Towards interactive model checking of biological pathway dynamics

Stefano Schivo*, Jetse Scholma*, Paul E. van der Vet, Marcel Karperien, Janine N. Post, Jaco van de Pol**, and Rom Langerak**

University of Twente, Enschede, The Netherlands

Abstract. ~~Computational support has become a fundamental component of biological laboratory experiments, where large amounts of data are generated on a daily basis. The analysis of such data gives rise to new questions and further experiments, whose direction can be better decided via preliminary modeling. The more complex a biological process is,~~ Biological systems, such as e.g. regulatory or gene networks, can be seen as a particular type of distributed systems, and for this reason they are modelled with the same tools that were developed in the ~~more important computational methods become in supporting the biologist's work.~~ computer science context. However, tools designed to model distributed systems often require a computer science background, making their use less attractive for biologists. ANIMO (Analysis of Networks with Interactive ~~MOdeling) is a software tool that allows the experimental biologist to interactively explore the dynamic behavior of signaling networks. We present here a new modeling~~ MOdelling) was built with the purpose to provide biologists with access to the powerful modelling formalism of Timed Automata without the need to learn its working. This brings computational support closer to the biological laboratories, where large amounts of data are generated on a daily basis, but where modelling is hardly ever performed. Thanks to computational models, biological data can be analysed in different ways, helping biologists to formulate new hypotheses, drive experimental research and share knowledge on biological processes.

In this paper we introduce an improved modelling approach that allows ~~ANIMO~~ us to considerably increase ~~its~~ ANIMO's performances, opening the way for the analysis of bigger models, which could not otherwise be investigated by human faculties alone. Moreover, this improvement makes the introduction of model checking in ANIMO a more realistic extension, allowing for reduced computation times. The user interface of ANIMO allows to rapidly build non-trivial models and check them against properties formulated in a human-readable language, making modelling an even more powerful support for the biological research.

* These authors contributed equally to this work.
** Corresponding authors.

1 Introduction

To study the possible causes of a disease and design effective cures it is necessary to closely study the behavior exhibited by biological cells under particular conditions. A *signaling pathway* describes the chain of interactions occurring between the reception of a *signal* and the response with which the cell reacts to such signal. A signal is typically ~~restricted~~ represented by a substance which can bind to specific receptors on the cell surface, activating them. *Active* molecules relay the signal inside the cell by activating other molecules until a target is reached. The target of a signaling pathway is usually a transcription factor controlling the production of some protein. Such regulation is considered to be the response of the cell to the received signal. ~~Experimental evidence has shown~~

The current knowledge on signaling pathways (mostly organized in databases such as KEGG [1] or PhosphoSite [2]) suggests that the interactions ~~leading to~~ involved in a cellular response assume more often the shape of a network than that of a simple chain of signal relays. Such networks are typically highly connected, involving feedback loops and crosstalk between multiple pathways, making it difficult for the human brain to understand their dynamic behavior. ~~Tools~~ For this reason, a computational support is required when studying non-trivial biological networks.

A number of software tools are available for ~~developing models of~~ modeling complex networks of biochemical interactions [3–7]. ~~However, hardly ever can these tools be used to design a signaling network, keeping a clear overview while defining the single interactions. In fact, the typical modeling tool allows the user to define a set of reactants and reactions, lacking the visual representation of a complete network in the node-edges form with which biologists usually represent a signaling network. Moreover, technical knowledge is typically very useful, or even required, when using tools based on formal methods to model a biological system, and this tends to limit the user base of such tools~~ These tools significantly contribute to the process of formalizing the knowledge on biological processes, rendering them amenable to computational analysis. However, a lack of familiarity with the formalisms underlying many available tools hampers their direct application by biology experts. ANIMO (Analysis of Networks with Interactive MOdeling, ~~[8, 9]~~ [9–11]) is a software tool based on the formalism of Timed Automata ~~[12] with the aim of supporting~~ [13] that supports the modeling of biological signaling pathways by adding a dynamic component to traditional static representations of signaling networks. ANIMO allows to compare the behavior of a model with wet-lab data, and to explore such behavior in an intuitive and user-friendly way~~, by adding a dynamic component to traditional static representations of signaling networks~~. In order to make the tool easily accessible to biologists, the complexity of Timed Automata is hidden *under the hood*, presenting ANIMO as a plug-in to Cytoscape [14], a tool specifically developed for visualizing and elaborating biological networks. Signaling networks are represented in ANIMO using the nodes-edges notation normally used in the context of biology (see Fig. 1a).

Currently, ANIMO supports interactive exploration of ~~simulation-based dynamics~~ network dynamics based on simulation runs. Model checking queries can be answered through the UPPAAL tool [15], but the required knowledge of temporal logic together with the usually long response times slows down the investigation process. In order to pave the way for a widespread use of model checking on non-trivial models of signaling networks, we propose here a new Timed Automata model that marks a relevant improvement in terms of performances with respect to the model currently used in ANIMO. Moreover, consistently with the intents of our tool, we present also ~~an~~ a user interface for the definition of model checking queries in ANIMO. This will allow a user to interrogate an ANIMO model without previous experience in temporal logics.

The rest of the paper is organized as follows: Section 2 introduces the basic aspects of signaling pathways, Timed Automata and ANIMO; Section 3 illustrates a new way of using Timed Automata to model activity-based networks; Section 4 shows a comparison between the new modeling approach and the one ~~currently~~ previously used in ANIMO, focusing on model ~~checking~~ analysis performances; Section 5 describes the interface additions to make model checking accessible to ANIMO users; Section 6 concludes the paper, discussing future work.

