Rule-Based Modelling of Cellular Signalling

Vincent Danos[1,3,4], Jérôme Feret[2], Walter Fontana[3], Russell Harmer[3,4], and Jean Krivine[5]

[1] Plectix Biosystems
[2] École Normale Supérieure
[3] Harvard Medical School
[4] CNRS, Université Denis Diderot
[5] École Polytechnique

Abstract. Modelling is becoming a necessity in studying biological signalling pathways, because the combinatorial complexity of such systems rapidly overwhelms intuitive and qualitative forms of reasoning. Yet, this same combinatorial explosion makes the traditional modelling paradigm based on systems of differential equations impractical. In contrast, agent-based or concurrent languages, such as κ [1,2,3] or the closely related BioNetGen language [4,5,6,7,8,9,10], describe biological interactions in terms of rules, thereby avoiding the combinatorial explosion besetting differential equations. Rules are expressed in an intuitive graphical form that transparently represents biological knowledge. In this way, rules become a natural unit of model building, modification, and discussion. We illustrate this with a sizeable example obtained from refactoring two models of EGF receptor signalling that are based on differential equations [11,12]. An exciting aspect of the agent-based approach is that it naturally lends itself to the identification and analysis of the causal structures that deeply shape the dynamical, and perhaps even evolutionary, characteristics of complex distributed biological systems. In particular, one can adapt the notions of causality and conflict, familiar from concurrency theory, to κ, our representation language of choice. Using the EGF receptor model as an example, we show how causality enables the formalization of the colloquial concept of pathway and, perhaps more surprisingly, how conflict can be used to dissect the signalling dynamics to obtain a qualitative handle on the range of system behaviours. By taming the combinatorial explosion, and exposing the causal structures and key kinetic junctures in a model, agent- and rule-based representations hold promise for making modelling more powerful, more perspicuous, and of appeal to a wider audience.

1 Background

A large majority of models aimed at investigating the behavior of biological pathways are cast in terms of systems of differential equations [11,12,13,14,15,16]. The choice seems natural. The theory of dynamical systems offers an extensive repertoire of mathematical techniques for reasoning about such networks. It provides, at least in the limit of long times, a well-understood ontology of

L. Caires and V.T. Vasconcelos (Eds.): CONCUR 2007, LNCS 4703, pp. 17–41, 2007.
© Springer-Verlag Berlin Heidelberg 2007

behaviors, like steady states, oscillations, and chaos, along with their linear stability properties. The ready availability of numerical procedures for integrating systems of equations, while varying over parameters and initial conditions, completes a powerful workbench that has successfully carried much of physics and chemical kinetics. Yet, this workbench is showing clear signs of cracking under the ponderous combinatorial complexity of molecular signalling processes, which involve proteins that interact through multiple post-translational modifications and physical associations within an intricate topology of locales [17].

Representations of chemical reaction networks in terms of differential equations are about chemical kinetics, not the unfolding of chemistry. In fact, all molecular species made possible by a set of chemical transformations must be explicitly known in advance for setting up the corresponding system of kinetic equations. Every molecular species has its own concentration variable and an equation describing its rate of change as imparted by all reactions that produce or consume that species. These reactions, too, must be known in advance. Many ion channels, kinases, phosphatases, and receptors – to mention just a few – are proteins that possess multiple sites at which they can be modified by phosphorylation, ubiquitination, methylation, glycosidilation, and a plethora of other chemical tagging processes. About one in a hundred proteins have at least 8 modifiable sites, which means 256 states. A simple heterodimer of two distinct proteins, each with that much state, would weigh in at more than 65,000 equations. It is easily seen that this combinatorics can rapidly amount to more possible chemical species than can be realized by the actual number of molecules involved in a cellular process of this kind. The problem is not so much that a deterministic description is no longer warranted, but rather that the equations, whether deterministic or stochastic, can no longer be written down—and if they could, what would one learn from them?

This difficulty is well recognized. One way out is to use aggregate variables describing sets of modification forms. For example, one might bundle together all phosphoforms of a receptor, regardless of which sites are phosphorylated. This, however, is problematic. First, the choice of what to aggregate and not is unprincipled. Second, the appropriate level of aggregation may change over time as the system dynamics unfolds. Third, the aggregation is error prone, since it has to be done without a prior microscopic description. A further, more subtle, difficulty is that an extensional system of differential equations describes the constituent molecules only in terms of interactions that are relevant in a given context of other molecules. It does not characterize molecular components in terms of their potential interactions that could become relevant if the composition of the system were to change. As a consequence, "compositional perturbations", such as adding a novel molecular component (a drug) or modifying an extant one (to model the effects of knocking out a site or adding a new domain) are virtually impossible to carry out by hand, since they require, again, enumerating all chemical consequences in advance and then rewriting all affected equations.

These problems have led to recent attempts at describing molecular reaction networks in terms of molecules as "agents", whose possible interactions are

defined by rules that specify how a local pattern of "sites" and their "states" is to be rewritten [18,19]. This resembles good old organic chemistry, except that biologists think of post-translational modifications less as chemical transformations (which, of course, they ultimately are) than as state changes of the *same* agent. A phosphorylated kinase is, at some useful level of description, still the same entity - though in a different state - than its unphosphorylated version. Indeed, biologists think of the state of an agent as a specific set of interaction capabilities. The discontinuous change of such capabilities despite an underlying continuity in agent-type hinges on the large size of a protein, which allows for a significant change in hydrophobic and electrostatic dispositions without much changing the protein's overall chemical identity.

A rule may specify, for example, that if site Y996 of protein A is phosphorylated, protein B can bind to it with its site SH2. Since this rule applies regardless of whether A or B are bound to other partners or possess other sites with particular states, it captures a potentially large set of individual reactions between distinct molecular species. The need for spelling out all these reactions was spelling problems for "flat" (extensional) reaction network representations, whereas modifying or extending a reaction system is now as simple as modifying a single rule or merging sets of rules, respectively.

Our stance in this paper is to forgo writing out differential equations, and directly operate at the level of rules defining the interactions among a set of agents. Biological signalling and control processes are, in fact, massively distributed systems, and this has led Regev et al. to propose Milner's π-calculus [20], a minimal language for describing concurrent systems, as a language for modelling biological systems [21,22,23]. Since then, numerous variants of representations emphasizing different types of biological processes have been put forward [19,24,25,26,27]. We shall use the language κ [2,3], as a direct and transparent formalisation of molecular agents and their interactions in signalling networks. Most of the points we shall advance here are, however, independent of any committment to a particular syntax, as long as domain-level modifications and bindings can be represented, and one can condition those on the binding and internal states of the various entities participating to a reaction.

Taking concurrency seriously means understanding the organization of such systems in terms of observables defined from within rather than outside these systems. Time is a particular case in point. In molecular systems, the temporal precedence among events cannot be defined (at first) on physical time, since cells or molecules do not bear watches, let alone synchronized ones. It is well known in concurrency that temporal precedence is a logical relation that gives rise to a partial order, as opposed to a total order. Some events must occur before others can happen, while other events may happen in any sequence, reflecting their mutual independence. Clearly, in any particular physical realization one will observe a particular sequence of events. The issue, however, is to uncover which aspects of that sequence are necessary and which contingent. The issue is to discover the invariant structure underlying all observable sequences. Differential

equations are unable to resolve this causality, precisely because they treat time as global, as if everything proceeded in a synchronized fashion.

In contrast, concurrency approaches have long sought to understand the dependencies that constrain an observable event. The traditional notions of causality developed in concurrent models (such as Petri nets, a language which is equivalent to structure-less reactions [28,29]) can be adapted to κ, and used to clarify how a path toward a specified event has unfolded from initial conditions. It is worth mentioning that κ can be seen as an applied graph-rewriting framework, and as such, belongs to a family of formalisms where the notions of causality, conflict, and the attendent concept of event structures, are well-understood [30]. Similar notions of causality have been derived for π-calculus itself, and some have been put to use in a bio-modelling scenario with an ambition to decribe the inner workings of a pathway by deriving causal traces (aka minimal configurations in the event structure terminology) [31,32]. What we shall use here is a related but subtler notion of *minimal* causal path, or *story*, which seems an appropriate formalization of what biologists colloquially call a "signalling pathway", and may have an interest which is independent of the intended application. Used in conjunction with stochastic simulation, stories can generate insights into the collective properties of rule-based systems. We will show how this works using an EGF receptor signalling model that would be quite large and unwieldy by traditional standards. On the basis of relationships of inhibition (or conflict) between rules, we will also see that one can identify key junctures in the system's dynamics that yield explanatory insights and suggest numerical experiments.

