Modeling and Analysis of Spatio-Temporal Processes in Biomolecular Systems

Andreas Schäfer, Mathias John

Modeling and Analysis of Spatio-Temporal Processes in Biomolecular Systems

Andreas Schäfer[1] **Mathias John**[2]

[1] Department of Computing Science, University of Oldenburg, Oldenburg, Germany
Email: schaefer@informatik.uni-oldenburg.de

[2] Department of Computing Science, University of Rostock, Rostock, Germany
Email: mjohn@informatik.uni-rostock.de

Abstract

In life science, deeper understanding of biomolecular systems is acquired by computational modeling and analysis. For the modeling of several kinds of reaction networks, e.g. signaling pathways, information on intracellular space, like the locations and motions of molecules, has to be taken into account. In this paper, we introduce Labeled SpacePi, an extension of the π-calculus, in order to model spatio-temporal processes in cells. The formalism is tailored to the available data and knowledge about biomolecular systems. For the analysis, we employ model checking techniques known from the field of safety-critical systems. To this end, we develop a translation of Labeled SpacePi models into hybrid automata. Two use cases - one considering the activation of a signaling pathway and the other one concerning active transport in cells - demonstrate our concept by making use of the established analysis tools HyTech and HySat.

Keywords: conceptional modeling, spatio-temporal modeling, biological data, π-calculus, hybrid systems, model checking

1 Introduction

In the context of life science, intracellular processes on a molecular level become of increasing interest, e.g. (Polakis 2007, Thompson 1995). However, even with the help of modern wet-lab techniques, a complete understanding of biomolecular systems is hard to obtain. Therefore, biologists are supported on their investigations by deploying methods of computational modeling. To this end, the modeling needs to take spatial information into account, because several processes in human, i.e. eukaryotic, cells are strongly dependent on the locations of molecules (Kholodenko 2006). This is mainly due to the fact, that eukaryotic cells are not just simple containers, but complex structures that are crowded with molecules (Takahashi et al. 2005), see Figure 1. The subject of this work is to present a modeling concept, that allows for the investigation of spatial effects in biomolecular systems. As a first contribution, based on the process algebra SpacePi (John, Ewald & Uhrmacher 2008), we define a modeling formalism that can take various kinds of spatial data into account but also avoids to require data that is hard or impossible to collect - an essential point for a modeling approach, which is both, meaningful and practical. We extend the SpacePi semantics by labels similar to the labels known from hybrid logics (Areces & ten Cate 2006). Therefore, we call our formalism Labeled SpacePi. The π-calculus allows for recursive definition of multiple parallel processes and their possible interaction. In our setting, we consider biological entities like cell or molecules for which we use the general term agent. We use the constructs from the π-calculus to specify their behavior and interaction. So, for each agent we obtain a process term that can be arbitrarily complex. The whole model of the biological system itself is also a big process term, i.e. the parallel composition of all the terms for the agents which can involve an arbitrary number of recursive definitions and calls. The possible interaction of the agents depends on the spatial configuration, like for example whether a neighboring agent is within a reaction distance. Therefore we need to be able to specify which subprocess of the process for the whole model belongs to which agent and keep track of this information throughout all recursive calls. To this end, we use the labels for grouping and syntactically identifying subprocesses of a system that constitute individual agents. We furthermore let the labels carry the information on position and movement of the agent. This allows for a clearer and more concise presentation than the original semantics. We develop a formal verification technique for a subset of Labeled SpacePi. It considers a system's entire set of possible evaluations and proceeds by constructing hybrid system models (Alur et al. 1992). As it is a domain of active research, we profit from the results achieved in the field of verification of hybrid systems and can make use of the existing tools. In this paper, we employ the bounded verification tool HySAT (Fränzle et al. 2007) and the classical tool HyTech (Henzinger et al. 2001). Finally, two use cases are given, that illustrate the application of Labeled SpacePi to the

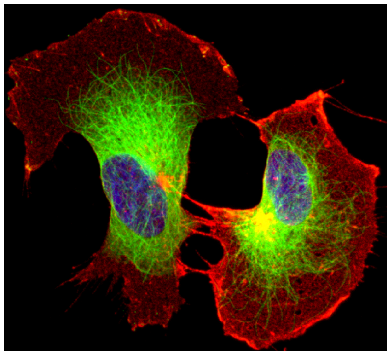

Figure 1: A neuronal cell under the fluorescence microscope: blue marks the cell core (nucleus), red a species called β-catenin, a basic substance in neuronal cells, and green microtubles, i.e. the skeleton of the cell. Courtesy of Benjamin Bader, University of Rostock.

We thank Orianne Mazemondet for her support on the biological background. This research was partially supported by the DFG in the context of the Research Training School "dIEM oSiRiS".

spatial modeling of intracellular processes. One focuses on the activation of the Wnt signaling pathway (Polakis 2007) and the other on active transport in cells.
For the spatial modeling of biomolecular systems, input data is mainly provided in the form of molecule locations, i.e. the spatial distribution of the different protein sorts, including intra-cellular structures, like membranes or microtubules. In this context, the image analysis technique presented in (Zhao & Murphy 2007) recently gained much attention. Based on a set of microscopic images, it computes intracellular maps, i.e. spatial models representing sets of observed cells. They deploy image processing techniques (segmentation) to automatically obtain properties of cells, which are used to generate clusters. The cluster representatives form the actual spatial models. Another technique, stemming from the field of microscopy, called *Fluorescence Recovery After Photobleaching* (FRAP) (Meyvis et al. 1999), allows to obtain diffusion constants that describe the spreading of molecules. This is done by tagging the molecule of interest with fluorescent agents, bleaching a circular area with a laser beam of given radius, and measuring the time for Fluorescence to recover in that area. Databases provide the volumes of many molecules, e.g. (Letunic et al. 2006), and different tools exist for predicting the three-dimensional structure of proteins, e.g. (Zuker 2003, Rivas & Eddy 1999). Thus, a detailed image of intracellular space can be drawn by combining data of different sources. By contrast, only little information is given about the interaction of proteins, i.e. their logical order is mostly only assumed. Quantitative descriptions in the form of rate constants are rarely given, because reaction constants are hardly observable in experiments and although important for the modeling only of marginal interest for the biologists. Therefore, projects are planned that shall investigate the deployment of computational methods to determine rate constants (Takahashi et al. 2003).
The desired results of our modeling are statements about the temporal development of the spatial distribution of molecules. In particular, it is the goal to check, if, given some spatial topology, initial distribution of molecules, and logical order of reactions, some observed spatial distribution of molecules can occur at a certain point in time. By this means, existing hypotheses about the system under study can be evaluated. Additionally, it shall be possible to approximate earliest time points of specific events, e.g. a certain number of molecules reaches some location, and thus to suggest new experimental settings.
Our modeling describes molecules as individuals of certain size and shape with some starting position in real space and some motion function. Reactions are represented by interactions of individuals resulting in new individuals, see Sec. 2. They occur whenever interaction partners are sufficiently close. It is also possible to define sets of non-interacting molecules of the same sort (multiplicities). In this way, model complexity can be reduced, such that the analysis process is less computational costly, an important point regarding practicability, see Sec. 6. In order to introduce obstacles that interfere with molecular motion, e.g. membranes, interactions can be marked as urgent, i.e. they are performed as soon as possible. By contrast, all other interactions can occur but do not necessarily need to. This allows for the approximation of stochasticity as e.g. the fact, that not every collision of two molecules is leading to a reaction.
The paper is structured as follows: in Sec. 2, the syntax and semantics of Labeled SpacePi is introduced, including the concepts of obstacles and multiplicities. Sec. 3 presents the model analysis approach that makes use of a translation of Labeled SpacePi into hybrid automata. In order to relate Labeled SpacePi processes to the constructed hybrid automata, a spatio-temporal bisimilarity is defined. Furthermore, the Sec. contains a short introduction to the verification tools HyTech and HySAT. These are used in Sec. 4 for the analysis of the exemplary models. In Sec. 6 an overview about related work is given and Sec. 7 concludes the paper.

2 Labeled SpacePi

Our modeling formalism is based on SpacePi, which is itself a derivative of the π-calculus (Milner 1999). Therefore, it adopts the ideas presented in (Regev & Shapiro 2002) for describing biomolecular systems. Molecules are represented as concurrent processes and reactions as communication channels, on which processes can synchronize. Since communication in the π-calculus is always performed by one sender and one receiver, reactions are restricted to two reactants. By contrast, there is no maximum number of products, since after communication, a process can proceed with any number of concurrent processes. Sending and receiving on channel x is denoted by $\overline{x}\langle\rangle$ and $x()$. It is also possible to send and receive channels, see Section 2.1. Concurrent processes are described as $P_1 \mid P_2$. In general, a reaction network

$$r_1 : R_1 + R_3 \Rightarrow P_1 + \cdots + P_n$$
$$r_2 : R_2 + R_3 \Rightarrow Q_1 + \cdots + Q_n$$

can be described with three processes

$$R_1 = \overline{r_1}\langle\rangle.(P_1|\ldots|P_n), R_2 = \overline{r_2}\langle\rangle.(Q_1|\ldots|Q_n)$$
$$R_3 = r_1() + r_2()$$

where $+$ denotes the choice of a process to participate in one of two communications and $P.Q$ means that P proceeds with Q. Notice, that the mapping from reaction networks to π-calculus models is not unambiguous.
SpacePi extends the π-calculus, such that processes can be associated with real space positions in two (or more) dimensions. Movement functions describe the motion of processes. Constraints can be defined on communications, such that processes can only synchronize if they are sufficiently close. Figure 2 shows a small example: Two processes C and S are located at some initial position (v_0). The movement function of C is defined to be circular. Its radius f_c describes the possible positions of C in a subsequent time step. Similarly, S can locate along f_s. C and S can get close enough in order to communicate on channel x (v_1). Alternatively, C can also move without communicating until it changes its movement function (v_1').