2 Preliminaries

2.1 Signaling pathways in biology

A signaling pathway is ~~the~~ an abstract representation of the reactions occurring inside a biological cell when, e.g., a signaling substance comes in contact with the cell surface receptors. In this setting, a reaction is the interaction between two components: the upstream enzyme (the molecule holding the active role in the reaction) and the downstream substrate (the passive molecule). The enzyme can be for example a kinase, which attaches a phosphate group to its substrate, performing a phosphorylation: this determines a change in the shape of the substrate and consequently a new function (Fig. ??1). The new state reached by the substrate is often called *active*: if the substrate of the reaction is itself a kinase, it can then proceed in passing on the signal by activating its own target molecule, continuing a chain of reactions leading to the target of the signaling chain. Such target is usually a transcription factor, i.e. a molecule that influences the genetic response of the cell, for example promoting the production of a particular protein.

Pathways are traditionally represented in a nodes-edges form (see Fig. ??1a), with nodes representing molecular species and arcs standing for reactions, where $\rightarrow$ represents activation and $\dashv$ represents inhibition (i.e. inactivation).

~~Experimental evidence shows~~ The current knowledge on signaling pathways [1, 2] evidences the fact that signaling interactions ~~often~~ are rarely a simple chain of activations as represented in Figure 1a. More often, they assume the shape of a network ~~instead of a simple chain of activations, with~~ with multiple feedback loops and

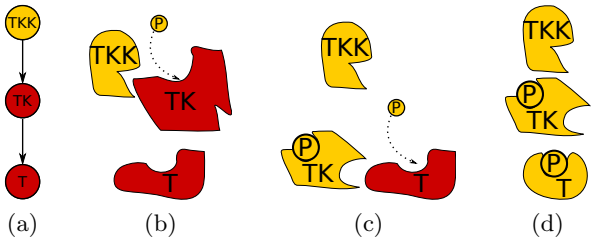


Fig. 1. ...

crosstalk from different signaling sources. This ~~complicates~~ complexity is an added difficulty for the study of such networks, ~~as it is often difficult~~ reducing the possibilities for the human brain alone to deduce the dynamic behavior of a signaling network by inspecting its static representation. For this reason, an efficient computational support is essential when representing and analyzing the behavior of complex signaling networks.

2.2 Timed Automata

~~Timed Automata~~ Timed Automata (TA) are finite-state automata enriched with real-valued clocks and synchronization channels. All clocks in a TA system advance with the same rate, and transitions between the locations of an automaton depend on conditions on clocks. In particular, a *guard* defines when a transition may be taken, while an *invariant* is the condition for permanence in a location. A transition can also allow two automata to *synchronize*, with each participant performing one of two complementary actions (*input* and *output*) on a ~~communication~~ synchronization channel. A set of clocks may also be reset by a transition, causing them to restart counting from 0.

~~A TAis a tuple $\langle L, l_0, C, S, E, I\rangle$, where L is a set of locations, $l_0 \in L$ is the initial location, C is the set of clocks, $S = A \times \{input, output\}$ is a set of synchronization channels (each containing an action label and identified as either inputor output), $E \subseteq L \times S \times \mathcal{B}(C) \times 2^C \times L$ is a set of transitions between locations with a synchronization channel, a guard and a set of clocks to be reset, and $I : L \to \mathcal{B}(C)$ assigns invariants to locations.~~

~~$\mathcal{B}(C)$ is the set of boolean formulas defining bounds on clock values. For example, taken $x, y \in C$, the expression $x \leq 10 \wedge y > 4$ is an element of $\mathcal{B}(C)$.~~

The models we will present here were implemented using the software tool UPPAAL [15], which adds a number of features to the basic definition of TA. Some of these extensions include: support for integer variables in addition to clocks, broadcast ~~communication~~ synchronization channels (one sender, many receivers), definition of C-like functions to perform more operations ~~other than~~ besides clock resets. UPPAAL also allows for a special type of locations, named *committed* (marked with a C in the graphical representation). As long as an automaton is in a committed location, time is not allowed to flow. This feature

can be used for example to perform immediate updates to local variables before letting the computation proceed.

Examples of the listed features can be found in the TA model described in Section 3.2.

2.3 Activity-based models in ANIMO

ANIMO allows the definition of *activity-based* models. This means that we assume each signaling molecule in a cell to be at any time in one of two states: active or inactive. Active molecules can take an active role in reactions, changing the state of other molecules, activating inactive molecules or inhibiting (i.e. ~~de-activating~~deactivating) active molecules. In a kinase-based signaling network an activation process can be a phosphorylation, and it is usually countered by the corresponding dephosphorylation. However, our models are not limited to kinase networks: other features like different post-translational modifications or gene promotion can be likewise represented, as long as their role has immediate effects on the ability of a target to perform its task.

A model in ANIMO is defined through the Cytoscape plug-in user interface (see Fig. 2), where the user inserts a node for each molecular species and an ~~arrow~~ edge for each reaction, with $\rightarrow$ indicating activation and $\dashv$ indicating inhibition similarly to traditional representations of signaling networks (~~see~~ Fig. ~~??~~). 1a).

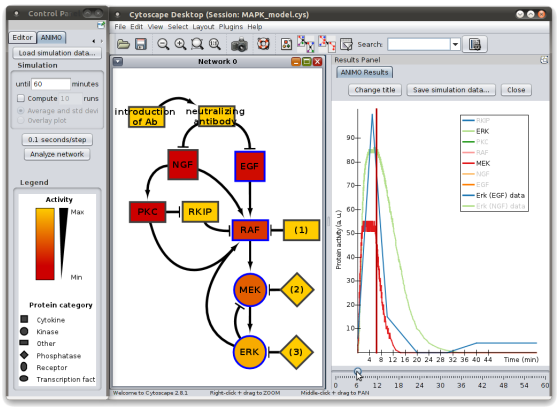


Fig. 2. ...

We consider a *molecular species* (also called *reactant*) to include all the molecules of the same substance in both their active and inactive state inside the cell. In order to distinguish between the two activity states in which each molecule can be, we define the *activity level* to represent the percentage of active molecules over an entire molecular species. In an ANIMO model, this value is discretized on a given integer interval, to adapt to the quality of available

experimental data and (for large models) allow for a trade-off between performance and precision. The user can choose the granularity for each molecular species separately, on a scale between 2 (the reactant is seen as either completely inactive or completely active) and 101 levels (allowing to represent activity as $0, 1\%, 2\% \ldots 100\%$). ~~This choice is made via the user interface as shown in Figure ??.~~ The activity level of a molecular species is represented in the ANIMO network by coloring the corresponding node according to the scale shown in the Activity legend in Figure 2, where the minimum indicates that all molecules of the given species are inactive.