To introduce the framework, we warm up with a simple example of an ubiquitous control motif in cellular signal processing: a futile cycle of enzymatic modification and demodification of a target substrate. In general, we have set for an easy and accessible style of explanation, where definitions are precise but not formal. The interested reader will find it easy to reconstruct the technical underpinnings, as we have given a measure of further technical details in an appendix. It may also be worth mentioning that the notions presented here have been implemented, and in both the short preliminary example and the larger EGF receptor one, we have used those implementations to obtain the various simulations, causal traces, and stories.

2 A Futile Cycle

2.1 Agents and Rules

The κ description of a system consists of a collection of *agents* and *rules*. An agent has a name and a number of labeled sites, collectively referred to as the agent's interface. A site may have an internal state, typically used to denote its phosphorylation status or other post-translational modification. Rules provide a concise description of how agents interact. Elementary interactions consist of the binding or unbinding of two agents, the modification of the state of a site, and the deletion or creation of an agent. This seems limited, but closely matches the

style of reasoning that molecular biologists apply to mechanistic interactions in cellular signalling. While this approach does not address all aspects of signaling (such as compartmentation), it does cover a substantive body of events sufficient for the present purpose.

To develop the main concepts, we start with a system consisting of three agents: a kinase K, a target T with two phosphorylatable sites x and y, and a phosphatase P. We first describe a phosphorylation event by means of three elementary actions and their corresponding rules: (1) the kinase K binds its target T either at site x or y; (2) the kinase may (but need not) phosphorylate the site to which it is bound; (3) the kinase dissociates (unbinds) from its target. For ease of reference, we label rules with a mnemonic on the left. Using a textual notation, we represent internal states as '`~u`' (unphosphorylated), and '`~p`' (phosphorylated), and physical associations (bindings or links) as '`!`' with shared indices across agents to indicate the two endpoints of a link. The left hand side of a rule specifies a condition in the form of a pattern expressed as a partial graph, which represents binding states and site values of agents. The right hand side of a rule specifies (usually elementary) changes to agents mentioned on the left. A double arrow indicates a reversible rule, the name refers to the forward version of the rule say `r`, while the opposite rule is written `r_op`. With these conventions, the phosphorylation process of sites x or y translates into:

```
'KT@x' K(a),T(x) <-> K(a!1),T(x!1)
'Tp@x' K(a!1),T(x~u!1) -> K(a!1),T(x~p!1)
'KT@y' K(a),T(y) <-> K(a!1),T(y!1)
'Tp@y' K(a!1),T(y~u!1) -> K(a!1),T(y~p!1)
```

Note that not all sites of an agent interface need to be present in a rule, eg in the first rule `KT@x`, T's interface does not mention y. Likewise, if a site is mentioned at all, its internal state may be left unspecified, eg in the same first rule one does not say whether site x in T is phosphorylated or not. This is the '*don't care, don't write*' convention, only the information which is conditioning the triggering of a rule needs to be represented.

The action of the phosphatase P, which undoes the action of K, is described by a set of similar rules:

```
'PT@x' P(a),T(x) <-> P(a!1),T(x!1)
'Tu@x' P(a!1),T(x~p!1) -> P(a!1),T(x~u!1)
'PT@y' P(a),T(y) <-> P(a!1),T(y!1)
'Tu@y' P(a!1),T(y~p!1) -> P(a!1),T(y~u!1)
```

It is possible to associate rate constants with each rule, as we shall do later. We refer to this rule set as the Goldbeter-Koshland (GK) loop [33]. It is a frequent motif that appears in many variants throughout cellular signal transduction. Notice how the specification of elementary actions forces us to make our mechanistic choices explicit. The example views phosphorylation of T by K as a distributed mechanism, whereby the kinase lets go of its target before phosphorylating it (or another instance) again, since it cannot remain bound to site x and phosphorylate site y. Other variants of multisite phosphorylation involve a processive

mechanism whereby the same kinase acts sequentially on some or all sites of its target. Further variants still would have to specify whether multiple kinases can be bound to the same target or not.

2.2 The Contact Map

Large rule sets can be difficult to understand, and it is important to have a suite of views that report at a glance information implied by the rules. Such cognitive devices are useful for a piecewise modular construction of the system of interest. They also allow for a modicum of reasoning about the system. A first useful view of the rule set is the *contact map*, which is akin to a protein-protein interaction (PPI) map and is shown in Fig. 1. The contact map is a graph whose nodes are the agents with their interfaces and whose edges represent possible bindings between sites. Potential site modifications are indicated by a colour code. The contact map does not provide causal information, in the sense that it does not specify which conditions must be met for a modification or binding to occur. It only shows possibilities.

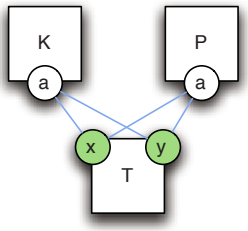


Fig. 1. A Goldbeter-Koshland contact map: nodes represent the three kinds of agents in the rule set, together with their sites, and each site is connected to sites it can bind to; whether a site can be modified is indicated by a colour code (green). Although very simple, the associated rule set generates already 38 non-isomorphic complexes (36 of which contain T).

2.3 Stochastic Simulation

With a rule set in place, one can generate time courses, or stochastic trajectories, for user-defined *observables*. Here we choose an initial state consisting of a 100 of each of the three agents with their interfaces in defined states, `T(x~u,y~u)`, `K(a)`, `P(a)`. We decide to track two observables: (1) the number of doubly phosphorylated target molecules, regardless of whether sites x and y are bound to enzymes, which is written as `T(x~p?,y~p?)`, and (2) the number of target instances T that are fully phosphorylated but free on both sites x and y, `T(x~p,y~p)`. As can be verified in Fig. 2, the latter observable has to be smaller since it is more stringent. The trajectories are obtained using an entirely rule-based version of Gillespie's kinetics which generates a continuous time Markov chain [34]. At any given time a rule can apply to a given state in a number of ways. That number is multiplied by the rate of the rule and defines the rule's *activity* or flux in that state of

the system. It determines the likelihood that this rule will fire next, while the total activity of the system determines probabilistically the associated time advance. This simulation principle can be implemented within κ with rather nice complexity properties, since the cost of a simulation event (triggering a rule) can be made independent on the size of the agent population and depends only logarithmically in the number of rules. This is very useful when it comes to larger systems. One thing to keep in mind is that any obtained trajectory is but one realization of a stochastic process that will differ slightly when repeated. Sometimes, but not always, their average behaviour can be captured in a suitable differential system. Also, and evidently, the trajectories will depend on the choice of rates (see the captions to Fig. 2(a) and 2(b)).

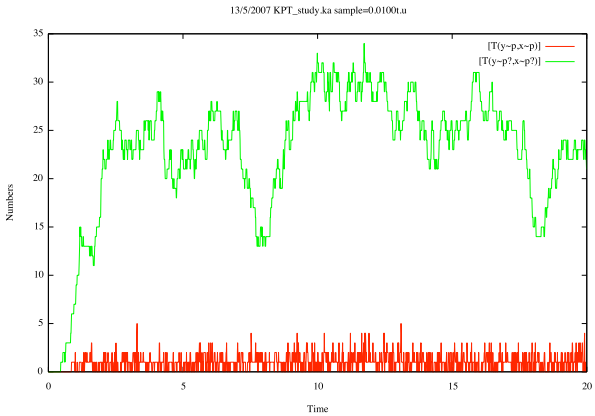
2.4 Precedence and Stories

The trajectory samples obtained above represent an agent-centric view of the system's evolution. For instance, they do not answer directly the question of which succession of events results in a fully phosphorylated form of the target T. This is where the notion of pathway or *story* comes into play. An event, which is the application of a rule, is said to precede a posterior event, if those events don't commute, which can be 1) either because the latter cannot possibly happen before the former, ie has to be after, or 2) because the former can no longer happen after the latter, ie has to be before. Eg an event of type `Tu@x` has to be after an event of type `PT@x`, and before an event of type `PT@x_op`. This notion of logical precedence or causation defines a partial order on any sequence of events along a trajectory. The fact that two events may succeed one another in a particular trajectory does not imply that they are in such a relationship. An example is provided by two successive events of type `KT@x` and `KT@y`. Events that are not related by precedence are said to be concurrent.