In this section, we refine SpacePi, such that processes are now associated with shapes and have the ability to transmit positions and movement functions - a helpful feature as shown in Section 4.2. Additionally, a new, more concise, semantics is given, which makes use of labels as known from hybrid logics. Finally, concepts are introduced to describe membranes (obstacles) and to represent molecule sets as single individuals (multiplicities).

2.1 Syntax

The π-calculus employs the alphabet $\mathcal{N}_c$ of *channel names*. Channels can be used for communication and also be communicated over other channels. As

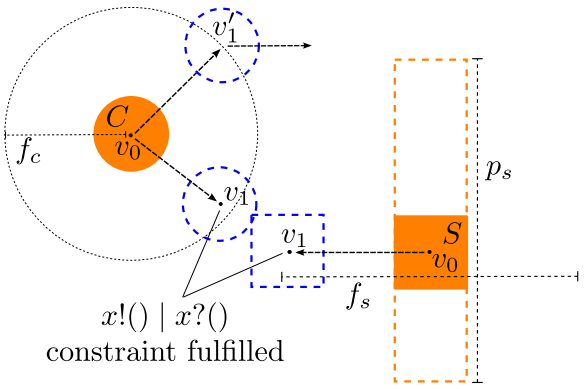

Figure 2: Two processes C and S moving in 2D space. p_s describes the possible positions of S at configuration v_0. f_c and f_s represent the possible future locations of the processes, respectively. v_1 defines a subsequent configuration, where C and S can communicate via x, as the corresponding constraint is fulfilled. v_1' describes an alternative to v_1, where C changes its movement.

an extension, Labeled SpacePi additionally incorporates the alphabets $\mathcal{N}_f$ of *symbols of movement constraints*, $\mathcal{N}_p$ of *symbols of position constraints*, and $\mathcal{N}_s$ of *symbols of size constraints* in order to assign spatial parameters, i.e. positions, sizes, and movements, to processes. The symbols in these alphabets are interpreted by an *interpretation* ι which assigns a predicate to each of the constraints. Positions and sizes are predicates over coordinates in $\mathcal{R}^n$ and movements are predicates over coordinates and their derivatives.

Example 1. Consider the agent S from Figure 2. The predicate $\iota(f_s) = -1 \leq \dot{x}^2 \leq 1 \wedge \dot{y}^2 = 0$ specifies that S moves with maximal velocity 1 in x direction forward or backward and does not move in y direction.

Two agents can communicate over a channel if their distance is lower than or equal the real number specified by ι for this channel. To identify processes, we use an alphabet $\mathcal{N}_i$ of *identification labels*. The union of all alphabets, that are assumed to be disjoint, is called $\mathcal{N}$, the alphabet of *names*. Processes can exchange names of all these types.
Technically, we need to identify, which part of a process definition is considered to form an *agent* with an own identity, initial position, movement and size and also to assign the corresponding spatial parameters to it. To this end, we adopt the idea of labels, which stems from the field of hybrid logics (Areces & ten Cate 2006). The scope of the label defines, which subprocesses "belong together" and the label itself contains the corresponding parameters. We now define the syntax and semantics of Labeled SpacePi. We first consider that one agent represents a single biological entity. Subsequently, we show how to extend the model such that one agent can be interpreted as a representative of a set of equivalent biological entities.

Definition 1 (Syntax)**.** The set of Labeled SpacePi processes is defined by

$$\begin{aligned} P ::= &\sum \pi_i.P_i \mid P_1 \mid P_2 \mid \nu a : \mathcal{E}.P \mid \\ &\alpha_i(i).P \mid \alpha_p(p).P \mid \alpha_f(f).P \mid \alpha_s(s).P \mid A(\tilde{a}) \mid \\ &m : P \text{ if P does not contain a label} \end{aligned}$$

where π_i is an action, $a \in \mathcal{N}$ is a name and $\mathcal{E}$ is a boolean combination of (in)-equalities for defining the interpretation. The names $p \in \mathcal{N}_p$, $f \in \mathcal{N}_f$, $s \in \mathcal{N}_s$ are symbols for a position, movement, and size constraint, respectively. A is a process identifier, and $\tilde{a}$ a vector of names. We assume that for each A there is exactly one defining equation $A(\tilde{a}) \triangleq P_A$ with the corresponding number of parameters. An *action* π_i can be of one of the three following forms: $\overline{x}\langle y \rangle$, i.e. send y over channel x, $x(y)$, i.e. receive y over channel x, or τ_q, the silent timed action with $q \in \mathbb{Q} \cup \{\infty\}$ indicating a delay of q time units. The ν operator is used to create new names, i.e. *bound* names. Bound names are subject to alpha conversion, such that it is not possible to fix the interpretation ι for symbols in beforehand. Therefore, we allow to put a defining (in)-equality $\mathcal{E}$. For channels $\mathcal{E}$ is a distance $\in \mathbb{R}^+$, for positions and sizes a set of predicates over coordinates $\in \mathbb{R}^n$, and for movements a set of predicates over coordinates and their derivatives. As labels represent the identity of agents and the three spatial parameters, we assume that they are 4-tuples having the form $m = \left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]$ where $i \in \mathcal{N}_i$ is an identifier, and $f \in \mathcal{N}_f$, $p \in \mathcal{N}_l$, $s \in \mathcal{N}_s$ are predicate symbols for the three spatial parameters. To increase intelligibility, a label is denoted by a symbol m, if we do not refer to its components. The components of a label are modified using the apply operators α. E.g. the process $A \triangleq m : \nu i \in \mathcal{N}_i.\alpha_i(i).\alpha_f(f).B \mid A$ creates a new identifier i and assigns it to $\alpha_f(f).B$. The $\alpha_f(f)$ action then modifies the movement of the new agent B. Due to the recursive definition, the process A can create an unbounded number of new agents, each having a movement function defined by f.

2.2 Semantics

The reduction semantics of a π-calculus process is defined by a transition system, where each node is a process term and each transition a possible evolution. The definition is usually given in terms of reduction rules. We adopt this technique. However, in Labeled SpacePi, nodes not only contain process terms but also information on the interpretation ι of the predicate symbols from the alphabets $\mathcal{N}_f$, $\mathcal{N}_c$, and $\mathcal{N}_s$, the current position for each label, and a clock for checking timeouts. In the two dimensional case, we write the interpretation ι for position and size as an inequality over the variables x and y and for the movement as an inequality over the variables $x, y, \dot{x}$, and $\dot{y}$. The dotted variables represent derivatives. We further write $(a, b) \in \iota(p)$ to denote that the pair (a, b) satisfies the predicate $\iota(p)$. $\iota(p)(a, b)$ refers to the defining equality of $\iota(p)$ in which the free variables are substituted by a and b.

Example 2. Let $\left[\begin{smallmatrix} i_c & f_c \\ p_c & s_c \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} i_s & f_s \\ p_s & s_s \end{smallmatrix}\right]$ be the labels of the agents C and S in Figure 2. Then, to specify that C moves with a maximal velocity of 2, we use an interpretation $\iota(f_c) = \dot{x}^2 + \dot{y}^2 \leq 2$. With this definition C has the nondeterministic choice in which direction to move which is illustrated by the two dashed circles in Figure 2 representing two possible evolutions. The property that C is a disc with radius 1 is further defined by $\iota(s_c) = x^2 + y^2 \leq 1$. Similarly, the interpretation $\iota(s_s) = -1 \leq x \leq 1 \wedge -1 \leq y \leq -1$ specifies that the agent S is a square of length 2. An example of the specification of the initial position is a predicate $\iota(p_s) = x = 6 \wedge 0 \leq y \leq 6$ fixing the x position while allowing a nondeterministic choice over the y position in a given range as sketched in Figure 2 by the rectangle surrounding S.

For each label m, we introduce three variables: a timeout clock c_m for the τ_t action that is reset after every reaction in which the agent is involved and two variables x_m, y_m that represent a reference point for

the shape of the process that is identified by the label m. The states of the transition system are composed of a labeled process term P, an interpretation ι, and a real valued valuation v of the clock and the position variables. We denote by $v[c := a]$ the valuation that coincides with v except that c is reset to the value a.

Similarly to the timed automata semantics (Bengtsson & Yi 2003, Alur & Dill 1994), we distinguish two types of transitions. *Delay transitions* $\xrightarrow{\delta}$ model the elapsing of δ time units. *Actions transitions* $\rightarrow$ arise from communication and other reductions and do not consume time. They correspond to the reductions of the π-calculus.

The π-calculus uses structural congruence rules for the semantics definition. To handle the labels, we extend the structural congruence to let the labels distribute over parallel composition and restriction.

$$m : (P_1 \mid P_2) \equiv m : P_1 \mid m : P_2$$
$$\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right] : \nu x.P \equiv \nu x.\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right] : P \text{ if } x \notin \{i, f, p, s\}$$

Furthermore, we modify the rule for alpha conversion and allow a bound name to be alpha converted to a name of the same alphabet.