~~The Cytoscape user interface running the ANIMO plug-in (image from [8]). The *Network* panel in the center contains the nodes-arrows model of the crosstalking signaling pathways of NGF and EGF, with colors indicating node activity levels and shapes representing different protein categories (see the *Legend* on the left). The *Results Panel* on the right contains a graph plotting activity levels of selected nodes during the first hour of evolution of the model. The slider under the graph allows the user to select the time instant (marked as a vertical red line in the graph) on which the colors nodes in the *Network* are based. The *Edit reactant* dialog: it is possible to assign to each reactant a name, molecule type (changing how the node is displayed to the user), granularity and initial activity level. The *Edit reaction* dialog: the user chooses a kinetic scenario, sets a value for the kinetic constant k (either quantitative or qualitative), and defines whether the reaction is activating or inhibiting.~~

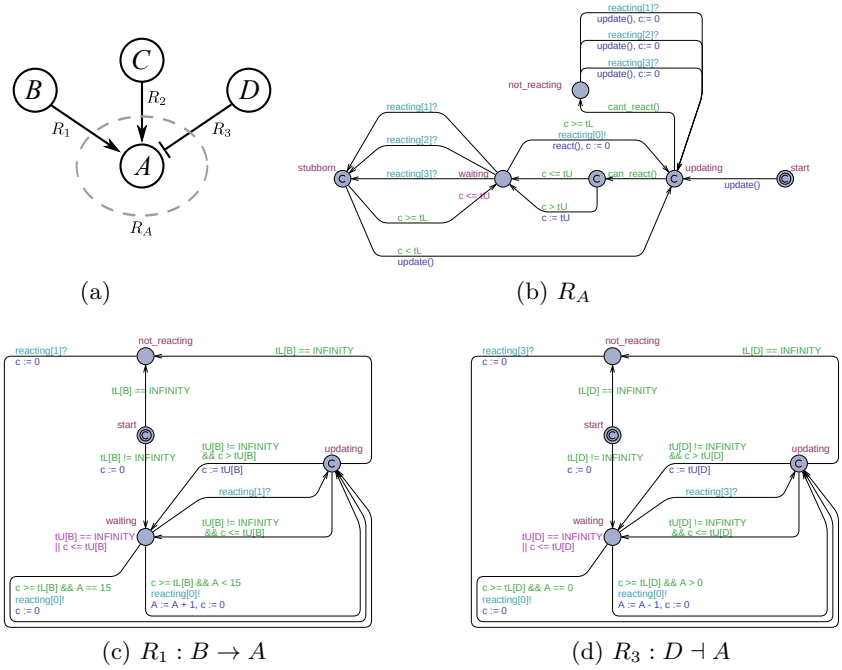
The occurrence of a *reaction* modifies by one discrete step the activity level of its target reactant, making it increase or decrease depending on whether the reaction is defined, respectively, as activating or inhibiting. The rate with which a reaction occurs depends on a formula selected by the user~~via the dialog represented in Figure ??~~. Choosing one of ~~the~~ three available scenarios allows the user to make the reaction rate depend on the activity of one or two reactants. The rate of each reaction can be scaled by modifying the value of ~~the~~ one kinetic constant k, possibly using a qualitative measure from a predefined set (very slow, slow, medium, fast, very fast~~, see Fig. ??~~). Approximating the actual reaction dynamics with less detailed single-parameter scenarios leads to a kinetic model that is still precise enough to faithfully represent the behavior of a signaling network. The approximation allows us to reduce the dependence of a model from often unavailable quantitative parameters for biochemical reaction kinetics. For a more precise dissertation on how reaction rates are computed in ANIMO, we ~~refer the interested reader to [8]~~ recommend [10] , where the ~~currently~~ previously used TA model is presented.

The reader interested in the current methods for parameter setting in ANIMO can refer to [16] .

3 A new way of modeling signaling pathways with TA

We propose here a novel ~~TA~~model to represent signaling pathways ~~.~~ with TA. We define the model ~~currently~~ previously used in ANIMO to be *reaction-centered*,

as for each reaction in the network an instance of a TA template is generated to ~~mimick~~ mimic the occurrences of that reaction~~, whose dynamics depend on its upstream reactant(s) and whose influence changes the activity level of its downstream reactant~~. Observing that signaling networks tend to be highly connected, containing noticeably more reactions than reactants, we propose to shift the focus on reactants instead, achieving what we call a *reactant-centered* model. This change of view is inspired by the classical way in which biological events are modeled with ordinary differential equations (ODEs) [17] .



(a) (b) R_A

(c) $R_1 : B \to A$ (d) $R_3 : D \dashv A$

Fig. 3. ...

3.1 Reactant-centered model

The reactant-centered model presented here is based on the concept of *net effect* of a set of reactions on a reactant: instead of considering each reaction in isolation, we consider their combined influence on each reactant. As an example, consider a reactant A activated by reactions R_1 and R_2, and inhibited by reaction R_3 (see Fig. 3a). The net effect of these three reactions on A defines the *net reaction* $R_A = R_1 + R_2 - R_3$. Applying a concept similar to the definition of an ~~ordinary differential equation~~ODE, where the rate of change of each reactant

depends on the rate of the reactions influencing it, the rate of R_A is computed as the sum of the rates of the reactions influencing A:

$$r_A = r_1 + r_2 - r_3$$

where r_i is the rate of reaction R_i and is defined as follows. Consider R_1 to be ~~defined as~~ the reaction $B \to A$ with kinetic constant k_1. Suppose the settings in the ANIMO network for R_1 make its rate depend only on the activity level of B. Then we compute the rate of R_1 as ~~$r_1 = B \times k_1$~~ $r_1 = [B] \times k_1$, with $[B]$ the current activity level of B.