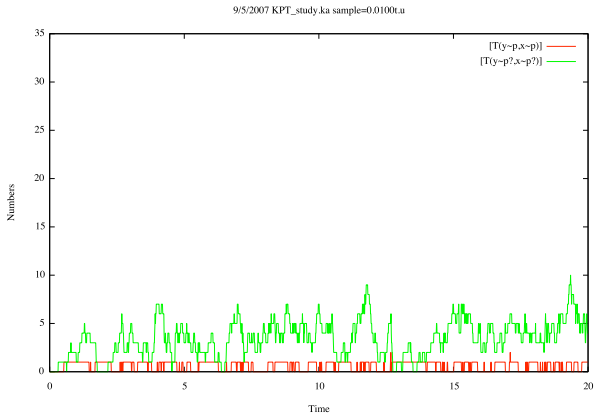
The idea behind a story is to retain only those events in the causal lineage that contributed a net progression towards an event of interest, or in other words a story summarises how a given event type can be obtained. This means in particular that circular histories which generate a situation that is subsequently undone without leaving a side effect that impacts the causal lineage later on should be eliminated. We therefore define a story as a sequence of events that:

- begins with the initial condition and ends with an event of a given type called the observable,
- consists only of events that are in the causal lineage to the observable (which eliminates events that are concurrent to the observable)
- contains no event subsequence with the same properties (which in particular eliminates circles).

Taking as an initial condition one instance of each agent, and as an observable the doubly phosphorylated form of the target, one obtains two stories depending on whether K hits x or y first (Fig. 3 shows the former). If the initial condition were to contain more than one K, there would be a third story in which both sites are phosphorylated by different K-agents.



(a) Association and modification rates are set to 1, and dissociation rates are set to 10 (per time units).



(b) Same model perturbed by a tenfold increase in the association and modification rate of P at site x.

Fig. 2. Goldbeter-Koshland loop simulation: the initial state has a hundred copies of each agent, each disconnected and unphosphorylated; the level of doubly phosphorylated T is lower in the perturbed case (right)

2.5 Inhibition and Activation

Say a rule inhibits another one if the former can destroy an instance of the latter. Note that this may not be a symmetric relationship. An example is given by the application of the rules `KT@x` and `PT@x` (symmetric), or the rules `PT@x_op` and `Tu@x` (dissymmetric). Similarly, say a rule activates another one if the former can create a new instance of the latter. An example is `PT@x` which may create a new instance of `Tu@x`.

Superimposing such inhibitions on a story, as in Fig. 3 above, suggest ways in which one can prevent or delay a story's ending. Indeed, numerically, a tenfold

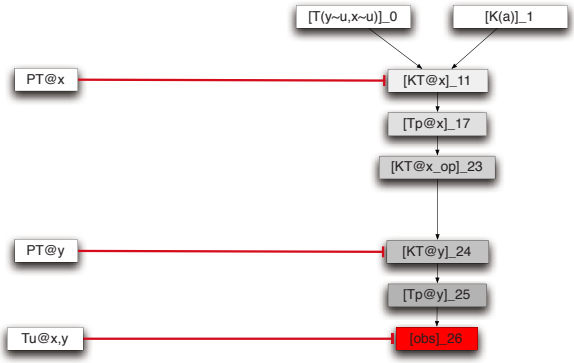


Fig. 3. A story: the story observable is at the bottom, causal depth is represented by shades of grey. Event numbers represent the actual step in the simulation where these events occurred (missing events are either concurrent, or compressible). Various inhibitions attached to the story are shown on the left (explained below).

increase in the rate constants of `PT@x` and `Tu@x` yields a much lower value for the observable (see Fig. 2(b)).

We now proceed to apply these ideas to the more complex example of an EGF receptor model coupled to a MAP kinase cascade. In its full form, as presented in Ref. [35], EGFR signaling is a complex suite of pathways whose boundaries to other signaling systems appear increasingly blurred. While the present model falls far short of representing this complexity, it goes some way towards demonstrating how to eventually represent and face it.

3 The EGFR Model

3.1 EGFR Model Elements

The EGFR signalling network plays an important, yet only partially understood, role in regulating major events in mammalian cells, such as growth, proliferation, survival, and differentiation. In outline, a signal arrives at the cell membrane in the form of a ligand, EGF, which binds to the extra-cellular portion of a special receptor protein, EGFR, that straddles the membrane. With the arrival of EGF, an EGFR becomes capable of binding to a neighbouring EGFR also bound to a ligand. Such receptor pairs can cross-activate one another, meaning that certain of their intra-cellular residues become phosphorylated. These phosphorylated residues now serve as binding sites for a variety of proteins in the cytoplasm. This in turn leads to the activation of a small protein, Ras, that serves as a kind of relay for triggering a cascade of phosphorylations comprising three stacked GK loops in which the fully phosphorylated form of one loop acts as the kinase of the next, causing the overall cascade to behave like an amplifier and culminating in the activation of ERK.

The pathway to the activation of ERK has been the target of an intense modelling effort over the past decade. The ubiquity and importance of the

pathway for biomedical applications [36, Chap. 5] have spurred extensive studies at the mechanistic level of protein-protein interactions and localizations. At the same time, the subtleties uncovered by these investigations (see, for example, the receptor network combinatorics of the pathway [37]) have made it clear that intuition alone, however sharp, cannot confront its complexity and only risks flushing enormous amounts of drug development money down the drain. To calibrate the reader on the magnitudes involved, the particular model presented below contains 70 rules and generates over 10^{23} distinct molecular species (see appendix for the complete and commented rule set). An exhaustive approach with differential equations would, in principle, require that many equations—and that is by no means a large example.

Fig. 4 depicts the model's contact map. The associated rule-based model over-

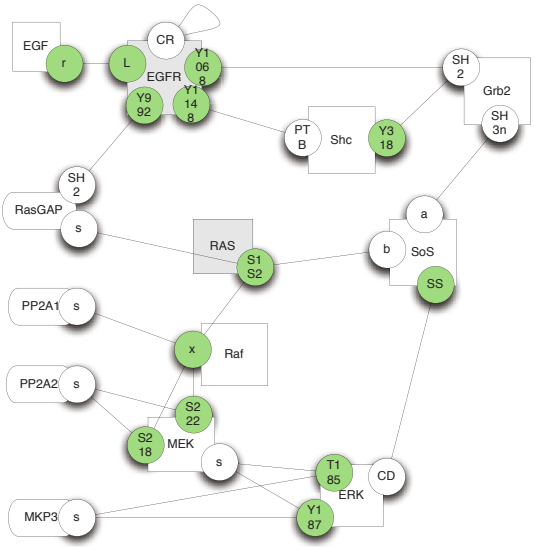


Fig. 4. The contact map of the EGFR/ERK model

lays a causal structure on the contact map by specifying, for example, that Ras can only bind Raf if it has been phosphorylated beforehand at site S1S2. The rule set itself was mainly obtained from refactoring two existing ordinary differential equations (ODE) models [11,12] treating two different aspects of EGF-EGFR signalling: down-regulation of Erk activity through a negative feedback and down-regulation of the signal through internalization.

In its present form, our EGFR model comprises three modules. Firstly, a receptor module, which for illustration purposes retains only the EGFR (or ErbB1) receptor, but includes internalization dynamics using fictitious sites whose state flags localization. This module can be refined to include a receptor heterodimerization network comprising all four receptors of the ErbB family. Secondly, adaptors and relays, such as Sos, Grb2, and Ras. These too can be extended as the

complexity of the model is built up in a stepwise fashion. Finally, we have the module containing the target MAPK cascade.

3.2 Ras Activation

An examination of the contact map immediately reveals a potentially critical role for Ras in the behaviour of the model: Ras has only one site but can bind to three distinct agents (SoS, RasGAP and Raf) and so probably forms a bottleneck. Inspection of the rules governing these four agents allows us to refine the contact map: Ras's site has an internal state (representing active or inactive), only SoS can bind to an inactive Ras, whereas RasGAP and Raf must compete for active Ras.

In terms of signal propagation, SoS activates Ras so that Ras can in turn activate Raf. RasGAP plays an inhibitory role by deactivating Ras. We can observe this chain of causality by looking at the two stories leading to Ras's activation (Fig. 5 and 6, events are labeled by rule names as given in the appendix). Each of the two stories contains a rule that inhibits the other one, respectively `Shc_Grb2` and `EGFR_Grb2`, a clue that these stories are competing ways to obtain the observable. In the former, Grb2 binds to the receptor directly, in the latter it binds to Shc. Only the former would resist a Shc knock-out.