We now define the transition rules for Labeled SpacePi. Every process can perform a delay transition where δ time units elapse, the clock values are increased by δ, and the position changes according to the predicate. This is captured by the following rule.

$$(P, \iota, v) \xrightarrow{\delta} (P, \iota, v') \qquad \text{(DELAY)}$$

if v' satisfies to following conditions:

1. $v'(c) = v(c) + \delta$ for all clocks c,
2. for all positions $x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}, y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}$ used in labels occurring in P there are continuous differentiable functions $\phi^x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}[0, \delta] \rightarrow \mathbb{R}$ and $\phi^y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}[0, \delta] \rightarrow \mathbb{R}$ such that the function $(\phi^x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}, \phi^y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}) + (v(x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}), v(y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}))$ satisfies the predicate $\iota(f)$ for all points in $(0, \delta)$ and $(v'(x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}), v'(y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]})) = (\phi^x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}(\delta), \phi^y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}(\delta)) + (v(x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}), v(y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}))$.

Intuitively, the function $(\phi^x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}, \phi^y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]})$ can be seen as one of the arrows depicting the actual trajectory of the agent in Figure 2.

Example 3. Assume that process $m_s : S$ is at position $(6, 2)$, i.e. $v(x_{m_s}) = 6$ and $v(y_{m_s}) = 2$, the clock has the value $v(c_m) = 3$ and the predicate for the movement is $-1 \leq \dot{x} \leq 1 \wedge \dot{y} = 0$. If time progresses by 2 time units, denoted by a delay transition $(m_s : S, \iota, v) \xrightarrow{2} (m_s : S, \iota, v')$, the process can arrive at position $(4, 2)$, i.e., $v'(x_m) = 4$ and $v'(y_m) = 2$ and the clock has value $v'(c_m) = 5$.

A process $m : \tau_5.P$ waits 5 time units and can then perform an action transition, such that it evolves to P without consuming time. We use the clock valuation $v(c_m)$ to determine the elapsed waiting time of the process $m : \tau_t.P$. This is correct, as the clock is reset after every reaction involving the process identified by m and leads to the following rule:

$$(m : \tau_t.P + M, \iota, v) \rightarrow (m : P, \iota, v[c_m := 0]) \quad \text{if } v(c_m) = t \qquad \text{(TAU)}$$

Two sums labeled by $m_1 = \left[\begin{smallmatrix} i_1 & f_1 \\ p_1 & s_1 \end{smallmatrix}\right]$ and $m_2 = \left[\begin{smallmatrix} i_2 & f_2 \\ p_2 & s_2 \end{smallmatrix}\right]$ can communicate over a common channel, if the distance between the represented agents is equal or below the channel's distance. The set of points that are covered by the agent labeled m_1 is the set of points satisfying the size predicate $\iota(s_1)$ translated by the current position of the reference point $(v(x_{m_1}), v(y_{m_2}))$. The set of points covered by agent m_2 is defined analogously. The reaction rule reads:

$$(m_1 : \overline{x}\langle y\rangle.P_1 + N_1 \mid m_2 : x(z).P_2 + N_2, \iota, v) \rightarrow (m_1 : P_1 \mid m_2 : P_2\{^y/_z\}, \iota, v') \qquad \text{(REACT)}$$

if $\exists\, (\tilde{x}_1, \tilde{y}_1) : \exists\, (\tilde{x}_2, \tilde{y}_2) : (\tilde{x}_1, \tilde{y}_1) \in \iota(s_1)$ and $(\tilde{x}_2, \tilde{y}_2) \in \iota(s_2)$ and $\|(\tilde{x}_1, \tilde{y}_1) + (v(x_{m_1}), v(y_{m_1})) - (\tilde{x}_2, \tilde{y}_2) + (v(x_{m_2}), v(y_{m_2}))\| \leq \iota(x)$ and where $v' = v[c_{m_1} := 0, c_{m_2} := 0]$.

The condition formalizes the requirement that the first and the second agent each covers one of two points that have an euclidean distance that equal or below the channel distance. Additionally, it resets the corresponding clock after communication.

Example 4. Consider two processes with the shape of a square of size 1, i.e., $\iota(s) = 0 \leq x \leq 1 \wedge 0 \leq y \leq 1$. Let the lower left corner of $m_1 : \overline{a}\langle b\rangle.A$ be at $v(x_{m_1}, y_{m_1}) = (0, 2)$ and of $m_2 : \overline{a}\langle c\rangle.B$ be at $v(x_{m_2}, y_{m_2}) = (0, 3)$ and let the reaction distance $\iota(a)$ of the channel a be zero. Then both processes can react to $m_1 : A \| m_2 : B\{^b/_c\}$ because they share the point $(0, 3)$.

The α operators modify the label while performing an action transition. This cannot be captured by structural congruence as the clocks and positions need to be updated accordingly. At first, we define the $\alpha_f()$ operator for setting the movement.

$$(\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right] : \alpha_f(f').P, \iota, v) \rightarrow (\left[\begin{smallmatrix} i & f' \\ p & s \end{smallmatrix}\right] : P, \iota, v') \qquad \text{(APPL-F)}$$

where $v' = v[c_{\left[\begin{smallmatrix} i & f' \\ p & s \end{smallmatrix}\right]} := 0, x_{\left[\begin{smallmatrix} i & f' \\ p & s \end{smallmatrix}\right]} := x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}, y_{\left[\begin{smallmatrix} i & f' \\ p & s \end{smallmatrix}\right]} := y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}]$, i.e., clocks are reset and the position of the process is copied and does not change. The rule for the change of the identifier is similar:

$$(\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right] : \alpha_i(i').P, \iota, v) \rightarrow (\left[\begin{smallmatrix} i' & f \\ p & s \end{smallmatrix}\right] : P, \iota, v') \qquad \text{(APPL-I)}$$

where $v' = v[c_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]} := 0, x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]} := x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}, y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]} := y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}]$. The rule for the application of a new position $\alpha_p(\cdot)$ ensures that the new position satisfies the predicate $\iota(p')$:

$$(\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right] : \alpha_p(p').P, \iota, v) \rightarrow (\left[\begin{smallmatrix} i & f \\ p' & s \end{smallmatrix}\right] : P, \iota, v') \qquad \text{(APPL-P)}$$

where $v'(c_{\left[\begin{smallmatrix} m & f \\ p' & s \end{smallmatrix}\right]}) = 0$, $(v(x_{\left[\begin{smallmatrix} m & f \\ p' & s \end{smallmatrix}\right]}), v(y_{\left[\begin{smallmatrix} m & f \\ p' & s \end{smallmatrix}\right]})) \in \iota(p')$ and v and v' coincide on all other values. The $\alpha_s(\cdot)$ operator modifies the size component of the label, resets the clocks and copies the position:

$$(\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right] : \alpha_s(s').P, \iota, v) \rightarrow (\left[\begin{smallmatrix} i & f \\ p & s' \end{smallmatrix}\right] : P, \iota, v') \qquad \text{(APPL-S)}$$

where $v' = v[c_{\left[\begin{smallmatrix} i & f \\ p & s' \end{smallmatrix}\right]} := 0, x_{\left[\begin{smallmatrix} i & f \\ p & s' \end{smallmatrix}\right]} := x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}, y_{\left[\begin{smallmatrix} i & f \\ p & s' \end{smallmatrix}\right]} := y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}]$.

The semantics of the ν operator corresponds to the one of the π-calculus. However, in addition, Labeled SpacePi also allows for the creation of new symbols for position, movement or size constraints. These symbols are interpreted by the function ι. Concerning this intuition, the ν operator acts as an existential quantifier, where the interpretation of the quantified

name can change. This is captured by the following rule:

$$\frac{(m_1 : P, \iota, v) \to (m_2 : P', \iota, v')}{(\nu x : \mathcal{E}.m_1 : P, \iota', v) \to (\nu x : \mathcal{E}.m_2 : P', \iota', v)} \quad \text{(RES)}$$

where ι and ι' coincide on all values except the value of x and the value of x satisfies $\mathcal{E}$. For the definition of the parallel composition rule, we have to consider the case in which the domain of the valuation $\tilde{v}$ of the composite process is larger than the domain of the valuation v of a component:

$$\frac{(m_1 : P, \iota, v) \to (m_2 : P', \iota, v')}{(m_1 : P \mid m_3 : Q, \iota, \tilde{v}) \to (m_2 : P' \mid m_3 : Q, \iota, \tilde{v}')} \quad \text{(PAR)}$$

where v coincides with $\tilde{v}$ on all values from the domain of v and v' coincides with $\tilde{v}'$ on all values from the domain of v'.

The last group of rules directly corresponds to the original reduction rules of the π-calculus.

$$(m : A(\tilde{z}), \iota, v) \to (m : P_A \left\{ {}^{\tilde{z}}/_{\tilde{x}} \right\}, \iota, v[c_m := 0]) \quad \text{if } A(\tilde{x}) \triangleq P_A \quad \text{(CALL)}$$

$$\frac{(m_1 : P, \iota, v) \to (m_2 : P', \iota, v'), Q \equiv P, Q' \equiv P'}{(\nu x.m_1 : Q, \iota, v) \to (\nu x.m_2 : Q', \iota, v')} \quad \text{(STRUCT)}$$

Remark 2 (Multiple dimensions)**.** Although the semantics is presented for the two dimensional case, its generalization to more dimensions is straightforward by adding variables and extending the valuation v.

2.3 Extensions

Obstacles When modeling biological phenomena, the need arises to specify obstacles, which influence and restrict the movement of agents. The first solution is to impose invariants on movement functions, e.g. by specifying that a movement in x-direction does not exceed a given threshold. However, this does not allow for the representation of moving obstacles. A different possibility is to model the obstacles by agents that send a modified movement function to the moving agents. However, this requires the transmission of the new movement function and its application to occur as soon as possible. Therefore, we extend the model by the urgent transitions that must fire on activation. In particular, we use the concept of urgent channels as known from timed automata (Bengtsson & Yi 2003). For the definition of their semantics, we employ *urgent transitions* $\to_u$ as a third type of transitions relation. The rules REACT and APPL are modified to yield urgent transitions and the composition rules are generalized accordingly.