As with ODEs, if r_A is positive the activity level of A will increase; otherwise, A will decrease its activity level. The absolute value of r_A determines the speed with which such change happens. The value of the reaction rate is ~~then~~ thus translated into a time value to be used as time bound in the TA representing R_A (see Fig. 3b) by computing

$$T_A = \frac{1}{\text{abs}(r_a)}$$

with $\text{abs}(r_a)$ the absolute value of r_a. In order to represent a natural variability in reaction timing, we relax the exact value of T_A by defining bounds of $\pm$ 5%:

$$\mathsf{tL} = T_A \times 0.95$$
$$\mathsf{tU} = T_A \times 1.05$$

Analogously to the reaction-centered model of [10] , we will call tL the *lower time bound* and tU the *upper time bound*. Note that here, differently from what presented in [10] , we do not define pre-computed tables with all possible values for tL and tU: this is because the number of variables for the computation of a reaction rate can be arbitrarily high, and the size of such tables grows exponentially with the number of variables. In the example of Figure 3, a table to represent the time bounds for reaction R_A would have four dimensions, depending on the activity levels of reactants A, B, C and D. Assuming that all four reactants are discretized on 101 levels, each of the two tables tL and tU would contain $101^4 = 104\,060\,401$ elements. Adding one other reaction $E \to A$ to the pool would make this approach exceed the memory limits of most of the currently available computers, requiring about 78 gigabytes of memory to be stored. For this reason, the reaction rates are computed on the fly (see Section 3.3).

3.2 TA model

The behavior of the TA representing the net reaction R_A is defined as follows: after ~~T_A seconds~~ a number of seconds variable inside the continuous interval $[\mathsf{tL}, \mathsf{tU}]$ (measured with the local clock c), the activity level of A will increase or decrease by one step, depending on the sign of r_A; then ~~a new value~~ new values for r_A ~~and T_A~~ , tL and tU will be computed based on the (possibly) changed conditions.

The process described ~~above~~ in Section 3.1 to compute the ~~value of T_A~~ values of tL and tU is an instance of the function called update() in the TA template of Figure 3b~~(where T_A is called T)~~. The computation of update() is the first step taken when initializing a new model taking the transition from the initial location start (identified by two concentric circles) to updating. Location updating is then used to decide whether R_A can be executed or not (functions can_react() and cant_react() used as guards for the transitions exiting from updating). This decision is based on the computed value for r_A and the current activity level of A: if $r_A > 0$ and A is already completely active (or $r_A < 0$ and A is completely inactive), no update to A can be applied, so the automaton switches to location not_reacting. Moreover, r_A can be 0: this means that the reactions influencing A balance each other to complete cancellation, so also in this case R_A cannot occur and the next location will be not_reacting. Otherwise (i.e. if can_react() is true), the activity level of A can be changed, so the next location will be the one labeled waiting, which is reached after having made sure that its clock invariant is respected (committed location between updating and waiting).

When in location waiting, the local clock c is let run until ~~the time required~~ enough time has passed for the reaction to occur~~is reached~~: this is obtained by using c ~~<= T~~>= tL as ~~invariant for location~~ guard for the transition from waiting to updating. Because of ~~this invariant, the transition from~~ the invariant c <= tU on waiting~~to~~, we ensure that the reaction occurs after no more than ~~updating~~tU ~~is taken as soon as its guard~~ seconds have passed. The transition to ~~c >= T~~updating ~~holds, leading~~ leads to an update to the activity level of A (call to react()), together with the computation of new values for r_A~~and T_A~~, tL and tU (function react() calls update()). Moreover, we see in Figure 3b that a communication is sent on channel reacting[0] (0 is the index assigned to reactant A): this is used to warn the other automata that the activity level of A has just changed, possibly changing the conditions for other reactions.

On the other hand, one of the reactants influencing a reaction on which A depends may change its activity level before R_A can occur: this event may have an influence on the value of r_A, so a re-computation of r_A~~and T_A~~, tL and tU is needed. In the template of Figure 3b, the indexes of the influencing reactants are 1, 2 and ~~3.~~ 3, corresponding respectively to B, C, D in Figure 3a. Whenever a communication on one of the channels corresponding to an influencing reactant is received while in location waiting, the automaton moves to location stubborn. This location was introduced to avoid unwanted non-deterministic behavior[1] in the model. In the example, this could happen when the activity level of B is updated concurrently (in the same instant) with the activity level of A. As B influences A through reaction R_1, any change in the activity level of B causes a change in r_1, thus requiring a recomputation of r_A~~and T_A~~, tL and tU. Due to the interleaving nature of TA semantics, only one automaton may change location for the TA system to change state. This means that, should the activity level of B be changed first, the conditions for R_A would be recomputed, possibly preventing

[1] The term *unwanted* refers to a non-determinism that is not related to actual biological concepts, but to the semantics of the modeling paradigm.

the update to A to occur. Vice versa, should the activity level of A be changed first, both A and B could change their activity levels. In such a situation, the non-deterministic order in which the updates are computed decides the behavior of the model. To avoid this, we ensure that if a reaction can occur in a given instant (i.e. c ~~= T~~>= tL), it will ~~occur without considering~~ not consider the concurrent updates to influencing reactants (in the example, ~~R_A would always occur when~~ having the clock reach ~~c = T~~tL ensures that R_A occurs, independently from any concurrent update to B). Performing a check on the current value of the clock c with respect to the reaction ~~time limit~~ lower time bound allows an automaton to detect the case of a concurrent update, remaining "stubborn" in its resolution of reacting with the current configuration: the guard c >= ~~T~~tL on the transition returning from stubborn to waiting is the same as the guard on the transition from waiting to updating and ~~ensures that the reaction will~~ implies that the transition could occur in the ~~same~~ current time instant. If no conflict is found, i.e. c < ~~T~~tL, a new time limit for the reaction is computed via function update() while moving to location updating. Note that the clock c is not reset when moving from stubborn to updating: should the new conditions only change the value of ~~T~~tL and tU without disabling the reaction, the "partial work" of the reactions is not discarded.