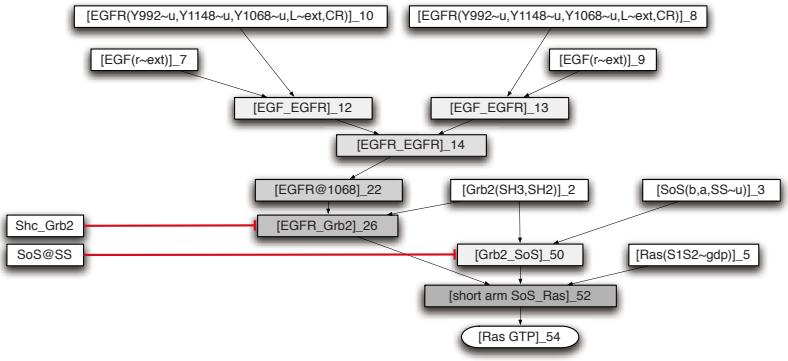


Fig. 5. Short arm activation of Ras (without Shc); inhibitions by `SoS@SS` and `Shc_Grb2` shown on the left; the observable is the activation rule `Ras GTP` (oval shape)

RasGAP does not appear in either of the above stories, confirming that it plays no role in the logical propagation of the signal. RasGAP, however, does play a role in shaping the kinetics of signal propagation. Indeed, most sequences of events leading to Raf's recruitment do exhibit RasGAP intervention (and are therefore not stories). This slightly paradoxical effect of the EGF signal is neatly captured by the *negative feed-forward loop* in Fig. 7 (negative because it has a negative effect on the story, via the rule `direct RasGAP_Ras`, forward because that effect proceeds from a prior event to the story end). In order to propagate the signal, SoS is induced to activate Ras (and hence the downstream cascade to ERK) but, at the

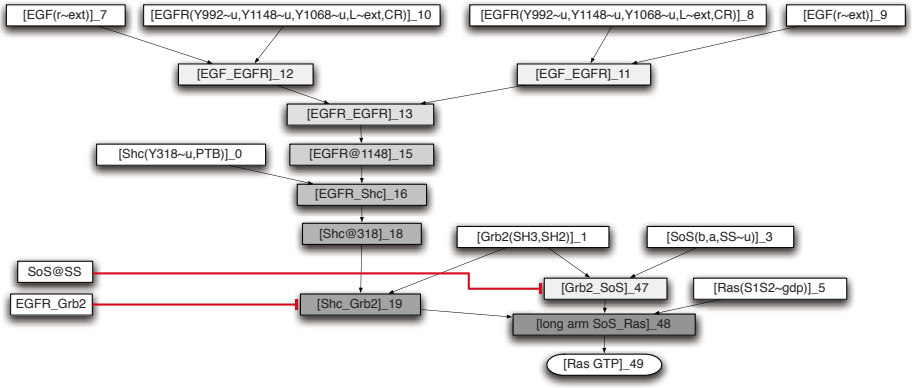


Fig. 6. Long arm activation of Ras (with Shc); inhibitions by `SoS@SS` and `EGFR_Grb2` shown on the left

same time, the signal also induces RasGAP to frustrate SoS's work. Of course, a signal needs to be controlled and eventually down-regulated, so the existence of a RasGAP-like agent should be expected. Both the positive (SoS) and the negative (RasGAP) influences on Ras depend on the same signal which suggests that Ras's activation dynamics only depend on the relative concentrations of SoS and RasGAP: if RasGAP dominates, Ras activation will be weak and short-lived; if SoS dominates, it will be stronger and last longer.

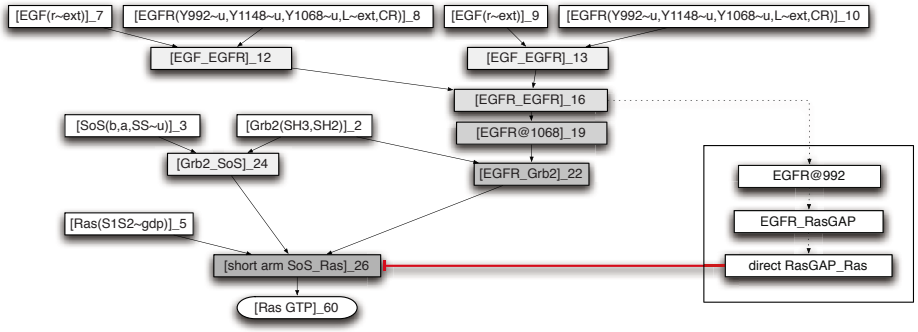


Fig. 7. Battle between SoS and RasGAP; negative feed-forward loop from EGFR dimerisation to Ras activation shown on the right (dotted arrows are activations, the blunted arrow is an inhibition)

3.3 SoS Deactivation

As can be seen in Fig. 5, SoS has a second enemy in the form of activated ERK which by rule `SoS@SS` may inhibit the formation of the complex between Grb2 and SoS by phosphorylating SoS. Fig. 8 shows one of the two stories leading to activated ERK (the other uses the long arm), and there appears a *negative feedback loop*, where the end inhibits an event internal to its own story.

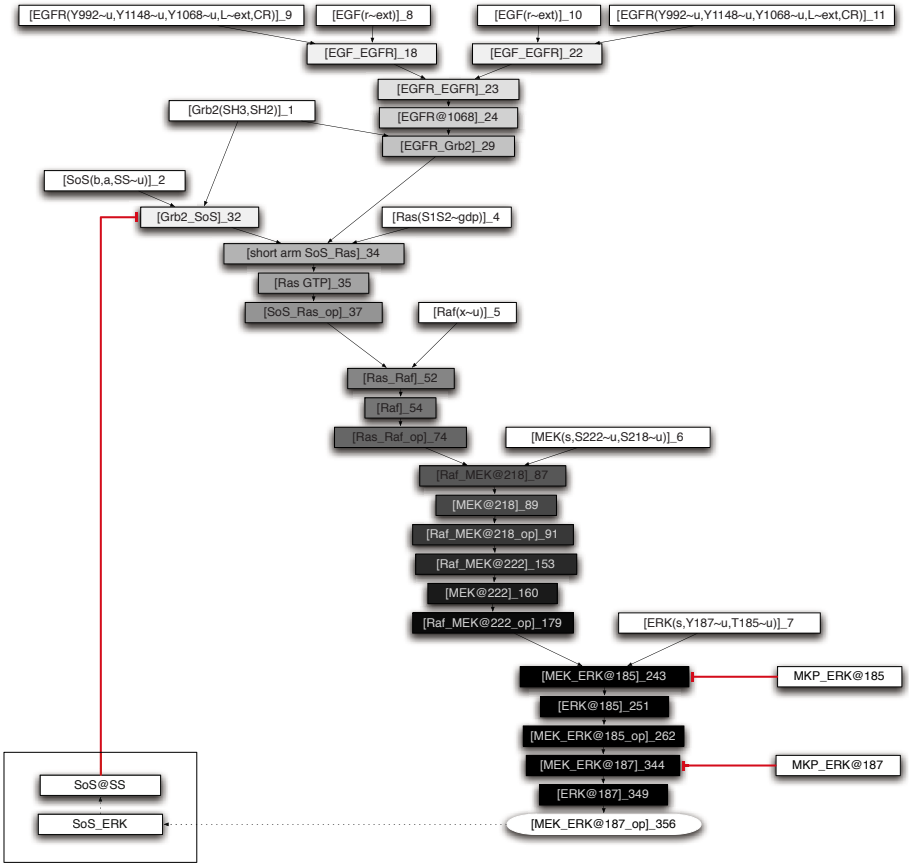
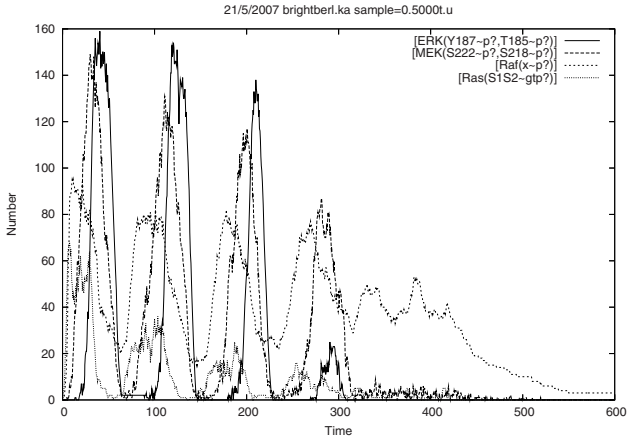


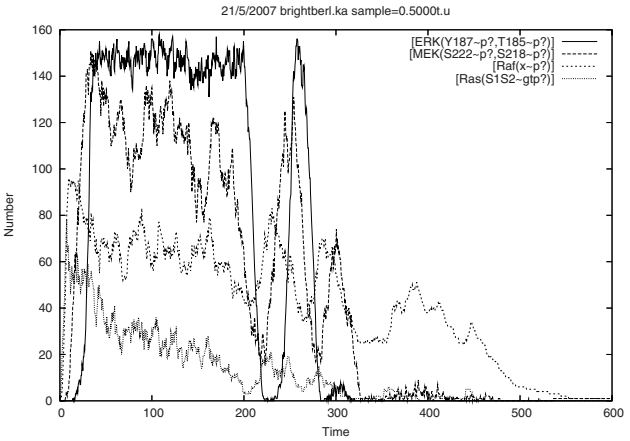
Fig. 8. Short arm story to ERK activation; negative feedback look from active ERK to SoS shown on the left; MKP3 inhibitions shown on the right

For the duration of the SoS's phosphorylation, this substantially weakens the signal from SoS to Ras, essentially shifting the balance in favour of RasGAP. As a result, the level of active Ras decreases which, with a small delay, causes a significant reduction in active ERK at the bottom of the cascade. At that moment, SoS is no longer being strongly targeted by ERK and can, once more, signal to Ras. We thus expect a cyclic process of activation and then inhibition of Ras, leading to an oscillating activation pattern for ERK, typical of a cascade embedded in a negative feedback loop [16].