A delay transition can only occur if no urgent transition $\to_u$ is possible, formally $(P, \iota, v) \xrightarrow{\delta} (P, \iota, v + \delta)$ only if there are no $\delta' < \delta$, ι and P', v' such that $(P, \iota, v) \xrightarrow{\delta'} (P, \iota, v + \delta')$ and $(P, \iota, v + \delta') \to_u (P', \iota', v')$.

For the rest of the paper, we assume, that all apply transitions are urgent.

Multiplicities Up to now, we have focused on the modeling of an individual agent, like one molecule. However, it is possible to regard Labeled SpacePi agents as representatives of an equivalence class of non-interacting molecules, all exhibiting the same behavior (multiplicity). This helps to reduce computational costs of the analysis process, see Sec. 3, and therefore raises practicability. We introduce multiplicities by attaching a further variable $\#_m$ to each label m similar to the clock c_m that denotes the number of represented elements. We use a predicate symbol $\mu(\#_m, \#'_m)$ over the variable $\#_m$ for the old state and the variable $\#'_m$ for the new state to define evolution of multiplicities. We extend ι to also yield the predicate corresponding to the predicate symbol μ. Like for the spatial parameters, we use an apply $\alpha_c(\mu)$ operator for setting the multiplicity predicate. The semantics is given by the following urgent transition rule:

$$(m : \alpha_c(\mu).P, \iota, v) \to_u (m : P, \iota, v'). \quad \text{(APPLY-S)}$$

where v and v' coincide on every value except for $\#$ and the predicate $\iota(\mu)$ evaluates to true, i.e., $(v(\#_m), v'(\#'_m)) \in \iota(\mu)$.

Example 5 (Circular distribution)**.** Let an agent start in point p representing n molecules, that diffuse in all directions. A circular target of radius r is located in distance d. The number of molecules that will eventually arrive at the target can be approximated as follows: Consider the two tangent lines to the circle passing through the starting point p. The angle between both tangential lines is given by $2 \arctan(\frac{r}{d})$. The fraction of the molecules arriving at the target is therefore $\pi^{-1} \arctan(\frac{r}{d})$. Hence, the number of molecules that will eventually arrive at the target is approximated by the constraint $\mu := \#' = \# \cdot \frac{\arctan(\frac{r}{d})}{2\pi}$.

3 Verification

In order to analyze Labeled SpacePi processes, we introduce a translation into hybrid automata. Thereby, we make use of the work that has been done in the fields of automatic verification of hybrid systems, in particular of the tools HyTech (Henzinger et al. 2001) and HySAT (Fränzle et al. 2007).

3.1 Hybrid Automata

Hybrid automata have been introduced in (Alur et al. 1992) as an automaton model for describing the behavior of hybrid systems. Based on finite automata, timed automata (Alur & Dill 1994) are equipped with real valued clocks that guard transitions. This idea is generalized by hybrid automata. Instead of clocks, a set of real valued variables is used. The evolution of the variables is guarded by differential equations that are associated to the states (also called modes) of the hybrid automaton.

Definition 3 (Hybrid Automaton)**.** We fix a set $X = \{x_1, \ldots, x_n\}$ of real valued variables. A *hybrid automaton* A over X as defined in (Henzinger et al. 1998) is a directed multigraph (V_A, E_A). The states are called *control modes* and the transitions *control switches*. To each state an *invariant*, i.e. a guard on the variables, and an *activity*, i.e. a guard on the derivatives of the variables, is assigned, that are usually written as (in-) equalities. Activities are also called *flow conditions*. The transitions of hybrid automata are guarded by predicates over the free variables $X \cup X'$ where $x \in X$ denotes the value *before* the transition and $x' \in X'$ the value of the same variable *after* the transition. Furthermore, events for synchronization can be assigned to transitions. The semantics is defined in terms of a transition system, whose states are pairs of control modes and valuations of the variables. As for timed automata action and delay transitions are used.

3.2 From Labeled SpacePi to Hybrid Automata

We construct a hybrid automaton for a Labeled SpacePi process by computing the state space. To keep track of timeouts and the spatial positions of each movement unit, we introduce three variables and define flows and transition guards accordingly.

Construction of the Hybrid Automaton Let P be a Labeled SpacePi process and ι be a name interpretation. The state space of the corresponding automaton is the set of process terms modulo structural congruence that are reachable from an initial state P according to the Labeled SpacePi semantics. For each label m occurring in the transition system, we introduce a clock c_m for handling timeouts. Additionally, the two variables x_m and y_m are defined to capture the actual position of the process that is identified by the label m. For the transitions $\xrightarrow[u]{c}$ of the hybrid automaton, we split the guards of the Labeled SpacePi semantics into a condition c, which must evaluate to true for the transition to fire, and an update statement u defining the new values. This notation is also used by the tool HyTech. The definition of the transitions is inductive, similar to the definition of the Labeled SpacePi semantics.

To represent a delay in the hybrid automaton, we use the clock c_m corresponding to the location of the waiting process. The automaton can perform the transition if the clock reaches the timeout value. After that the clock is reset.

$$m : \tau_t.P + M \xrightarrow[c_m := 0]{c_m = t} m : P$$

To encode the reaction, we add the constraint on the variables as the guard and reset the corresponding clocks.

$$m_1 : \overline{x}\langle y\rangle.P_1 + N_1 \mid m_2 : x(z).P_2 + N_2 \xrightarrow[u]{c} m_1 : P_1 \mid m_2 : P_2\,\{{}^{y}/_{z}\}$$

where $c = \iota(s_1)(\tilde{x}_1, \tilde{y}_1) \wedge \iota(s_2)(\tilde{x}_2, \tilde{y}_2) \wedge ((\tilde{x}_1, \tilde{y}_1) + (x_{m_1}, y_{m_1}) - (\tilde{x}_2, \tilde{y}_2) + (x_{m_2}, y_{m_2}))^2 \leq \iota(r)^2$ and $u = c'_{m_1} = 0, c'_{m_2} = 0$. The notation $\iota(s)(\tilde{x}, \tilde{y})$ denotes the predicate, i.e., the corresponding (in)-equalities, in which the free variables are substituted by $(\tilde{x}, \tilde{y})$. The rules addressing the application operator mimic the semantics and reset the corresponding clocks.

$$\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right] : \alpha_f(f').P \xrightarrow[c'_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}=0]{} \left[\begin{smallmatrix} i & f' \\ p & s \end{smallmatrix}\right] : P$$

$$\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right] : \alpha_i(i').P \xrightarrow[c'_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}=0]{} (\left[\begin{smallmatrix} i' & f \\ p & s \end{smallmatrix}\right] : P)$$

$$\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right] : \alpha_p(p').P \xrightarrow[u]{} (\left[\begin{smallmatrix} i & f \\ p' & s \end{smallmatrix}\right] : P)$$

where $u = c'_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]} = 0, \iota(p)(x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}, y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]})$

$$\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right] : \alpha_s(s').P \xrightarrow[c'_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}=0]{} (\left[\begin{smallmatrix} i & f \\ p & s' \end{smallmatrix}\right] : P)$$

Calling a process identifier does not have a condition but resets the clock. This is necessary because the defining equation can start with a τ_t prefix.

$$m : A(\tilde{z}) \xrightarrow[c'_m=0]{} m : P_A\left\{{}^{\tilde{z}}/_{\tilde{x}}\right\}$$

if the identifier A is defined by $A(\tilde{x}) \triangleq P_A$. The last rules are needed for the inductive definition to handle parallel composition, the new operator and structural congruence.

$$\frac{m_1 : P \xrightarrow[u]{c} m_2 : P'}{(m_1 : P \mid m_3 : Q) \xrightarrow[u]{c} (m_2 : P' \mid m_3 : Q)}, \quad \frac{m_1 : P \xrightarrow[u]{c} m_2 : P', Q \equiv P, Q' \equiv P'}{m_1 : Q \xrightarrow[u]{c} m_2 : Q' \mid m_3 : Q},$$

$$\frac{m_1 : P \xrightarrow[u]{c} m_2 : P'}{m_1 : \nu x \tilde{P} \xrightarrow[u]{c'} m_2 : \tilde{P'}}$$

For the ν operator, the condition c' is obtained from c by removing the conditions involving the symbol x, thereby realizing the existential quantification of the symbol x regarding the interpretation ι. Furthermore, after constructing the state space, we ensure by alpha conversion that the set of bound and the set of free names are disjoint.
The activity is the same for all states. Each clock $c_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}$ has a derivative of 1 and for each pair of variables $x_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}, y_{\left[\begin{smallmatrix} i & f \\ p & s \end{smallmatrix}\right]}$ we add the corresponding (in)-qualities $\iota(f)$. Furthermore, to minimize the model we reduce the number of variables by reusing them when translating into the language of a model checker.

3.3 Spatio-Temporal Bisimilarity

The π-calculus is known to be turing complete. This clearly also holds for Labeled SpacePi being an extension of the π-calculus. It follows especially that the set of processes that are reachable by reduction does not need to finite. So, to ensure that the construction defined above yields a hybrid automaton with a finite set of states, we therefore need to consider a restricted set of processes for which we can be sure that the set reachable processes is finite. Otherwise the result would not be a hybrid automaton.