Finally, location not_reacting is used to avoid reacting when not necessary (see the description of cant_react() above), and to respond to changes in those reactants which can influence the value of r_A. Should one of such reactants change its activity level, the conditions are updated and the local clock is reset, moving to updating to check whether the reaction can now occur.

3.3 Computations on the fly

Reaction rates are all computed at run-time by function update(), and are based on the user-chosen scenarios, their kinetic constant k and the activity level of the involved reactants. As such computations require floating-point precision but UPPAAL only provides integer variables and operators, we use a significand-and-exponent notation with 4 significant figures, which allows for an error in the order of 0.1% while avoiding integer overflow in UPPAAL's engine. As an example of such a notation, consider the two floating-point numbers $a = 1.23456$ and $b = 0.00023456$, and their sum $c = a + b = 1.23479456$. The corresponding representations in our integer-based notation will be:

$$\begin{array}{rrcrl}
\text{significand}(a) = 1235 & \text{exponent}(a) = -3 & \rightarrow & a = 1235 \times 10^{-3} & a = \langle 1235, -3 \rangle = 1235 \times 10^{} \\
\text{significand}(b) = 2346 & \text{exponent}(b) = -7 & \rightarrow & b = 2346 \times 10^{-7} & b = \langle 2346, -7 \rangle = 2346 \times 10^{} \\
\text{significand}(a+b) = 1237 & \text{exponent}(a+b) = -3 & \rightarrow & a + b = 1237 \times 10^{-3} & a + b = \langle 1237, -3 \rangle = 1237 \times 10^{}
\end{array}$$

In this case, the number representing the sum $a + b$ differs from the exact result by $c - 1237 \times 10^{-3} = 0.001456$. The interested reader can find the UPPAAL definitions and functions needed to compute rate and time values for the TA templates ~~in Appendix ?? or~~ inside any UPPAAL model file generated by ANIMO with a reactant-centered model type.

3.4 Comparison with previous TA model

With reference to Figures 3b and 3c-d, we propose a qualitative parallel between the reaction- (old) and reactant-centered (new) TA models. The first remark is that equally-named locations have the same function in the automaton: so for example location waiting is used to wait before updating the reactant. The central location is in both cases labelled waiting: there the automaton waits until the time has come for the reaction to occur. The amount of time to be waited has different meanings: in the reaction-centered model,

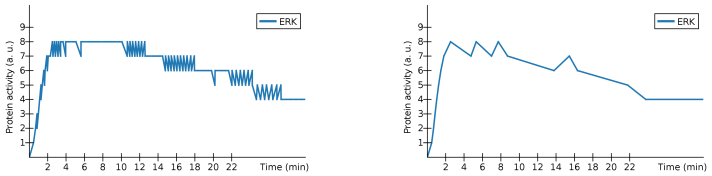
Being based on the same abstractions, the queries the two models can be analyzed with the same UPPAAL queries for all the cases for which biology is involved. The differences can be seen only if the analysis focuses on the single automaton level and explicitly refers to locations: such type of analysis is normally not necessary when considering the model as a whole and using it for what it is, i.e. an abstract representation of a set of biological processes.

We note that replacing all the existing reactions with the net reactions relative to each reactant in a model does change the behavior of the model in the sense that continuous oscillations of activity levels around an equilibrium point are less frequent. However, this does not mean that the behavior of two models using the two different approaches will diverge: the equilibrium points are still the same. Two opposite reactions occurring with the same (or similar) frequency will (nearly) cancel each other out: this behavior is explicitly represented in the reaction-centered model and gives rise to more oscillations. On the other hand, in the reaction-centered model, the trajectory of a reactant's activity level is determined taking into account also the cancelling effect of opposite reactions, so the simulations will generally contain less oscillations. From the user's point of view, graphs will look smoother while still exhibiting the same qualitative behavior, as can be seen in Figure 4. ~~Exhibiting less oscillations means generating less events (i.e. changes of state) in the underlying TAmodel, as in both approaches each change in reactant activity level is caused by an event in the~~ The TA ~~model. Less events means smaller state spaces, thus lower analysis times~~templates in Figure 3 show that the main difference between the two approaches is indeed due to the choice of lumping all the reactions influencing A into the single reaction R_A. In the next session we will show that having less oscillations leads in practice to better analysis performances.

4 ~~Comparison between reaction-centered and~~ Reaction-centered VS reactant-centered~~models~~

We will now apply some basic model checking queries to the case study presented in ~~[8]~~ [10] , measuring the performances of the two modeling approaches. This will allow us to evaluate the benefit brought by the shift in perspective from a reaction- to a reactant-centered model.

All experiments were carried out on an Intel® Core™ i7 CPU at 2.80GHz equipped with 4 Gb RAM and running Ubuntu GNU/Linux 12.10 64bit. UPPAAL version 4.1.11 was used to compute the result of the queries, asking for



(a) Reaction-centered (b) Reactant-centered

Fig. 4. Comparison of two ERK activity graphs generated from the model based on the case study from ~~[8]~~ [10] with treatment of 50 ng/ml NGF. ERK was set to 10 levels of granularity to evidence the oscillations in the graphs.

"some trace" with random depth-first search order when an execution trace was expected to be produced. For the simulation queries using the statistical model checking engine, we left all options at their default values.

The case study we use as a testbed is the network model shown in the *Network* panel in Figure 2, which represents signaling events downstream of growth factors EGF (epidermal growth factor) and NGF (nerve growth factor) in PC12 cells (a cell line used to study neuronal differentiation). The model topology proposed in [18] was ~~analysed~~ analyzed with an ANIMO model based on the reaction-centered approach, reproducing the experimentally observed ERK (extracellular signal-regulated kinase) activity changes ~~[8]~~ [10] . In particular, a 10 minutes stimulation with EGF resulted in transient behavior (i.e. peak-shaped, see also the graph in Fig. 2), while NGF stimulation led to sustained activity (see Fig. 4).