A crucial parameter determining the shape of these oscillations is the "recovery rate" of SoS from its phosphorylation by ERK. A slow recovery rate leads to a clear oscillation of the cascade. As the rate of recovery increases, the cascade oscillates more quickly, albeit with the same amplitude. With a sufficiently rapid recovery rate, the cascades achieves a transient activation, again with the same amplitude, with a little oscillation as the signal dies (Fig. 9). Thus the qualitative observation embodied in Fig. 8 is rather well echoed at the quantitative level.



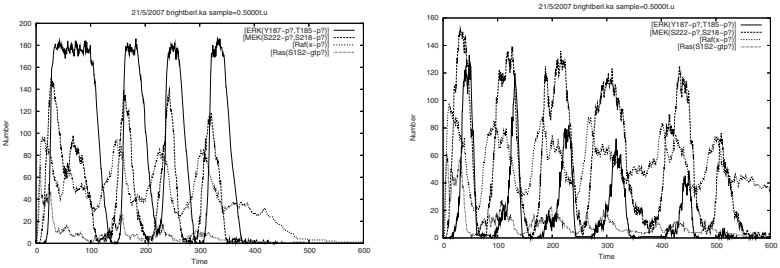
(a) Slow recovery rate of SoS.



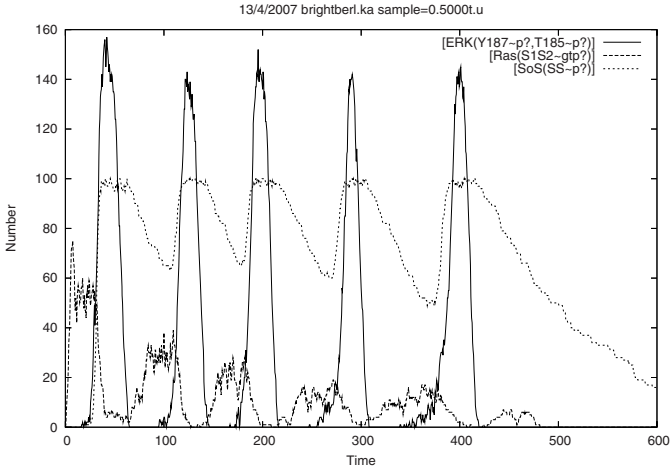
(b) Fast recovery rate for SoS.

Fig. 9. Oscillations and rates of SoS recovery

Another factor regulating the effect of the negative feedback to SoS comes from MKP3, the phosphatase targeting ERK, as suggested by the two inhibitions shown on the right of the activated ERK story (Fig. 8). A higher concentration of MKP3 will tend to hamper activation of ERK and this impacts the rate of oscillation at the cascade: increasing the concentration of MKP3 over successive simulations, one observes (without surprise) that the amplitude of ERK activation decreases. However, one also observes that ERK remains active for less time, leading to a gradual increase in the frequency of oscillation (Fig. 10). In addition, the signal attenuates faster: with more phosphatase in the system, ERK requires a higher "threshold" concentration of its kinase (MEK) in order to achieve significant activation. While this obviously goes against ERK activation, it also protects SoS from being in turn inhibited by ERK. However, this



(a) Low concentration of MKP3. (b) High concentration of MKP3



(c) Combination of fast feedback and slow recovery

Fig. 10. Impact of MKP3 concentration on pathway oscillations

secondary effect turns out to be fairly minor, since a typical system contains more ERK than SoS molecules. Therefore, even a reduced level of ERK activation suffices to mount a powerful inhibition of SoS.

One final parameter substantially influences ERK's level of activation: the speed of SoS phosphorylation by ERK (ie the strength of the `SoS@SS` inhibition shown Fig. 5 and 6). A slow rate limits the effect of the negative feedback, leading to a longer and steadier period of Ras and ERK activation and little to no oscillation. A fast rate accentuates the negative feedback effect, considerably shortening the time during which Ras signals and, in tandem with a slow recovery rate, leads to a pronounced, low-frequency oscillation (Fig. 10).

4 Conclusions

We have illustrated how rule-based modeling transcends the bottleneck of the traditional ODE-based framework. It does so in many important ways, both

practical and conceptual, which we summarize here. First, and this is the starting point, rule-based modeling tames combinatorial explosion by decontextualizing reactions between molecular species into rules defined on patterns. As an immediate corollary, modifications and extensions become fairly straightforward.

Rules represent nuggets of mechanistic knowledge that current experimental practice is rapidly accumulating. Rather than expressing such knowledge in terms of human language or non-executable graphical information, it seems vastly more useful to represent it in a context-free grammar ready for computational consumption. Much like chemical reactions, rules can be viewed as operational "instructions" that can be let loose on a set of molecular agents, driving the unfolding of pathways and their kinetics. In this sense, κ-rules make knowledge executable. The granularity of such rules is, in principle, adaptable to the needs of lab scientists. We believe that the current level of granularity offered by κ or the variant language BNG [18] meets the most urgent practical needs.

Sets of rules are not just inputs to simulators, like systems of differential equations are not just inputs to numerical integrators. Rather, rule sets replace systems of differential equations as formal entities that can be subject to rigorous analysis from which to extract predictive and explanatory information about the behavior of systems. In contrast to the synchronicity of differential equations, rules operate in a concurrent execution model, which is a far more appropriate representation of what is actually going on in cells. This constitutes rather unfamiliar terrain for many biologists, yet this is a turf that has been successfully plowed for over thirty years in computer science. We have shown how notions of conflict and causation can be used to build maps that relate rules to one another. We have defined a concept of story which we believe formalizes the intuitive notion of "pathway" that biologists entertain. Stories are partial orders representing causal lineages that explain how a given observable arises in logical time. Stories change over time, since they depend on available molecular resources. The superposition of stories with rule inhibition maps identifies potential "story spoilers", junctures at which logical structure meets kinetics. We have made extensive use of this trick to explain several dynamical and logical features of a simplified EGFR signalling system. Our experience is that this mode of reasoning matches quite naturally the way biologists intuitively go about telling their "stories". The only difference is that our framework formalizes this process and therefore enables computational procedures to tackle much more complicated systems in rigorous ways when the story-telling of biologists risks degenerating into just that.

It is worth emphasizing that the main dynamic characteristics of our modest EGFR case were obtained with uniform rate constants for all rules. The impact of certain rules on these characteristics was then explored by varying certain rate constants. This is not to say that rate constants don't matter, but it does hint at the importance of the causal architecture of a system in shaping dynamics. Disentangling the contribution of causal structure and rates to overall systems dynamics is hardly possible in large systems of differential equations.

By forgoing this separation, modelers who fit rate constants to ODE systems risk engaging in an idle encoding exercise rather than a modeling process, since many behaviors can be inscribed into any sufficiently large system of ODEs by appropriate choice of rate parameters. We believe that rule-based modeling affords a strategy whereby one first tries to get the logical structure to generate key dynamical characteristics and then tunes the rate constants to obtain the fine structure. If the logical structure is insufficient, the odds are that our knowledge is insufficient and that more experiments would be better than more rate tuning.

It is useful to think of rule-based modeling as a task in concurrent programming, where rules are computational instructions that contribute to system behavior, as in the bigraphical reactive systems [38]. It is difficult to grasp how concurrent systems – in particular natural ones like cells, tissues, and organisms – function and why they function the way they do. Modeling in a rule-based format yields a better appreciation of the role played by individual mechanisms in generating collective behavior. Linking architecture to behavior will produce more informed strategies for intervention in the case of disease and will help us distill the principles that enabled cells to evolve such versatile information processing systems in the first place. As in programming, however, there is ample opportunity for mistakes. In fact, a model might be wrong not because it isn't a correct description of the world, but because it may not express what the modeler intended (think typo). To catch such mistakes is crucial and will eventually necessitate a veritable "modeling environment" with sophisticated debugging and verification tools. The complete absence of such tools in the traditional ODE framework, makes classic models beyond a certain size highly prone to error and exceedingly difficult to maintain. As in programming, rule-based models are "grown" by merging smaller models to build larger models that are easily refined by incorporating new empirical knowledge. Rule-based modeling is as much a scientific instrument as it is effective knowledge management.