In (Dam 1997), Dam defines *finite control π-calculus processes* as processes that do not involve parallel composition under recursion and for these processes the set of reachable states is known to be finite. We adopt the notion of finite control for Labeled SpacePi and sketch that also for finite control Labeled SpacePi processes the set of reachable states is finite. This ensures that for the finite control Labeled SpacePi the construction above yields a finite state hybrid automaton. Furthermore, we have to ensure that the hybrid automaton has the same behavior as the Labeled SpacePi process, i.e., the construction is correct. To this end, we have to formalize the notion of the same behavior considering also time and space and do this by defining (strict) spatio-temporal bisimilarity. For this definition, we refine the standard notion of bisimilarity to cope with time and spatial positions of processes. Subsequently, we state the correspondence theorem relating the Labeled SpacePi process and the hybrid automaton and provide a proof sketch.

Definition 4. Let V_1 and V_2 be two finite sets of real valued variables. Let further $T_1 = (Q_1 \times \mathbb{R}^{|V_1|}, \rightarrow_1)$ be a transition system, where the states are pairs of states from Q_1 and a valuation of the variables in V_1. Let further $T_2 = (Q_2 \times \mathbb{R}^{|V_2|}, \rightarrow_2)$ be a transition system, where the states are pairs of states from Q_2 and a valuation of the variables in V_2.

A pair $\sigma = (\sigma_s, \sigma_v)$ is a *spatio-temporal simulation* if there are constants $\lambda_1, \ldots \lambda_{|V_1|}$ and $b_1, \ldots b_{|V_1|}$ such that the following conditions are met:

1. $\sigma_s \subseteq (Q_1 \times \mathbb{R}^{|V_1|}) \times (Q_2 \times \mathbb{R}^{|V_2|})$ is a relation on the states.

2. $\sigma_v \subseteq V_1 \times \mathbb{R}^2 \times V_2$ is a relation on the variables involving a translation.

3. $(s_1, v_1)\sigma_s(s_2, v_2)$ and $v_1(x_1) = \lambda v_2(x_2) + b$ for all $(x_1, \lambda, b, x_2) \in \sigma_v$ and $(s_1, v_1) \to (s_1', v_1')$ implies that there is a (s_2', v_2') with $(s_2, v_2) \to (s_2', v_2')$, $(s_1', v_1')\sigma_s(s_2', v_2')$, and $v_1'(x_1) = \lambda v_2'(x_2) + b$ for all $(x_1, \lambda, b, x_2) \in \sigma_v$

A simulation is *strict* iff $\sigma_v \subseteq V_1 \times \{1\} \times \{0\} \times V_2$. A simulation $\sigma = (\sigma_s, \sigma_v)$ is a *spatio-temporal bisimulation* if $\sigma^{-1} = (\sigma_s^{-1}, \sigma_v^{-1})$ with $(x_2, \lambda, b, x_1) \in \sigma_v^{-1}$ iff $(x_1, \lambda^{-1}, -b, x_2) \in \sigma_v$.

Note, that this definition considers a τ_d statement to be observable from the outside. Using this definition, we can relate the Labeled SpacePi process to the corresponding hybrid automaton.

Theorem 5. Let P be a *finite control* Labeled SpacePi process, i.e., a Labeled SpacePi process not involving parallel composition under recursion and let ι be an interpretation of the names. Let further R be the set of states reachable from P modulo structural congruence and let A by the hybrid automaton constructed as described above. Then the transition system for the semantics of P and the transition system for the semantics of the hybrid automaton A are strictly spatio-temporal bisimilar.

At first, we show that the set of reachable states by the reduction relation is finite. Starting with a finite control Labeled SpacePi process Q, we construct a π-calculus process P by removing labels, and replacing the α and τ_t operators by τ prefixes. It is easy to see that the state set of the hybrid automaton is a subset of the set $R = \{P' \mid P \to^* P'\}/\equiv$ of π-calculus terms reachable by reduction from P modulo structural congruence. The π-calculus process is furthermore finite control by construction. From Montari and Pistore's result (Montanari & Pistore 2001) follows that every finite control process P is also finitary and therefore the set R must be finite. Therefore, the state space of the hybrid automaton is finite. As the construction of the guards of the automaton encodes the Labeled SpacePi semantics, it is clear that the semantics of the automaton and the process are strictly bisimilar related by the identity on the states and the variables.

Urgency Urgencies can be modeled by introducing the negation of the conjunction of all guards for the outgoing transitions as an invariant plus the bound. For example if the outgoing edge has guard $x <= 5$ then the invariant is $x >= 5$. The case $x = 5$ must be allowed, because the transition does not consume time and the invariant must be satisfied when the transition takes place. However, in tools, invariants are normally required to be convex, i.e., if the invariant is satisfied when entering and leaving the state, it must also be satisfied at all time points in between. Thus, e.g. for an invariant of the type $x < a \vee y < b$ which is not convex, the state must be duplicated, one having the invariant $x < a$ and the other $y < b$.

3.4 Verification Tools

HyTech is the classical tool but is restricted to the checking of linear hybrid automata, having only conjunctions of linear guards as flow conditions. Starting from a given state of the automaton and a linear conjunction of conditions on the initial values of the variables, it iteratively computes the possible successor states. Since this is a semi-decision procedure, termination is not guaranteed. Additionally, the tool is very sensitive to the number of variables. However, in contrast to the bounded model checking tool HySAT, analysis is not limited to a predetermined number of steps, such that the entire space of reachable states can be explored.

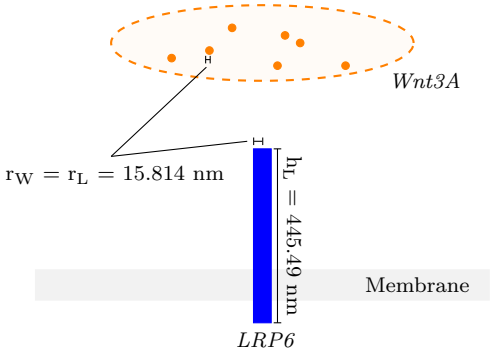

Figure 3: A simplistic illustration of the arrival of *Wnt3A* molecules at an *LRP6* receptor.

HySAT is a bounded model checker that combines a SAT solver with a solver for real arithmetics. Its main advantage is that it can handle more variables than HyTech. As it turned out in experiments, it is also faster and thus more suitable for analyzing larger systems with more agents. Furthermore, HySAT offers a richer language for specifying arithmetical constraints, including e.g. trigonometric functions. Yet, analysis results only consider a bounded number of state transitions. However, we believe that bounded model checking is well suited for the analysis of biological systems, where observation time is naturally limited. In HySAT, the discrete state space and the transition relations are encoded in boolean formulae and the continuous behavior in arithmetical constraints. The SAT approach creates boolean variables for every step of the automaton. HySAT is able to determine whether a state is not reachable within a given number of steps. However, if it determines that a given state is reachable, this result is only an approximation depending on the accuracy of the arithmetic. The tool provides intervals for each real valued variable in which a satisfying valuation is expected to be found.

4 Examples

This Sec. gives two examples for Labeled SpacePi models and their analysis. The first example, which describes an initial step of the activation of the Wnt signaling pathway, focuses on the integration of available biological data and the handling of multiplicities. By contrast, the second example, a model of active transport in cells, is not based on biological data. Its purpose rather on presenting the treatment of process communication. In the lines of Sec. 3, the two tools HyTech and HySAT are used for model analysis.

4.1 Activation of the Wnt Pathway

Signaling pathways are reaction networks that relay signals from the cell membrane to the cell core (nucleus) leading to changes in gene expression, see e.g. (Gomperts et al. 2002). In life science, they are of major interest, since they play a key role in the cure of e.g. cancer or Parkinson's disease. An important step in the Wnt signaling pathway is its activation by the arrival of *Wnt3A* proteins at *LRP6* receptors. In the following, a simplistic model of this event is described, see Figure 3. It illustrates the concept of composing

//channels
stop: 0.0
//movements
stay := $\dot{x}^2+\dot{y}^2 = 0$
mov := $\dot{x}^2+\dot{y}^2 \leq 4000$ *//4000 = diffusion constant*
//positions
pos_L := x=0 ∧ y=0
pos_W := $-25 \leq x \leq 25 \wedge 725 \leq y \leq 775$
//shapes
$shape_L$:= $0 \leq x \leq 31.628 \wedge 0 \leq y \leq 445.49$
$shape_W$:= $x^2+y^2 \leq 15.814^2$ *//circle, radius = r_W*
//multiplicities
$\mu := \#' = \#\arctan(\frac{r_W}{750-445.49})\frac{1}{\pi}$
//processes
Wnt3A(stay) ≙
$\alpha_p(pos_W).\alpha_s(size_W).\alpha_f(mov).\overline{stop}\langle\rangle.\alpha_c(\mu).\alpha_f(stay).\tau_\infty$
LRP6() ≙ $\alpha_p(pos_L).\alpha_s(size_L).\alpha_f(stay).stop().LRP6()$
//initial process
(m_1: $\alpha_c(\# = 100)$.Wnt3A(stay) | m_2: LPR6())

Figure 4: A simplistic Labeled SpacePi model of the arrival of *Wnt3A* at *LRP6*.

and analyzing Labeled SpacePi models, based on a given set of biological data.
The model is given in Figure 4. *Wnt3A* is represented by the process `Wnt3A`. `Wnt3A` first applies an initial position and size and then performs the diffusive motion `mov` until it reaches `LRP6`, where it stops. The arrival of `Wnt3A` at `LRP6` is signalized by synchronization on the channel `stop`.
The data to be integrated into the model are: the volumes and shapes of molecules, their positions and diffusion constants, and their numbers. Positions and shapes are defined by boolean combinations of (in)-equalities over the considered spatial dimensions. They are bounded by the molecules' possible locations and their volumes, respectively. For *Wnt3A* we consider the volume $V_W = 1.6566 * 10^4\ nm^3$ and for *LRP6* $V_L = 3,5281 * 10^5\ nm^3$ (provided by (Letunic et al. 2006)). Because of folding processes the shapes of proteins need to be defined for each model individually. As *Wnt3A* is about 20 times smaller than *LRP6*, its shape has rather little impact, such that it is abstracted as a sphere. We assume that, as a receptor, *LRP6* is rarely folded. Thus, it is represented as a cylinder. To simplify, the radius of *LRP6* is set to $r_L = r_W = 15.814\ nm$, yielding the height $h_L = 445,49\ nm$. Additionally, we reduce the system to its two-dimensional projection. Molecular motion is defined by combinations of (in)-equalities over the considered spatial dimensions and their derivatives, which are upper bounded by diffusion constants. For *Wnt3A*, we assume the diffusion constant $D_W = 4000\ nm^2/s$, which is a common value for intracellular motion. Molecule numbers are mapped to multiplicities, see Sec. 5. In the initial process, the number of *Wnt3A* molecules is set to 100.