4.1 Simulation cost

We start by evaluating the cost of simulation with the two different models. This is a particularly important aspect to consider, as during the model building phase ~~an~~ a user may need to perform a large number of simulations, continuously adapting the topology or quantitative parameters of a network model. In order to make the modeling approach in ANIMO as interactive as possible, it is desirable to decrease dead times, and this translates into reducing the computational cost of model analysis as much as possible. ~~Applying~~ UPPAAL's statistical model checking engine~~, we ask~~ [19] is based on the generation of random walks (i.e., simulation runs) from a TA model. In this experiment, we query UPPAAL for simulation runs on the models generated by ANIMO applying the reaction- and reactant-centered modeling approaches to the case study. ~~The starting condition we consider is~~ To define the initial state of the model, we consider the starting condition to be the treatment with 50 ng/ml NGF, which translates into setting the activity level of node NGF to 15/15, while changing EGF to be at 0/15 activity. We have chosen this configuration as the model is expected to continue reacting indefinitely (sustained ERK activity), without reaching a deadlock state where all reactions are inactive. This makes the model more demanding when generating the state space because having more reactions occur means having

more states in the transition system underlying the TA model. Table 1 illustrates the computation time and memory usage when performing 100 simulation runs on each of the two considered models. Computing the simulation runs took about 92% less time with the reactant-centered model, using 87% less memory. This decrease in computation time for long simulation runs brings the approach nearer to the idea of interactive exploration of a network~~, especially considering that reactant-centered models are completely deterministic: thus only one simulation is enough to let the user inspect the dynamic behavior of the model with a given parameter setting~~.

Model type	~~Computation time~~ Time (s)	Memory ~~usage~~ (peak KB)
Reaction-centered	37.14	257,928
Reactant-centered	2.83	33,464

Table 1. UPPAAL processor time and memory usage for reaction- and reactant-centered modeling approaches when computing the query `simulate 100 [36000] { R1, R2, ..., R11 }` on the model from ~~[8]~~ [10] with starting condition NGF = 15/15, EGF = 0/15, corresponding to a treatment with 50 ng/ml NGF. The query asks for 100 time series of the activity levels of all reactants in the model over the first 60 minutes of execution.

4.2 Model checking performances

Next, we set out to test the model checking ~~performance~~ performances on the two versions of the TA model, comparing the execution times and memory requirements for a number of interesting queries:

- (1) and (2): `A[] not deadlock`. The model continues to execute indefinitely (`[]` refers to all possible paths in the transition system of the model, and `A` asks the property to always hold along a path).
- (3): `RKIP < 10 --> ERK >= 40`. After RKIP (Raf kinase inhibitory protein) activity has been lowered, ERK activity increases. As in the model RKIP has 20 levels of granularity and ERK has 100 levels, `RKIP < 10` means that RKIP is less than half active, and `ERK >= 40` means that ERK activity is at least 40%.
- (4): `E<> RKIP < 10`. Find a point when RKIP is low ~~, in order to use it~~ (`<>` asks for the existence of at least one path for which the property holds, while `E` requires the property to hold at least once in a given path). This query is expected to generate a trace, the last point of which will be used as initial configuration for model checking queries (5) and (6).
- (5): `A[] ERK < 70` and (6): `A[] ERK > 35`. Once RKIP activity has significantly decreased, ERK activity is sustained at an intermediate level.

The initial conditions are:

– (1): EGF = 15/15 and NGF = 0/15, all others as the original configuration, corresponding to the treatment condition with 100 ng/ml EGF.
– (2) - (4): EGF = 0/15, NGF = 15/15, all others as the original configuration, corresponding to the treatment condition with 50 ng/ml NGF[2].
– (5) and (6): all activities as in the last state of the trace computed from query (4).

~~The analysis of query (1) returned false and all other queries returned true. In particular, query (1) confirms that under EGF treatment no other activity is observed in the model after the initial peak, while query (2) confirms that with NGF activity continues indefinitely. Moreover, queries (3) - (6) confirm the result of the simulations shown in [8] , with NGF treatment leading to sustained ERK activity. The model checking performance of UPPAAL using the reaction- and reactant-centered model types is shown in Table 2.~~

Query	Reaction-centered		Reactant-centered		Improvement	
	Computation time (s)	Memory usage (peak KB)	Computation time (s)	Memory usage (peak KB)	Time (n-fold)	Memory (n-fold)
(1)	~~127.248~~ 127.25	351 804	~~1.415~~ 1.42	25 716	90	14
(2)	~~167.635~~ 167.64	269 932	~~1.932~~ 1.93	26 112	87	10
(3)	~~150.901~~ 150.90	318 460	~~1.439~~ 1.44	26 136	105	12
(4)	~~2.154~~ 2.15	216 920	~~0.113~~ 0.11	25 316	19	9
(5)	~~470.560~~ 470.56	311 568	~~2.223~~ 2.22	26 648	212	12
(6)	~~453.900~~ 453.90	357 980	~~2.229~~ 2.23	26 652	204	13

Table 2. UPPAAL processor time and memory usage for reaction- and reactant-centered modeling approaches when computing the given queries on the ~~model~~ case study from ~~[8]~~ [10] .

The analysis of query (1) returned false and all other queries returned true. In particular, query (1) confirms that under EGF treatment no other activity is observed in the model after the initial peak, while query (2) confirms that with NGF activity continues indefinitely. Moreover, queries (3) - (6) confirm the result of the simulations shown in [10] , with NGF treatment leading to sustained ERK activity. The model checking performance of UPPAAL using the reaction- and reactant-centered model types is shown in Table 2.

4.3 Analysis of the results

Note that we focus most of our attention on the NGF treatment, as with this configuration the model was hypothesized to keep reacting permanently (sustained

[2] In the laboratory experimental setting, NGF is used at a lower concentration than EGF, but it is still enough to saturate all NGF receptors, which are rarer than EGF receptors.