References

1. Curien, P.L., Danos, V., Krivine, J., Zhang, M.: Computational self-assembly (submitted) (February 2007)
2. Danos, V., Laneve, C.: Formal molecular biology. Theoretical Computer Science 325(1), 69–110 (2004)
3. Danos, V., Laneve, C.: Core formal molecular biology. In: Degano, P. (ed.) ESOP 2003 and ETAPS 2003. LNCS, vol. 2618, pp. 302–318. Springer, Heidelberg (2003)
4. Faeder, J., Blinov, M.B.G., Hlavacek, W.: BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. Complexity 10, 22–41 (2005)
5. Blinov, M., Yang, J., Faeder, J., Hlavacek, W.: Graph theory for rule-based modeling of biochemical networks. In: Proc. BioCONCUR 2005 (2006)
6. Faeder, J., Blinov, M., Hlavacek, W.: Graphical rule-based representation of signal-transduction networks. In: Proc. ACM Symp. Appl. Computing, pp. 133–140. ACM Press, New York (2005)

7. Faeder, J.R., Blinov, M.L., Goldstein, B., Hlavacek, W.S.: Combinatorial complexity and dynamical restriction of network flows in signal transduction. Systems Biology 2(1), 5–15 (2005)
8. Blinov, M.L., Yang, J., Faeder, J.R., Hlavacek, W.S.: Depicting signaling cascades. Nat. Biotechnol. 24(2), 1–2 (2006)
9. Blinov, M.L., Faeder, J.R., Goldstein, B., Hlavacek, W.S.: A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity. BioSystems 83, 136–151 (2006)
10. Hlavacek, W., Faeder, J., Blinov, M., Posner, R., Hucka, M., Fontana, W.: Rules for Modeling Signal-Transduction Systems. Science's STKE 2006. 344 (2006)
11. Brightman, F., Fell, D.: Differential feedback regulation of the MAPK cascade underlies the quantitative differences in EGF and NGF signalling in PC12 cells. FEBS Lett. 482(3), 169–174 (2000)
12. Schoeberl, B., Eichler-Jonsson, C., Gilles, E.D., Müller, G.: Computational modeling of the dynamics of the map kinase cascade activated by surface and internalized EGF receptors. Nature Biotechnology 20, 370–375 (2002)
13. Orton, R.J., Sturm, O.E., Vyshemirsky, V., Calder, M., Gilbert, D.R., Kolch, W.: Computational modelling of the receptor tyrosine kinase activated MAPK pathway. Biochemical Journal 392(2), 249–261 (2005)
14. Huang, C., Ferrell, J.: Ultrasensitivity in the mitogen-activated protein kinase cascade (1996)
15. Kholodenko, B., Demin, O., Moehren, G., Hoek, J.: Quantification of Short Term Signaling by the Epidermal Growth Factor Receptor. Journal of Biological Chemistry 274(42), 30169–30181 (1999)
16. Kholodenko, B.: Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades (2000)
17. Pawson, T., Nash, P.: Assembly of Cell Regulatory Systems Through Protein Interaction Domains. Science 300(5618), 445–452 (2003)
18. Blinov, M., Faeder, J., Hlavacek, W.: BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. Bioinformatics 20, 3289–3292 (2004)
19. Eker, S., Knapp, M., Laderoute, K., Lincoln, P., Meseguer, J., Sonmez, K.: Pathway logic: Symbolic analysis of biological signaling. In: Proceedings of the Pacific Symposium on Biocomputing. pp. 400–412 (January 2002)
20. Milner, R.: Communicating and mobile systems: the π-calculus. Cambridge University Press, Cambridge (1999)
21. Regev, A., Silverman, W., Shapiro, E.: Representation and simulation of biochemical processes using the π-calculus process algebra. In: Altman, R.B., Dunker, A.K., Hunter, L., Klein, T.E. (eds.) Pacific Symposium on Biocomputing, vol. 6, pp. 459–470. World Scientific Press, Singapore (2001)
22. Priami, C., Regev, A., Shapiro, E., Silverman, W.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Information Processing Letters (2001)
23. Regev, A., Shapiro, E.: Cells as computation. Nature 419 (September 2002)
24. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: An abstraction for biological compartments. Theoretical Computer Science (2003) (to appear)
25. Cardelli, L.: Brane calculi. In: Proceedings of BIO-CONCUR'03, Marseille, France. Electronic Notes in Theoretical Computer Science, Elsevier, Amsterdam (2003) (to appear)

26. Priami, C., Quaglia, P.: Beta binders for biological interactions. Proceedings of CMSB 3082, 20–33 (2004)
27. Danos, V., Krivine, J.: Formal molecular biology done in CCS. In: Proceedings of BIO-CONCUR'03, Marseille, France. Electronic Notes in Theoretical Computer Science, Elsevier, Amsterdam (2003) (to appear)
28. Nielsen, M., Winskel, G.: Models For Concurrency. In: Handbook of Logic and the Foundations of Computer Science, vol. 4, pp. 1–148. Oxford University Press, Oxford (1995)
29. Nielsen, M., Plotkin, G., Winskel, G.: Petri nets, event structures and domains. Theoretical Computer Science 13, 85–108 (1981)
30. Baldan, P., Corradini, A., Montanari, U.: Unfolding and event structure semantics for graph grammars. In: Thomas, W. (ed.) ETAPS 1999 and FOSSACS 1999. LNCS, vol. 1578, pp. 367–386. Springer, Heidelberg (1999)
31. Baldi, C., Degano, P., Priami, C.: Causal pi-calculus for biochemical modeling. In: Proceedings of the AI*IA Workshop on BioInformatics 2002, pp. 69–72 (2002)
32. Curti, M., Degano, P., Priami, C., Baldari, C.: Modelling biochemical pathways through enhanced–calculus. Theoretical Computer Science 325(1), 111–140 (2004)
33. Goldbeter, A., Koshland, D.: An Amplified Sensitivity Arising from Covalent Modification in Biological Systems. Proceedings of the National Academy of Sciences 78(11), 6840–6844 (1981)
34. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem 81, 2340–2361 (1977)
35. Oda, K., Matsuoka, Y., Funahashi, A., Kitano, H.: A comprehensive pathway map of epidermal growth factor receptor signaling. Molecular Systems Biology 1 (May 2005)
36. Weinberg, R.A.: The Biology of Cancer. Garland Science (June 2006)
37. Hynes, N., Lane, H.: ERBB receptors and cancer: the complexity of targeted inhibitors. Nature Reviews Cancer 5(5), 341–354 (2005)
38. Milner, R.: Bigraphical reactive systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 16–35. Springer, Heidelberg (2001)
39. Winskel, G.: An introduction to event structures. In: REX Workshop 1988. LNCS, Springer, Heidelberg (1989)

5 Appendix

5.1 κ, Briefly

In section 2.1 we introduced κ by means of an example. Here we provide some formal definitions to fix concepts more precisely and convey a sense of what we have implemented.

Agents. Let $\mathcal{A}$ be a countable set of names, $\mathcal{S}$ a countable set of sites, and $\mathbb{V}$ a finite set of values. An *agent* is a tuple consisting of an agent name $\lambda(a) \in \mathcal{A}$, a finite set of sites $\sigma(a) \subseteq \mathcal{S}$, and a partial valuation $\mu(a)$ in $\mathbb{V}^{\sigma(a)}$ assigning values to some of the agent's sites, and called the agent's internal state. In the context of signalling, agents are typically proteins (but they need not be).

Solution of agents. By a *solution* we usually have container of molecules in mind. In the case of chemistry proper, agents would be atoms and molecules are atoms connected in particular ways. Likewise, in biological signalling, agents are usually proteins and connected proteins (proteins that are noncovalently bound to one another) are complexes. More precisely, a solution S is a set of agents, together with a partial matching on the set $\sum_{a \in S} \sigma(a)$ of all agent sites. The matching specifies how agents are connected through their sites, but no site can be connected twice. One writes $(a, i), (b, j) \in S$ to express the fact that sites i, j in agents a, b are connected in S.

A solution is essentially a graph whose nodes are agents and whose edges are bonds between agents. Consequently, graph-theoretic notions such as subgraph, connected component, path, etc. apply. We can think of a connected component as a *complex* of proteins – the nodes of the component – bound to one another as specified by the edges of the component.