4.1.1 Analysis Using HySAT

We choose HySAT for analysis as this tools offers a richer variety of built-in mathematical functions. Bounded model checking is sufficient to explore the whole state space here as the hybrid automaton has only two states and no loops. In the following, we first discuss the HySAT model resulting from the translation. This also sheds additional light on the translation method in general. Subsequently, we demonstrate how HySAT can be used to analyze the model.

We exemplify the translation from the Labeled SpacePi process into the input language of HySAT by the snippet presented in Listing 1. The input of HySAT is a transition system. The nodes are valuations of boolean and real variables. The transition system is defined by a set of boolean formulae over the variables, where the unprimed version corresponds to the valuation in the source state of the transition and the primed version to the valuation in the destination state. To encode a hybrid automaton, we introduce one transition for every control switch of the hybrid automaton and one transition for the flow. In the run of a hybrid automaton every control switch is followed by a flow and vice versa. We employ the boolean variable *jump* to indicate that the next transition is a control switch. Accordingly, !*jump* indicates a flow. HySAT automatically generates the corresponding expanded boolean formula from this specification.

We now discuss the model of our example in more detail. The discrete state space of the hybrid automaton has two states, *init* and *stop*. The state *init* represents the system before and the state *stop* after the communication on `stop`. The continuous state space is defined by the variables corresponding to the label m_1 of the `Wnt3A`, i.e. $xm1$, $ym1$ representing the spatial position, and the clock $cm1$. More precisely, the variables $xm1$, $ym1$ denote the center of the circle of `Wnt3A`. We further use auxiliary variables for the definition of the transitions and flow conditions.

The flow condition (flow) defines how the continuous variables evolve while the system is in the *init* state. This flow represents the movement of `Wnt3A` towards `LRP6`. In the HySAT model, the flow condition is indicated by the `!jump` condition in the premise of the implication. The condition $cm1' = cm1 + dt$ defines that the clock $cm1$ advances by dt. The movement of the `Wnt3A` in x and y direction corresponds to the variables $dx1$ and $dy1$, respectively. The constraint $(dx1*dx1+dy1*dy1)*pi <= DW*dt$ ensures that `Wnt3A` can at maximum proceed according to the diffusion rate DW. The variable $m1count$ models the multiplicity.

The jump condition, indicated by `jump`, defines the discrete state transitions. The second formula encodes that *stop* is the only possible successor of the state *init*. The third formula defines the transition from state *init* to state *stop* when `Wnt3A` and `LRP6` communicate over `stop`. This transition can only occur if there is an overlapping of the participants. Formally, this requires one point ($xsm2$,$ysm2$) inside `LPR6`, defined by conditions (i1) - (i4), that is equal to a point ($xsm1$,$ysm1$) inside `Wnt3A`. The equality condition is formalized in (e1) and (e2), whereas Condition (i5) specifies that the point $(xsm1, xsm2)$ is inside `Wnt3A`. Conditions (a1) and (a2) calculate the constraint on the multiplicity similar to the approach shown in Example 5.

```
(!jump and init ->
 init ' and cm1' = cm1 + dt and xm1'= xm1 − dx1 and
 ym1'= ym1 − dy1 and (dx1*dx1 + dy1*dy1)*pi<=DW*dt
 and m1count' = m1count);  --   ( flow )

(jump and init -> stop');

(jump and init and stop' ->
 xsm1 = xsm2                  --   ( e1 )
 and ysm1 = ysm2              --   ( e2 )
 and xsm2 > xiLPR             --   ( i1 )
 and ysm2 > yiLPR             --   ( i2 )
 and xsm2 < (xiLPR + xsLPR)   --   ( i3 )
 and ysm2 < (yiLPR + ysLPR)   --   ( i4 )
 and (xsm1 − xm1)^2+(ysm1 − ym1)^2 < rW^2 (i5)
 and cm1'=cm1+dt
 and sin(angle) *(yiW−ysLPR) = rW*cos(angle)  -- ( a1 )
 and m1count'*pi=angle*m1count );  --   ( a2 )
```

Listing 1: Snippet of HySAT Specification

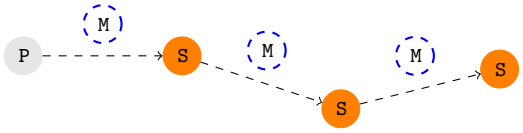

Figure 5: Active transport in SpacePi - P produces molecules M that move along static parts S.

//channels
trans: 0
//constants investigated in analysis Sec.
t_S := ..., t_P := ...
//shapes
$shape_M := x^2+y^2 \leq 0.5^2$ *//circle, radius = 0.5*
$shape_S := x^2+y^2 \leq 0.5^2$ *//circle, radius = 0.5*
//movements
$m_u := \dot{x}=1, \dot{y}=1$
$m_d := \dot{x}=1, \dot{y}=-1$
$m_s := \dot{x}=1, \dot{y}=0$
//processes
$M() \triangleq trans(m).\alpha_f(m).M()$
$S(m) \triangleq \overline{trans}\langle m\rangle.\tau_{t_S}.S(m)$
$P() \triangleq \tau_{t_P}.(P()|\nu m.\alpha_i(m).\alpha_p(p_1).\alpha_f(m_s).M())$
//initial process
m_1:$(\alpha_p((0,0)).P()|\alpha_p((4,1)).S(m_d)|$
$\alpha_p((4,5)).S(m_u)|\alpha_p((6,1)).S(m_d))$

Figure 6: A Labeled SpacePi model of active transport

We are interested in the timing behavior of the system. The goal is to establish a bound on the time of the interaction at LPR6. To this end, we define the target property that the state *stop* is reached and the clock is below a given value and let HySAT check this property for our model. Iteratively decreasing the given value for the clock in the target property, reveals that HySAT cannot find a solution in which Wnt reaches LRP6 in 17 seconds. However, for 18 seconds it can find a solution. The analysis further yields that on the long run 1.5% of Wnt will eventually reach LRP6. Using different constraints for the movement function or the constraint on the multiplicity, more specialized models of diffusion, e.g. with the Fick-Equations, can be modeled and analyzed in Labeled SpacePi. This is, however, outside the scope of this paper.

4.2 Active Transport

The term active transport addresses different sorts of molecular motion that are performed against concentration gradients and thus enable cells to overcome equal molecular distributions. The form of active transport, we focus on in this example, refers to the motion of molecules along fixed intracellular structures, like microtubules, by consecutive binding to structure parts under energy consumption. It is significantly faster than diffusion and thus leads to an acceleration of ongoing intracellular processes. The main purpose of this example is to illustrate the use of abstraction as part of the analysis process.
Figure 5 shows the basic principle of the model. The shapes of all molecules are abstracted as circles, since the included values are not related to experimental data. Processes that represent moving molecules, called M, start at process P and move along static processes S, abstracting the structure parts. M moves between two S processes by receiving the needed movement function from the first S to reach the second one.
Figure 6 reveals more details about the model. The three movement functions describe the movement of M to an S above it, underneath it, and to its right, respectively. Instances of M are produced by P with a delay of t_P. M moves from one S to the next, by receiving the appropriate movement function on trans. In order for M and S to communicate, the arrival time between two M at S has to be greater t_S, as denoted by τ_{t_S}. The impact of t_S and t_P on the model is investigated in the analysis Sec. below.

4.2.1 Analysis

To illustrate the treatment of process communication in more detail, we analyze the interplay of the constants t_P and t_S defining the frequency with which new agents can be created at P and the time it takes at each location S to handle one molecule. For a concise presentation of the example, we focus on the first instance of S located at (4,1) and check whether it is possible that a M does not synchronize at this location. To obtain a finite state hybrid automaton, we underapproximate the system behavior by fixing the number of M. This is further justified as due to the t_P delay, there can always be only a finite number of M between P and the first S. For conciseness, we choose to analyze two M.

HyTech The construction of the HyTech model from the Labeled SpacePi process is very similar to one presented in Sec. 4.1.1. However, HyTech can only check hybrid automata in which the guards are conjunctions of linear inequalities. Therefore, we abstract the reaction radius by a rectangle. Listing 2 shows a snippet of the HyTech specification. In HyTech, the hybrid automaton can be specified directly. The keyword **loc** precedes the definition of a location (state). The **while** *invariant* **do** *flow* statement defines the state invariants and flows in which the first derivate of a variable xm is denoted by dxm. The mode switches are defined by the statements of the form **when** *condition* **do** *set variables* **goto** *target state*.

