherical in 3D), applying a straightforward algorithm to this task would be expensive and thus slow down simulation. This is a problem that has been widely referred to in the context of collision detection as known from computer graphics [11]. Proposed solutions exploit the fact that many objects of a scene simply do not qualify for collision because of their position and size (geometrical coherence) or their motion (temporal coherence). Thus, many collision checks can be avoided, which leads to more efficient calculations.

Secondly, as we expect a SpacePi simulation to cost more computing power than the simulation of, e.g., stochastic π models, a parallel and distributed execution might be a desirable option. One could partition the space to subspaces, distribute these over a set of processors and then exploit the parallelism of processes that are moving and communicating locally and in concurrence. Obviously, a parallel and distributed SpacePi simulator will have to cope with synchronization issues when a process is moving from one subspace to another. Still, given that the channel radii are comparatively small with respect to the overall space, specialized partitioning methods that take the movement patterns of processes into account (e.g., to partition along membranes) could alleviate this problem and speed up a parallel execution.

References

[1] Andrews, S. S. and D. Bray, *Stochastic simulation of chemical reactions with spatial resolution and single molecule detail*, Phys. Biol. **1** (2004), pp. 137–151.

[2] Baeten, J. C. M. and J. A. Bergstra, *Real space process algebra*, Formal Aspects of Computing **5** (1993), pp. 481–529.
URL http://dx.doi.org/10.1007/BF01211247

[3] Banach, R., *Review: Process Algebra with Timing*, Journal of Logic and Computation **14**, pp. 881+.

[4] Cardelli, L., *Membrane interactions*, in: *BioConcur 03, Workshop on Concurrent Models in Molecular Biology*, 2003.

[5] Deutsch, A. and S. Dormann, "Cellular Automaton Modeling of Biological Pattern Formation," Birkhuser Boston, 2004.

[6] Engelmann, T., *Ueber Licht- und Farbenperception niederster Organismen*, Pflügers Archiv European Journal of Physiology **29** (1882), pp. 387–400.
URL http://dx.doi.org/10.1007/BF01612047

[7] Grell, K. G., "Protozoologie," Springer-Verlag, 1968.

[8] Haugh, J. M., *A Unified Model for Signal Transduction Reactions in Cellular Membranes*, Biophys. J. **82** (2002), pp. 591–604.

[9] Isobe, Y., Y. Sato and K. Ohmaki, *Approximative Analysis by Process Algebra with Graded Spatial Actions*, in: *Proc. AMAST '96* (1996), pp. 336–350.

[10] Kholodenko, B., *Cell-signalling dynamics in time and space*, Nature Reviews Molecular Cell Biology **7** (2006), pp. 165–176.

[11] Lin, M. and S. Gottschalk, *Collision detection between geometric models: A survey*, in: *Proc. of IMA Conf. on Mathematics of Surfaces*, 1998.
URL http://citeseer.ist.psu.edu/lin98collision.html

[12] Mayawala, K., D. G. Vlachos and J. S. Edwards, *Computational modeling reveals molecular details of epidermal growth factor binding.*, BMC Cell Biol **6** (2005).
URL http://dx.doi.org/10.1186/1471-2121-6-41

[13] Milner, R., "Communicating and Mobile Systems: the Pi-Calculus," Cambridge University Press, 1999.

[14] Nicollin, X. and J. Sifakis, *An Overview and Synthesis on Timed Process Algebras*, in: *Proc. CAV '91* (1992), pp. 376–398.
URL http://portal.acm.org/citation.cfm?id=735193

[15] Phillips, A. and L. Cardelli, *A correct abstract machine for the stochastic pi-calculus*, Transactions on Computational Systems Biology (2005).

[16] Plimpton, S. J. and A. Slepoy, *Microbial cell modeling via reacting diffusive particles*, Journal of Physics: Conference Series **16** (2005), pp. 305+.
URL http://dx.doi.org/10.1088/1742-6596/16/1/042

[17] Posse, E. and H. Vangheluwe, *Kiltera: A Simulation Language for Timed, Dynamic Structure Systems*, in: *ANSS '07. 40th Annual Simulation Symposium.*

[18] Prandi, D., C. Priami and P. Quaglia, *Process calculi in a biological context*, Concurrency Column (2005).

[19] Priami, C., *Stochastic pi-Calculus*, The Computer Journal **38** (1995), pp. 578–589.
URL http://dx.doi.org/10.1093/comjnl/38.7.578

[20] Priami, C. and P. Quaglia, *Beta Binders for Biological Interactions*, in: *Computational Methods in Systems Biology*, 2005, pp. 20–33.
URL http://dx.doi.org/10.1007/b107287

[21] Regev, A., E. M. Panina, W. Silverman, L. Cardelli and E. Shapiro, *BioAmbients: an abstraction for biological compartments*, Theor. Comput. Sci. **325** (2004), pp. 141–167.
URL http://dx.doi.org/10.1016/j.tcs.2004.03.061

[22] Regev, A. and E. Shapiro, *The π-calculus as an abstraction for biomolecular systems*, in: G. Ciobanu, Gabriel Rozenberg, editor, *Modeling in Molecular Biology*, Springer, 2004 .
URL http://citeseer.ist.psu.edu/704367.html

[23] Schwartz, J. H., *Axonal transport: components, mechanisms, and specificity.*, Annu Rev Neurosci **2** (1979), pp. 467–504.
URL http://dx.doi.org/10.1146/annurev.ne.02.030179.002343

[24] Smolen, P., D. A. Baxter and J. H. Byrne, *Modeling Circadian Oscillations with Interlocking Positive and Negative Feedback Loops*, J. Neurosci. **21** (2001), pp. 6644–6656.

[25] Stiles, J. and T. Bartol, *Monte Carlo methods for simulating realistic synaptic microphysiology using MCell*, Computational Neuroscience: Realistic Modeling for Experimentalists (2001), pp. 87–127.

[26] Takahashi, K., S. Nanda, V. Arjunan and M. Tomita, *Space in systems biology of signaling pathways : towards intracellular molecular crowding in silico*, FEBS letters **579** (2005), pp. 1783–1788.

[27] Tolle, D. P. and N. Le Novre, *Particle-Based Stochastic Simulation in Systems Biology*, Current Bioinformatics **1** (2006), pp. 1–6.

[28] Uhrmacher, A. M. and C. Priami, *Discrete event systems specification in systems biology - a discussion of stochastic pi calculus and DEVS*, in: *WSC '05: Proc. of the 37th conference on Winter simulation*, 2005, pp. 317–326.
URL http://portal.acm.org/citation.cfm?id=1162708.1162767

[29] Wang, Y., *The Real-Time Process Algebra (RTPA)*, Annals of Software Engineering **14** (2002), pp. 235–274.

[30] Zeigler, B. P., H. Praehofer and T. G. Kim, "Theory of Modeling and Simulation, Second Edition," Academic Press, 2000.