A *signature* map $\Sigma : \mathcal{A} \to \wp(\mathcal{S})$ is an assignment of a finite set of sites to each agent name. We assume such a signature map to be fixed once and for all, and consider only solutions S such that for all $a \in S$, $\sigma(a) \subseteq \Sigma(\lambda(a))$. An agent a is said to be complete if $\sigma(a) = \Sigma(\lambda(a))$, and likewise a solution S is said to be complete if all its agents are.

Rules. The examples of Subsection 2.1 introduced a rule as a transformation of the graph of agents specified on the left-hand-side (lhs) of the rule into the graph specified on its right-hand-side (rhs). Since the graph on the lhs can occur in many ways within a solution, we refer to it as a "pattern". A solution is a graph, and so the lhs of a rule can be viewed as a solution as well (usually a small one compared to the solution that represents the whole system). A rule then is as a solution together with an action transforming it. The application of a rule to a system means first identifying an embedding of the rule's solution (the lhs pattern) in the solution representing the system and then applying the action to that location. This is made precise in the following by first defining the notion of embedding.

A map ϕ between solutions S and T is an *embedding* if it is an injection on agents, that preserves names, sites, internal states, and preserves and reflects edges; that is to say for all a, $b \in S$, i, $j \in \mathcal{S}$:

$$\begin{aligned}
&\phi(a) = \phi(b) \Rightarrow a = b \\
&\lambda_S(a) = \lambda_T(\phi(a)) \\
&\sigma_S(a) \subseteq \sigma_T(\phi(a)) \\
&\mu_S(a)(i) = v \Rightarrow \mu_T(\phi(a))(i) = v \\
&(a, i), (b, j) \in S \Leftrightarrow (\phi(a), i), (\phi(b), j) \in T
\end{aligned}$$

Hereafter, whenever we write $\phi : S \to T$ we mean to say that ϕ is an embedding, we also write $cod(\phi)$ for the set of sites in T which are in the image of ϕ, and $\mathcal{J}(S, T)$ for the set of all embeddings of S into T.

A *rule* is a pair (S, α), where S is a solution (the left-hand-side in the notation of Subsection 2.1) and an action α over S (the rule action). An *atomic action*

changes the value of some site, or creates/deletes an edge between two sites, or creates/deletes an agent. An *action* on S is a sequence of atomic actions. A rule (S, α) is said to be *atomic*, if α is atomic. It is said to have *arity* n, if S has n connected components, in which case any $\phi : S \to T$ decomposes into n embeddings, one per connected component of S, which we call ϕ's *components*.

The number of all embeddings of the rule's S into the system T, $|\mathcal{J}(S,T)|$, plays an important role in the probabilistic simulation of a system specified by a concrete set of agents and complexes (the solution T) and a set of rules, as sketched in Subsection 2.3.

Causation (activation) and conflict (inhibition) between events (rules). Given an embedding $\phi : S \to T$, we write $\phi(\alpha) \cdot T$ for the result of α on T via ϕ. We say that a site in T is *modified* by $\phi(\alpha)$ if its internal state, or its connections are. A rule set R defines a labelled transition relation over complete solutions:

$$T \longrightarrow^{r}_{\phi} \phi(\alpha(r)) \cdot T$$

with $r = (S(r), \alpha(r)) \in R$, and ϕ an embedding from $S(r)$ into T. An *event* (r, ϕ) consists in identifying an embedding ϕ of the rule pattern $S(r)$ into a complete solution T, and applying the rule action $\alpha(r)$ along ϕ.

Let $\mathcal{E}(T)$ denote the set of events in T. We can define the notions of conflict and causation between events by comparing T with $\phi(\alpha(r)) \cdot T$. Given an event $(r, \phi) \in \mathcal{E}(T)$, we say that (r, ϕ) *conflicts* with (s, ψ), if $(s, \psi) \in \mathcal{E}(T) \setminus \mathcal{E}(\phi(\alpha(r)) \cdot T)$. We say that (r, ϕ) *causes* (s, ψ), if $(s, \psi) \in \mathcal{E}(\phi(\alpha(r)) \cdot T) \setminus \mathcal{E}(T)$.

Unlike in the classical notion of event structure [39], conflict here is not symmetric. It is quite possible that ψ does not conflict with ϕ, while ϕ conflicts with ψ.

The following definition is useful in projecting the definitions of conflict and causation at the level of events to the level of rules, where we refer to them as inhibition and activation, respectively. Given an event $e = (r, \phi)$ in $\mathcal{E}(T)$, the *negative support* of e, written $\lfloor e \rfloor_-$, is the set of sites in T which $\phi(\alpha(r))$ erases or modifies. Similarly, the *positive support* of e, written $\lfloor e \rfloor_+$, is the set of sites in $\phi(\alpha(r)) \cdot T$ which $\phi(\alpha(r))$ creates or modifies.

Using the notion of support, we can formulate necessary conditions for conflict and causation between events. Consider an event $e = (r, \phi)$ in $\mathcal{E}(T)$. If e conflicts with $(s, \psi) \in \mathcal{E}(T)$, then $cod(\psi) \cap \lfloor e \rfloor_- \neq \emptyset$. If e causes $(s, \psi) \in \mathcal{E}(\phi(\alpha(r)) \cdot T)$, then $cod(\psi) \cap \lfloor e \rfloor_+ \neq \emptyset$. At the level of rules, we say that r *inhibits* s if for some T, ϕ, ψ, $cod(\psi) \cap \lfloor e \rfloor_- \neq \emptyset$, and one says r *activates* s if for some T, ϕ, ψ, $cod(\psi) \cap \lfloor e \rfloor_+ \neq \emptyset$.

The notions of inhibition and activation between rules should not be confused with the notions of conflict and causation between events from which they are derived. The relations of inhibition and activation are static relationships between rules and can be computed once and for all for a given set of rules.

5.2 The Rule Set of the EGFR Model

This appendix contains the κ-representation of the Schoeberl et al. (2002) EGF receptor model [12], which introduced receptor internalisation, and combines it with the negative feedback mechanism described in the earlier Brightman & Fell (2000) model [11]. A useful review of these and many other models is provided in Ref. [13]. Refactoring those models in κ involved mining the literature and various databases to obtain the missing domain related information.

Rule names used in the main text and the figures of the paper are defined below. Certain rules use a shorthand '`!_`' notation, to mean that a site is 'bound to something' but the rule does not test anything further than that. This is a convenient way to shorten rules.

Rate constants of rules were set to 1 by default, except for internalisation rules. The numerical experiments summarized in Fig.9 and Fig.10 varied pertinent rate constants as explained above. The rules are presented roughly in the order implied by the ERK activation story shown in Fig. 8. The initial state used in our EGFR simulations is also shown below.

Activating receptor dimers

```
# external dimers:
'EGF_EGFR'  EGF(r~ext), EGFR(L~ext,CR) <-> EGF(r~ext!1), EGFR(L~ext!1,CR)
'EGFR_EGFR' EGFR(L~ext!_,CR), EGFR(L~ext!_,CR) <->
            EGFR(L~ext!_,CR!1), EGFR(L~ext!_,CR!1)
# simplified phosphorylation (internal or external)
'EGFR@992'  EGFR(CR!_,Y992~u) -> EGFR(CR!_,Y992~p)
'EGFR@1068' EGFR(CR!_,Y1068~u) -> EGFR(CR!_,Y1068~p)
'EGFR@1148' EGFR(CR!_,Y1148~u) -> EGFR(CR!_,Y1148~p)
# simplified dephosphorylation (internal or external)
'992_op'    EGFR(Y992~p) -> EGFR(Y992~u)
'1068_op'   EGFR(Y1068~p) -> EGFR(Y1068~u)
'1148_op'   EGFR(Y1148~p) -> EGFR(Y1148~u)
```

Internalization, degradation and recycling

```
# internalization:
'int_monomer'  EGF(r~ext!1), EGFR(L~ext!1,CR) ->
               EGF(r~int!1), EGFR(L~int!1,CR) @ 0.02
'int_dimer'    EGF(r~ext!1), EGFR(L~ext!1,CR!2),
               EGF(r~ext!3), EGFR(L~ext!3,CR!2) ->
               EGF(r~int!1), EGFR(L~int!1,CR!2),
               EGF(r~int!3), EGFR(L~int!3,CR!2) @ 0.02
# dissociation:
'EGFR_EGFR_op' EGFR(L~int!_,CR!1), EGFR(L~int!_,CR!1) ->
               EGFR(L~int!_,CR), EGFR(L~int!_,CR)
'EGF_EGFR_op'  EGF(r~int!1), EGFR(L~int!1,CR) ->
               EGF(r~int), EGFR(L~int,CR)
# degradation:
'deg_EGF'      EGF(r~int) ->
```

```
'deg_EGFR'     EGFR(L~int,CR) ->
# recycling:
'rec_EGFR'     EGFR(L~int,Y992~u,Y1068~u,Y1148~u) ->
               EGFR(L~ext,Y992~u,Y1068~u,Y1148~u)
```