We now turn to the model. The state *L1* represents the configuration where the first M has been released from P. The convex invariant 4 − xm1 >= distT & cm1 <= tP in line (1) ensures a transition as soon as either the first M labeled m1 arrives at position S or the P process has completed the waiting time. The second line encodes the movement function m_s for the first M. In this expression terms dx denote the time derivate of the variable x. For the transition from state *L1* two cases have to be distinguished. The first case (3) is that the timeout of P occurs first and the second case (4) is that the reaction occurs first.

To find out if one M does not interact at the first S, we check whether the variable $xm2$ associated with the label of the second instance of M can reach a value right of S. This is sufficient as we only consider one S for the analysis. The requirement is fulfilled if the time t_P is much lower than the reaction time t_S at the first S such that the second M passed the inactive S. But it turns out that this can also occur if t_P is higher than initially expected which is due to the fact that the first M going down after the reaction with S can react a second time at the same S if the delay time is smaller than the time for the molecule to leave the reaction radius of S. This is a result which is not directly obvious from the model.

The experiment shows, that the use of abstraction is crucial as HyTech is otherwise not able to analyze the whole state space. Abstraction removes potentially unreachable or irrelevant states. Due to space limitations, abstraction techniques are not discussed in this paper. Similar to the previous example, the

tool HySAT was also used for the analysis. It provided similar advantages as discussed above but does not explore the whole state state. Details have to be omitted due to space limitation.

```
loc L1: while xp1 − xm1 >= distT & cm1 <= tP wait  −−
 (1)
 {dxm1 = xmstr , dym1 = ymstr, dxm2 = 0, dym2 = 0}
−−  (2)
 when cm1 = tP
  do {cm1' = 0 , xm2' = xp0 , ym2' = yp0}
  goto L2;  −−  (3)
 when xp1 − distT <= xm1 & xm1 <= xp1 + distT & ...
  do {cmp1'=0} goto L3;  −− (4)
```

Listing 2: Snippet of HyTech Specification

5 Query Logic

Up to now, we have introduced the modeling language Labeled SpacePi and demonstrated the application on two examples. The queries used to analyze the system were developed ad hoc and considered the question whether there is evolution of the system that can lead to a specific state. Such properties are called liveness properties.

In this section, we will mainly focus on safety-properties, i.e., properties that something bad cannot happen in all possible evolutions of the system. These properties are dual to liveness properties. We give a general solution and introduce a query logic for expressing the biological properties similar to (Fages & Rizk 2008). We further discuss how it can be checked if a biological Labeled SpacePi model has a property defined in this logic and following the ideas of (Fages & Rizk 2008), we choose a linear time logic.

The atom formulas in our query logic specify a system state at a certain point in time, like the distance of two molecules, their velocities or numbers. To this end, we use arithmetical equations or inequations E over the variables corresponding to the labels as atom formulas (like c_m, x_m, y_m). The atom $x_{m_1} - y_{m_2} \leq 3$ for example specifies a state in which the agent with label m_1 and the agent with label m_2 have an distance in x-direction below 3. The semantics of an expression E depends on the valuation ν of the variables and we write $\nu \models E$ iff the valuation satisfies the expression.

The formulas are built using the usual boolean connectives and the LTL operators. The boolean connectives are conjunction $\wedge$, disjunction $\vee$, and negation $\neg$. The LTL operators as in (Fages & Rizk 2008, Clarke et al. 1999) are next state f ($\mathbf{X}f$), eventually f ($\mathbf{F}f$), always f ($\mathbf{G}f$), and f until g (f $\mathbf{U}$ g).

The property that an agent m_a never reaches a target m_t would be expressed by

$$F_s = \mathbf{G}(x_{m_a} \neq x_{m_t} \vee y_{m_a} \neq y_{m_t})$$

.

Let P be an Labeled SpacePi process term and let $\rightarrow, \xrightarrow{\delta}$ the transition relation defined by the semantics. Consider the following transition relation $\Rightarrow = \xrightarrow{\delta} \circ \rightarrow$ abstracting from the delay transitions. A path $\sigma = (P_0, \iota_0, \nu_0)(P_1, \iota_1, \nu_1)\ldots$ is a sequence of states linked by $\Rightarrow$, i.e., $\forall i : (P_i, \iota_i, \nu_i) \Rightarrow (P_{i+1}, \iota_{i+1}, \nu_{i+1})$.

We define then that $\sigma = (P_0, \iota_0, \nu_0)(P_1, \iota_1, \nu_1)\ldots$

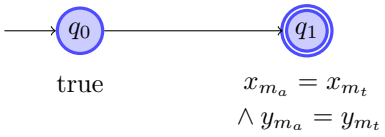

Figure 7: Automaton for $\neg F_s$

satisfies F, denoted by $\sigma \models F$ inductively as follows

$$\begin{aligned}
\sigma \models E &\iff \nu_0 \models E \\
\sigma \models \mathbf{X}F &\iff (P_1, \iota_1, \nu_1)(P_2, \iota_2, \nu_2)\ldots \models F \\
\sigma \models \mathbf{F}F &\iff \exists i : (P_i, \iota_i, \nu_i)\ldots \models F \\
\sigma \models \mathbf{G}F &\iff \forall i : (P_i, \iota_i, \nu_i)\ldots \models F \\
\sigma \models F\mathbf{U}G &\iff \exists i : (P_i, \iota_i, \nu_i)\ldots \models G \\
&\quad\ \forall 0 \leq k \leq i : (P_k, \iota_k, \nu_k)\ldots \models F
\end{aligned}$$

and the definition of semantics for conjunction, disjunction and negation is defined as usual. A state (P, ι, ν) satisfies a formula F iff all paths starting in this state satisfy F. The logic is compatible with the notion of spatio-temporal bisimilarity as stated by the following proposition.

Proposition 6. Let (P_1, ι_1, ν_1) and (P_2, ι_2, ν_2) be strictly spatio-temporal bisimilar and let F be an LTL formula, then $(P_1, \iota_1, \nu_1) \models F$ iff $(P_2, \iota_2, \nu_2) \models F$.

Using the technique in (Clarke et al. 1999), we can construct an automaton A accepting all paths that do not satisfy F. For checking a safety-property, this automaton can be translated into the HySAT representation and analyzed by bounded model checking if an accepting state is reachable. Consider the safety property above, the negation is $\mathbf{F}(x_{m_a} = x_{m_t} \wedge y_{m_a} = y_{m_t})$ and an automaton accepting all paths that satisfy this formula is sketched in Figure 7. We run the automaton in parallel with the model of the biological system. If the final state q_1 cannot be reached, we can be sure that there is no path satisfying $\neg F_s$, so all paths satisfy F_s. The bounded model checking with HySAT can check whether the state q_1 is reachable in a bounded number of steps. Thereby we can establish that the biological model satisfies F_s for all path of a given bounded length.

This bounded model checking approach does not work for liveness properties. The negation of a property that a all paths eventually satisfy F is that there is no path that always satisfies $\neg F$ which cannot be established if the state space is only explored up to a bounded depth. However, some of these properties can be checked directly as shown in the two example case studies.

6 Related work

Different approaches have been applied to spatial modeling. Differential equations support a population-based view on systems, where species concentrations are of interest. Thus, detailed information about individual molecules and their locations cannot be taken into account. Various concepts base on space discretization (Elf & Ehrenberg 2004, Regev et al. 2004, Cardelli & Gordon 1998, John, Lhoussaine, Niehren & Uhrmacher 2008), i.e. they assume sub-volumes in which molecules are equally distributed. Instead of continuous functions, these approaches make us of reactions to describe movements as events that lead to position changes. Therefore, the process definitions of the Wnt pathway example would read in e.g. stochastic Pi like $Wnt3A() \triangleq$

$\overline{\texttt{stop}}\langle\rangle.\tau_\infty, LRP6() \triangleq \texttt{stop}().LRP6()$. Thereby, the interaction on stop denotes that $Wnt3A()$ has fulfilled its movement from its initial position to $LRP6()$. This more abstract view hampers the modeling of important spatial effects in cells, like molecular crowding (Takahashi et al. 2005), where molecular motion is constrained by limited space. Additionally, methods that assume sub-volumes require reaction rate constants, that are in general not at hand and hard to estimate in case of complex systems. Molecular and Brownian Dynamics, e.g. (Takahashi et al. 2003), provide a very detailed view on systems. However, their computational costs for large systems are too high to cover the normal time scale of wet-lab experiments spanning several hours.

Multiple approaches for the model checking of biological systems exist, e.g. (Ciocchetta & Hillston 2008, Calzone et al. 2006, Heath et al. 2008, Batt et al. 2005). However, they only consider spatial information in form of compartments and do not take protein locations and shapes or intracellular structures into account. The spatial logics model checker (Vieira et al. 2005) allows for the checking of a subset of π-calculus specifications. Yet, it does not consider physical space. Other, more recent developments for checking the π-calculus (Meyer et al. 2008) use methods for Petri Nets. The tool MoDiShCa (Quesel & Schäfer 2006) verifies physical mobility of systems. It translates a Shape Calculus (Schäfer 2007) specification into monadic second order logic and uses a tool for this logic as verification back-end. However, the specification of the system is declarative in a logic, it does not allow for communication of positions and movements and the verification in MoDiShCa is limited to discrete time and finite space.

There are also several approaches that extend process algebras for the modeling of hybrid systems, like the Φ calculus (Rounds & Song 2003). However, as they do not provide built-in support for describing mobility, positions and movements have to be encoded. Developed for the modeling of satellite communication, J. C. M. Baeten and J. A. Bergstra introduce the Real Space Process Algebra in (Baeten & Bergstra 1991), where communication has a three or four dimensional position. Yet, this approach neither considers the movement of processes nor provides automatic verification.