ERK activity). The results from queries (1) and (2) show that this is indeed the case, as EGF treatment leads to all reactions dying out after a number of steps (i.e., deadlock due to all automata being in the not_reacting location). This means that the queries (3) - (6) deal with a larger state space, leading to presumably lower model checking performances. Requesting a full inspection of the state space as we do when using a query of the type `A[]` ϕ returning true in cases (5) and (6), allows us to indirectly compare the state space size of the two model versions. As the hundreds-fold computation time improvements in Table 2 show, the reactant-centered model produces indeed a noticeably smaller state space, allowing for a higher level of interactivity also when performing non-trivial model checking.

Moreover, our experiments point out that the reactant-centered approach considerably lowers the memory requirements for the model. This is not only due to the absence of possibly large precomputed time tables, which can contain up to ten thousand elements each.[3] Indeed, this point was further investigated by implementing a reaction-centered model which avoids the use of tables and instead makes on-the-fly computations of the time bounds with the same number representation as in the reactant-centered model. This resulted in improved performances in the cases of reachability and simulation-based queries, with memory requirements similar to the ones for the reactant-centered model. However, in all other cases a much larger amount of memory (more than 2 Gb) was used with respect to the table-based implementation of the same model, without leading to appreciable benefits in terms of execution time: in some cases performances were noticeably deteriorated. These findings support the intuitive idea that a reactant-centered approach leads in general to fewer events in a model, ~~thus~~ and to a smaller state space.

Note that while the network we consider for the case study is not particularly large, the gain in performances is still considerable and promising in the perspective of ~~analysing~~ analyzing more complex networks.

5 Model checking ~~interface~~ in ANIMO

In order to allow a non-expert user to profit from the power of model checking, we have implemented a template-based user interface to define queries directly inside the ANIMO Cytoscape plug-in: Figure 5 shows the interface for composing a model checking query in ANIMO. The mappings between user interface templates and actual model checking queries are inspired on the ones proposed in [20], and are shown in Table 3.

~~Moreover, if~~ If the answer to a model checking query contains a (counter-) example trace, the trace is automatically parsed by ANIMO and presented to the user in form of a graph of activity levels, in the same fashion as is normally done with simulation runs. Finally, a button positioned near the time slider under a

[3] In the reaction-centered model, having a reaction depend on two 100-levels reactants leads to the generation of two time tables containing $100 \times 100 = 10\,000$ elements each.

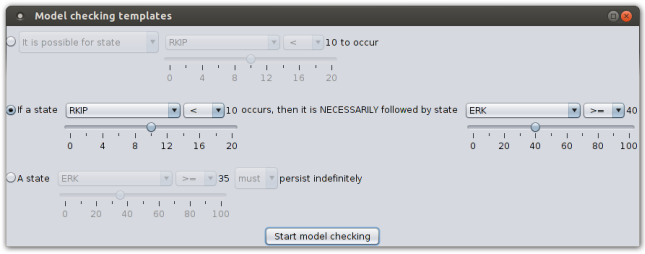


Fig. 5. The interface used in ANIMO to compose a model checking query. The settings on the three lines correspond, from top to bottom, to queries (4), (3) and (6).

ANIMO template	UPPAAL formula
It is possible for state ϕ to occur	`E<>` ϕ
~~It not is possible for state~~ State ϕ ~~to occur~~ never occurs	`E<> !(`ϕ`)`
If a state ϕ occurs, then it is ~~necessarily followed by a state ψ~~ necessarily followed by a state ψ	ϕ `-->` ψ
A state ϕ can persist indefinitely	`E[]` ϕ
A state ϕ must persist indefinitely	`A[]` ϕ

Table 3. Mapping between queries as presented in ANIMO user interface and the corresponding model checking queries in UPPAAL syntax. State formulas indicated by ϕ and ψ are all in the form ~~$A \bowtie nR \bowtie n$~~, with ~~A~~ R the identifier of a reactant in the model, $\bowtie \in \{<, \leq, =, \geq, >\}$ and ~~$n \in [0, nA]$~~ $n \in [0, g(R)]$ a valid activity level value between 0 and the granularity (number of ~~A~~discrete levels) of R.

simulation graph allows the user to easily change the initial activity levels of the whole network by setting them as in the currently selected time instant. This feature ~~can be used when executing queries~~ was used after executing query (4) ~~and~~ to set the initial conditions for queries (5)-(6)~~in the user interface, making~~. Such an addition makes it easier to inspect the behavior of a network by using a sequence of model checking interrogations.

6 Conclusions and future work

We have presented here how the ANIMO tool was improved to provide an interactive approach to model checking. Thanks to the increased performances of the new reactant-centered modeling approach, we are able to obtain answers to model checking queries in a matter of seconds. Coherently with the user friendliness of the tool, we ensure that the biologist can access the features of model checking without the need to directly deal with TA models. In this way, ANIMO acts as an intermediary between the biologist and a formal representation of biological signaling pathways, letting the experts concentrate on investigating the mechanisms of cellular responses.

In order to enforce the concept of user interaction as a primary focus of the tool, we plan to extend ANIMO by adding support for parameter sensitivity analysis and automatic parameter sweeps. We already allow for only one numeric parameter per reaction, but an automatic way of adjusting such parameters to fit experimental data will allow the user to concentrate more on defining a sensible topology for a pathway. Such an automatic search could be limited by setting specific bounds on reaction rates based on previous knowledge, or on a trade-off with performances. Moreover, inspired by works on automata learning [21], we plan to add also the possibility to automatically derive a network topology based on experimental data and ~~user input~~user-defined restrictions based on previous knowledge.