SoS and RasGAP recruitment

```
'EGFR_RasGAP'  EGFR(Y992~p), RasGAP(SH2) <-> EGFR(Y992~p!1), RasGAP(SH2!1)
'EGFR_Grb2'    EGFR(Y1068~p), Grb2(SH2) <-> EGFR(Y1068~p!1), Grb2(SH2!1)
'Grb2_SoS'     Grb2(SH3), SoS(a,SS~u) ->
               Grb2(SH3!1), SoS(a!1,SS~u)
'Grb2_SoS_op'  Grb2(SH3!1), SoS(a!1) -> Grb2(SH3), SoS(a)
'EGFR_Shc'     EGFR(Y1148~p), Shc(PTB) <-> EGFR(Y1148~p!1), Shc(PTB!1)
'Shc_Grb2'     Shc(Y318~p), Grb2(SH2) <-> Shc(Y318~p!1), Grb2(SH2!1)
'Shc@318'      EGFR(CR!_,Y1148~p!1), Shc(PTB!1,Y318~u) ->
               EGFR(CR!_,Y1148~p!1), Shc(PTB!1,Y318~p)
'Shc@318_op'   Shc(Y318~p) -> Shc(Y318~u)
```

Activating Ras

```
# activate:
'long arm SoS_Ras'  EGFR(Y1148~p!1), Shc(PTB!1,Y318~p!2),
                    Grb2(SH2!2,SH3!3), SoS(a!3,b), Ras(S1S2~gdp) ->
                    EGFR(Y1148~p!1), Shc(PTB!1,Y318~p!2),
                    Grb2(SH2!2,SH3!3), SoS(a!3,b!4), Ras(S1S2~gdp!4)
'short arm SoS_Ras' EGFR(Y1068~p!1), Grb2(SH2!1,SH3!2),
                    SoS(a!2,b), Ras(S1S2~gdp) ->
                    EGFR(Y1068~p!1), Grb2(SH2!1,SH3!2),
                    SoS(a!2,b!3), Ras(S1S2~gdp!3)
'Ras GTP'           SoS(b!1), Ras(S1S2~gdp!1) -> SoS(b!1), Ras(S1S2~gtp!1)
'SoS_Ras_op'        SoS(b!1), Ras(S1S2!1) -> SoS(b), Ras(S1S2)

# deactivate:
'direct RasGAP_Ras' EGFR(Y992~p!1), RasGAP(SH2!1,s), Ras(S1S2~gtp) ->
                    EGFR(Y992~p!1), RasGAP(SH2!1,s!2), Ras(S1S2~gtp!2)
'Ras GDP'           RasGAP(s!1), Ras(S1S2~gtp!1) ->
                    RasGAP(s!1), Ras(S1S2~gdp!1)
'RasGAP_Ras_op'     RasGAP(s!1), Ras(S1S2!1) -> RasGAP(s), Ras(S1S2)
'intrinsic Ras GDP' Ras(S1S2~gtp) -> Ras(S1S2~gdp)
```

Activating Raf

```
# activation:
'Ras_Raf'      Ras(S1S2~gtp), Raf(x~u) -> Ras(S1S2~gtp!1), Raf(x~u!1)
'Raf'          Ras(S1S2~gtp!1), Raf(x~u!1) -> Ras(S1S2~gtp!1), Raf(x~p!1)
'Ras_Raf_op'   Ras(S1S2~gtp!1), Raf(x!1) -> Ras(S1S2~gtp), Raf(x)
# deactivation:
'PP2A1_Raf'    PP2A1(s), Raf(x~p) -> PP2A1(s!1), Raf(x~p!1)
'Raf_op'       PP2A1(s!1), Raf(x~p!1) -> PP2A1(s!1), Raf(x~u!1)
'PP2A1_Raf_op' PP2A1(s!1), Raf(x!1) -> PP2A1(s), Raf(x)
```

Activating MEK

```
# activation:
'Raf_MEK@222'      Raf(x~p), MEK(S222~u) -> Raf(x~p!1), MEK(S222~u!1)
'MEK@222'          Raf(x~p!1), MEK(S222~u!1) -> Raf(x~p!1), MEK(S222~p!1)
'Raf_MEK@222_op'   Raf(x~p!1), MEK(S222!1) -> Raf(x~p), MEK(S222)
'Raf_MEK@218'      Raf(x~p), MEK(S218~u) -> Raf(x~p!1), MEK(S218~u!1)
'MEK@218'          Raf(x~p!1), MEK(S218~u!1) -> Raf(x~p!1), MEK(S218~p!1)
'Raf_MEK@218_op'   Raf(x~p!1), MEK(S218!1) -> Raf(x~p), MEK(S218)
# deactivation:
'PP2A2_MEK@222'    PP2A2(s), MEK(S222~p) -> PP2A2(s!1), MEK(S222~p!1)
'MEK@222_op'       PP2A2(s!1), MEK(S222~p!1) -> PP2A2(s!1), MEK(S222~u!1)
'PP2A2_MEK@222_op' PP2A2(s!1), MEK(S222!1) -> PP2A2(s), MEK(S222)
'PP2A2_MEK@218'    PP2A2(s), MEK(S218~p) -> PP2A2(s!1), MEK(S218~p!1)
'MEK@218_op'       PP2A2(s!1), MEK(S218~p!1) -> PP2A2(s!1), MEK(S218~u!1)
'PP2A2_MEK@218_op' PP2A2(s!1), MEK(S218!1) -> PP2A2(s), MEK(S218)
```

Activating ERK

```
# activation:
'MEK_ERK@185'    MEK(s,S218~p,S222~p), ERK(T185~u) ->
                 MEK(s!1,S218~p,S222~p), ERK(T185~u!1)
'ERK@185'        MEK(s!1,S218~p,S222~p), ERK(T185~u!1) ->
                 MEK(s!1,S218~p,S222~p), ERK(T185~p!1)
'MEK_ERK@185_op' MEK(s!1), ERK(T185!1) -> MEK(s), ERK(T185)
'MEK_ERK@187'    MEK(s,S218~p,S222~p), ERK(Y187~u) ->
                 MEK(s!1,S218~p,S222~p), ERK(Y187~u!1)
'ERK@187'        MEK(s!1,S218~p,S222~p), ERK(Y187~u!1) ->
                 MEK(s!1,S218~p,S222~p), ERK(Y187~p!1)
'MEK_ERK@187_op' MEK(s!1), ERK(Y187!1) -> MEK(s), ERK(Y187)
# deactivation:
'MKP_ERK@185'    MKP3(s), ERK(T185~p) -> MKP3(s!1), ERK(T185~p!1)
'ERK@185_op'     MKP3(s!1), ERK(T185~p!1) -> MKP3(s!1), ERK(T185~u!1)
'MKP_ERK@185_op' MKP3(s!1), ERK(T185!1) -> MKP3(s), ERK(T185)
'MKP_ERK@187'    MKP3(s), ERK(Y187~p) -> MKP3(s!1), ERK(Y187~p!1)
'ERK@187_op'     MKP3(s!1), ERK(Y187~p!1) -> MKP3(s!1), ERK(Y187~u!1)
'MKP_ERK@187_op' MKP3(s!1), ERK(Y187!1) -> MKP3(s), ERK(Y187)
```

Deactivating SoS

```
'SoS_ERK'    SoS(SS~u), ERK(s,T185~p,Y187~p) ->
             SoS(SS~u!1), ERK(s!1,T185~p,Y187~p)
'SoS_ERK_op' SoS(SS!1), ERK(s!1) -> SoS(SS), ERK(s)
# feedback creation
'SoS@SS'     SoS(SS~u!1), ERK(s!1,T185~p,Y187~p) ->
             SoS(SS~p!1), ERK(s!1,T185~p,Y187~p)
# feedback recovery
'SoS@SS_op'  SoS(SS~p) -> SoS(SS~u)

%init: 10*(EGF(r~ext))
+ 100*(EGFR(L~ext,CR,Y992~u,Y1068~u,Y1148~u))
```

```
+ 100*(Shc(PTB,Y318~u))
+ 100*(Grb2(SH2,SH3!1),SoS(a!1,b,SS~u))
+ 200*(RasGAP(SH2,s))
+ 100*(Ras(S1S2~gdp))
+ 100*(Raf(x~u))
+ 25*(PP2A1(s))
+ 50*(PP2A2(s))
+ 200*(MEK(s,S222~u,S218~u))
+ 200*(ERK(s,T185~u,Y187~u))
+ 50*(MKP3(s))
```