7 Conclusion & Outlook

In this paper, we introduced Labeled SpacePi, a formalism for the modeling of time and space in biomolecular systems, which is tailored to the available knowledge and data. It refines the former work in (John, Ewald & Uhrmacher 2008) by a more concise reduction semantics that makes use of labels as known from the field of hybrid logics. We believe that the idea of using labels in process algebra can be beneficially applied to other purposes, e.g. to specify in a π-calculus logic that agents share a common name. Furthermore, Labeled SpacePi provides the concept of multiplicity, i.e. sets of non-interacting can be represent as single processes, which helps to lower computational costs of model analysis. The presented approach for model analysis considers a system's entire set of possible evaluations and is based on a translation from Labeled SpacePi to hybrid automata. The advantage of having such a translation is that Labeled SpacePi allows for a simpler modeling of biological systems than hybrid automata. This is because, several concepts of Labeled SpacePi cannot be directly expressed in hybrid automata, e.g. the transmission of movement functions from one agent to another or the creation of new independent agents using the ν operator. They can only be encoded by building a single complex hybrid automaton for the entire system yielding high modeling effort. By defining a spatio-temproal bisimilarity, we were able to relate Labeled SpacePi processes to the constructed hybrid automata. We applied our approach to two use case studies, addressing the activation of the Wnt signaling pathway and active transport in cells, and presented how properties of Labeled SpacePi models can be derived using the established tools HyTech and HySAT.

Regarding future work, we would like to extend the logic for specifying system properties of biological systems, especially with regard to spatial phenomena. Another goal is to further lower the computational costs of the analysis process by additional abstraction techniques for obtaining state finite systems with finitely many variables. Furthermore, we would like to investigate how recent results on π-calculus verification like (Meyer et al. 2008) can be used for Labeled SpacePi.

References

Alur, R., Courcoubetis, C., Henzinger, T. A. & Ho, P.-H. (1992), Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems, *in* 'Hybrid Systems', pp. 209–229.

Alur, R. & Dill, D. L. (1994), 'A Theory of Timed Automata', *TCS* **126**(2), 183–235.

Areces, C. & ten Cate, B. (2006), Hybrid Logics, *in* 'Handbook of Modal Logics', Elsevier.

Baeten, J. C. M. & Bergstra, J. A. (1991), Real Space Process Algebra, *in* 'International Conference on Concurrency Theory', Vol. 527 of *LNCS*, Springer, pp. 96–110.

Batt, G., Ropers, D., de Jong, H., Geiselmann, J., Mateescu, R., Page, M. & Schneider, D. (2005), Analysis and Verification of Qualitative Models of Genetic Regulatory Networks: A Model-Checking Approach, *in* 'International Joint Conferences on Artificial Intelligence', pp. 370–375.

Bengtsson, J. & Yi, W. (2003), Timed Automata: Semantics, Algorithms and Tools, *in* 'Lectures on Concurrency and Petri Nets', Vol. 3098 of *LNCS*, Springer, pp. 87–124.

Calzone, L., Fages, F. & Soliman, S. (2006), 'BIOCHAM: An Environment for Modeling Biological Systems and Formalizing Experimental Knowledge', *Bioinformatics* **22**(14), 1805–1807.

Cardelli, L. & Gordon, A. D. (1998), Mobile Ambients, *in* 'Foundations of Software Science and Computation Structures', Vol. 1378 of *LNCS*, Springer, pp. 140–155.

Ciocchetta, F. & Hillston, J. (2008), 'Bio-PEPA: An Extension of the Process Algebra PEPA for Biochemical Networks', *ENTCS* **194**(3), 103–117.

Clarke, E. M., Grumberg, O. & Peled, D. (1999), *Model Checking*, The MIT Press.

Dam, M. (1997), 'On the Decidability of Process Equivalences for the Pi Calculus', *TCS* **183**(2), 215–228.

Elf, J. & Ehrenberg, M. (2004), 'Spontaneous Separation of Bi-Stable Biochemical Systems into Spatial Domains of Opposite Phases', *Systems Biology* **1**(2), 230–236.

Fages, F. & Rizk, A. (2008), 'On temporal logic constraint solving for analyzing numerical data time series', *Theor. Comput. Sci.* **408**(1), 55–65.

Fränzle, M., Herde, C., Teige, T., Ratschan, S. & Schubert., T. (2007), 'Efficient Solving of Large Non-linear Arithmetic Constraint Systems with Complex Boolean Structure', *J. on Satisfiability, Boolean Modeling and Computation* **1**, 209–236.

Gomperts, B. D., Kramer, I. M. & Tatham, P. E. R. (2002), *Signal Transduction*, 1 edn, Academic Press.

Heath, J., Kwiatkowska, M., Norman, G., Parker, D. & Tymchyshyn, O. (2008), 'Probabilistic Model Checking of Complex Biological Pathways', *TCS* **319**(3), 239–257.

Henzinger, T. A., Kopke, P. W., Puri, A. & Varaiya, P. (1998), 'What's Decidable about Hybrid Automata?', *J. Comput. Syst. Sci.* **57**(1), 94–124.

Henzinger, T. A., Preussig, J. & Wong-Toi, H. (2001), Some Lessons from the HyTech Experience, *in* 'Conference on Decision and Control', Vol. 3, IEEE Press, pp. 2887–2892.

John, M., Ewald, R. & Uhrmacher, A. M. (2008), 'A Spatial Extension to the Pi Calculus', *ENTCS* **194**(3), 133–148.

John, M., Lhoussaine, C., Niehren, J. & Uhrmacher, A. M. (2008), The Attributed Pi Calculus, *in* 'Computational Methods in Systems Biology', Vol. 5307 of *LNBI*, Springer, pp. 83–102.

Kholodenko, B. N. (2006), 'Cell-Signalling Dynamics in Time and Space', *Nature Reviews Molecular Cell Biology* **7**(3), 165–176.

Letunic, I., Copley, R. R., Pils, B., Pinkert, S., Schultz, J. & Bork, P. (2006), 'Smart 5: domains in the context of genomes and networks', *Nucleic Acids Research* **34**(Database-Issue), 257–260.

Meyer, R., Khomenko, V. & Strazny, T. (2008), A Practical Approach to Verification of Mobile Systems Using Net Unfoldings, *in* 'Applications and Theory of Petri Nets', Vol. 5062 of *LNCS*, Springer, pp. 327–347.

Meyvis, T., De Smedt, S., Van Oostveldt, P. & Demeester, J. (1999), 'Fluorescence Recovery After Photobleaching: A Versatile Tool for Mobility and Interaction Measurements in Pharmaceutical Research', *Pharmaceutical Research* **16**(8), 1153–1162.

Milner, R. (1999), *Communicating and Mobile Systems: the Pi-Calculus*, Cambridge University Press.

Montanari, U. & Pistore, M. (2001), History-Dependent Automata, Technical report, Istituto Trentino di Cultura,.

Polakis, P. (2007), 'The Many Ways of Wnt in Cancer', *Current Opinion in Genetics & Development* **17**(1), 45–51.

Quesel, J.-D. & Schäfer, A. (2006), Spatio-Temporal Model Checking for Mobile Real-Time Systems, *in* 'Theoretical Aspects of Computing', Vol. 4281 of *LNCS*, Springer, pp. 347–361.

Regev, A., Panina, E. M., Silverman, W., Cardelli, L. & Shapiro, E. (2004), 'BioAmbients: An Abstraction for Biological Compartments', *TCS* **325**(1), 141–167.

Regev, A. & Shapiro, E. (2002), 'Cells as Computation', *Nature* **419**, 343.

Rivas, E. & Eddy, S. R. (1999), 'A Dynamic Programming Algorithm for RNA Structure Prediction including Pseudoknots', *J. Mol. Biol.* **285**(5), 2053–2068.

Rounds, W. C. & Song, H. (2003), The Phi-Calculus: A Language for Distributed Control of Reconfigurable Embedded Systems, *in* 'Hybrid Systems: Computation and Control', pp. 435–449.

Schäfer, A. (2007), 'Axiomatisation and Decidability of Multi-Dimensional Duration Calculus', *Information and Computation* **205**(1), 25–64.

Takahashi, K., Ishikawa, N., Sadamoto, Y., Sasamoto, H., Ohta, S., Shiozawa, A., Miyoshi, F., Naito, Y., Nakayama, Y. & Tomita, M. (2003), 'E-Cell 2: Multi-Platform E-Cell Simulation System', *Bioinformatics* **19**(13), 1727–1729.

Takahashi, K., Nanda, S., Arjunan, V. & Tomita, M. (2005), 'Space in Systems Biology of Signaling Pathways : Towards Intracellular Molecular Crowding in Silico', *FEBS letters* **579**(8), 1783–1788.

Thompson, C. B. (1995), 'Apoptosis in the Pathogenesis and Treatment of Disease', *Science* **267**, 1456–1462.

Vieira, H., Caires, L. & Viegas, R. (2005), The Spatial Logic Model Checker User's Manual v1.0, Technical Report TR-DI/FCT/UNL-05/2005, Universidade Nova de Lisboa.

Zhao, T. & Murphy, R. F. (2007), 'Automated Learning of Generative Models for Subcellular Location: Building Blocks for Systems Biology', *Cytometry Part A* **71A**(12), 978–990.

Zuker, M. (2003), 'Mfold Web Server for Nucleic Acid Folding and Hybridization Prediction', *Nucleic Acids Res.* **31**(13), 3406–3415.