We aim at widening the available set of model checking queries, in order to allow biologists to perform in silico experiments on an already fitting model and to obtain answers to more relevant questions. This would increase the usefulness of a model as a help to drive wet-lab investigation. In order to ~~do this~~allow for meaningful in silico experiments, we plan to purposefully introduce user-defined non-deterministic parts in our models, which would allow for drug dosage investigations through model checking. This can be done e.g. through the definition of intervals for the values of some reaction kinetic constants, adding considerable uncertainty in the timing of those reactions. A model of this type ~~would provide answers to~~ could then be interrogated with queries like "How much X, Y and Z do we need to provide, and at which time points, in order to obtain targets A, B and C to be active?". More useful results can be achieved through proper application of the statistical model checking part of UPPAAL [19] to an extended version of our model to include realistically significant stochastic behavior. Finally, in order to further improve performances, the applicability a multi-core model checking approach based on the works by Dalsgaard et al. [22] is under study.

7 ~~UPPAAL functions for floating-point operations~~

6.1 ~~Globlal functions and declarations~~

...

References

1. Kanehisa, M., Goto, S.: KEGG: Kyoto Encyclopedia of Genes and Genomes. Nucleic Acids Research **28**(1) (January 2000) 27–30
2. Hornbeck, P.V., Chabra, I., Kornhauser, J.M., Skrzypek, E., Zhang, B.: PhosphoSite: A bioinformatics resource dedicated to physiological protein phosphorylation. Proteomics **4**(6) (2004) 1551–1561
3. Ciocchetta, F., Hillston, J.: Bio-PEPA: A framework for the modelling and analysis of biological systems. Theor. Comput. Sci. **410** (August 2009) 3065–3084
4. Dematté, L., Priami, C., Romanel, A.: Modelling and simulation of biological processes in BlenX. SIGMETRICS Perform. Eval. Rev. **35** (March 2008) 32–39

5. Mendes, P., Hoops, S., Sahle, S., Gauges, R., Dada, J., Kummer, U.: Computational modeling of biochemical networks using COPASI systems biology. Volume 500 of Methods in Molecular Biology. Humana Press, Totowa, NJ (2009) 17–59
6. Tomita, M., Hashimoto, K., Takahashi, K., Shimizu, T.S., Matsuzaki, Y., Miyoshi, F., Saito, K., Tanida, S., Yugi, K., Venter, J.C., III, C.A.H.: E-CELL: software environment for whole-cell simulation. Bioinformatics **15**(1) (January 1999) 72–84
7. de Jong, H., Geiselmann, J., Hernandez, C., Page, M.: Genetic Network Analyzer: qualitative simulation of genetic regulatory networks. Bioinformatics **19**(3) (February 2003) 336–344
8. Schivo, S., Scholma, J., Wanders, B., Camacho, R.A.U., van der Vet, P.E., Karperien, M., Langerak, R., van de Pol, J., Post, J.N.: Modelling biological pathway dynamics with Timed Automata. In: 2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE), Los Alamitos, CA, USA, IEEE Computer Society (2012) 447–453
9. ANIMO. `http://fmt.cs.utwente.nl/tools/animo`
10. Schivo, S., Scholma, J., Wanders, B., Camacho, R.A.U., van der Vet, P.E., Karperien, M., Langerak, R., van de Pol, J., Post, J.N.: Modelling biological pathway dynamics with Timed Automata. IEEE Journal of Biomedical and Health Informatics **18**(3) (2013) 832–839
11. Scholma, J., Schivo, S., Camacho, R.A.U., van de Pol, J., Karperien, M., Post, J.N.: Biological networks 101: Computational modeling for molecular biologists. Gene **533**(1) (2014) 379–384
12. Bengtsson, J., Yi, W.: Timed Automata: semantics, algorithms and tools. Lectures on Concurrency and Petri Nets (2004) 87–124
13. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126** (April 1994) 183–235
14. Killcoyne, S., Carter, G.W., Smith, J., Boyle, J.: Cytoscape: a community-based framework for network modeling. Methods in molecular biology (Clifton, N.J.) **563** (2009) 219–239
15. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a nutshell. International Journal on Software Tools for Technology Transfer (STTT) **1** (1997) 134–152
16. Schivo, S., Scholma, J., Karperien, H.B.J., Post, J.N., van de Pol, J.C., Langerak, R.: Setting parameters for biological models with ANIMO. In André, E., Frehse, G., eds.: Proceedings 1st International Workshop on Synthesis of Continuous Parameters, Grenoble, France. Volume 145 of Electronic Proceedings in Theoretical Computer Science., Open Publishing Association (April 2014) 35–47
17. Daigle, B.J., Srinivasan, B.S., Flannick, J.A., Novak, A.F., Batzoglou, S.: Current progress in static and dynamic modeling of biological networks. In Choi, S., ed.: Systems Biology for Signaling Networks. Volume 1 of Systems Biology. Springer New York (2010) 13–73
18. Santos, S.D.M., Verveer, P.J., Bastiaens, P.I.H.: Growth factor-induced MAPK network topology shapes Erk response determining PC-12 cell fate. Nature Cell Biology **9**(3) (February 2007) 324–330
19. David, A., Larsen, K., Legay, A., Mikučionis, M., Wang, Z.: Time for statistical model checking of real-time systems. In Gopalakrishnan, G., Qadeer, S., eds.: Computer Aided Verification. Volume 6806 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2011) 349–355
20. Monteiro, P.T., Ropers, D., Mateescu, R., Freitas, A.T., de Jong, H.: Temporal logic patterns for querying dynamic models of cellular interaction networks. Bioinformatics **24**(16) (2008) i227–i233

21. Tretmans, J.: Model-based testing and some steps towards test-based modelling. In Bernardo, M., Issarny, V., eds.: Formal Methods for Eternal Networked Software Systems. Volume 6659 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2011) 297–326
22. Dalsgaard, A.E., Laarman, A.W., Larsen, K.G., Olesen, M.C., van de Pol, J.C.: Multi-core reachability for timed automata. In Jurdzinski, M., Nickovic, D., eds.: 10th International Conference on Formal Modeling and Analysis of Timed Systems, FORMATS 2012, London, UK. Volume 7595 of Lecture Notes in Computer Science., London, Springer Verlag (September 2012) 91–106