Rule-based Modelling and Model Perturbation

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

[1] University of Edinburgh
[2] INRIA–ENS–CNRS
[3] Harvard Medical School
[4] CNRS–Université Paris Diderot
[5] IHES

Abstract. Rule-based modelling has already proved to be successful for taming the combinatorial complexity, typical of cellular signalling networks, caused by the combination of physical protein-protein interactions and modifications that generate astronomical numbers of distinct molecular species. However, traditional rule-based approaches, based on an unstructured space of agents and rules, remain susceptible to other combinatorial explosions caused by mutated and/or splice variant agents, that share most but not all of their rules with their wild-type counterparts; and by drugs, which must be clearly distinguished from physiological ligands.
In this paper, we define a syntactic extension of Kappa, an established rule-based modelling platform, that enables the expression of a structured space of agents and rules that allows us to express mutated agents, splice variants, families of related proteins and ligand/drug interventions uniformly. This also enables a mode of model construction where, starting from the current consensus model, we attempt to reproduce *in numero* the mutational—and more generally the ligand/drug perturbational—analyses that were used in the process of inferring those pathways in the first place.

1 Introduction

In recent years, there has been extensive development in the use of modelling to understand cellular signalling networks (see [1, **?**,3, 4] among many others). To date, this line of work has focussed almost exclusively on describing wild-type behaviours, i.e. it deals with the interactions between proteins that take place in a normal healthy cell. This is already highly non-trivial since these signalling networks employ a strategy of binding, modification and unbinding between proteins that generates astronomical numbers of non-isomorphic molecular species. This poses an essentially unsolvable scalability problem for any modelling approach, such as ODE-based chemical kinetics or Petri nets, based on exhaustively enumerating *reactions* between fully-specified molecular species.

In recent years, a new modelling approach has been used to tame this combinatorial explosion, namely agent- or rule-based modelling [5]. In this setting, molecular species are left implicit; instead, *agents* are used to represent not complexes but their constituent proteins. Each type of agent has a name and a set of *sites*. Instead of reactions, we write *rules* that mention names of agent types and some, but not necessarily all, of their respective sites. In this way, and unlike reactions, a rule need only make explicit those aspects of the agents upon which it acts that are actually relevant to the interaction being described by the rule. So reaction-based models leave agents implicit, considering them at best as an aggregation of molecular species, whereas rule-based models make agents explicit but the reactions implicit, instead considering their rules to be aggregations of reactions.

It should be noted that such wild-type models, be they reaction- or rule-based, can already handle situations where, typically as a result of gene amplification or ablation, a protein is either over- or under-expressed. Under such circumstances, the response of a cell to external conditions may be exaggerated or attenuated as a consequence of the induced perturbation of mass-action kinetics and of the nature and numbers of complexes that exist in the cell's resting state (cf. [6]). This does not bring about *new* protein-protein interactions, it only affects the relative importance of the wild-type interactions, e.g. if protein X has a binding partner Y that is over-expressed, X will be attracted to the greater than usual mass of Ys to the detriment of its binding with other partners.

However, many disease states are the result of genetic mutations that build incorrect proteins, with aberrant behaviour, rather than the straightforward modulation of protein expression levels (although in some cases the two defects co-exist and synergize). Such mutant proteins may only differ by one or two amino acids from their wild-type cousins and yet have radically different behaviour, e.g. erbB1 with the single substitution L858R, which exists in many kinds of solid tumour, has a constitutively active kinase domain, as does B-Raf with the single V600E mutation. The flip side of this is that much of the wild-type behaviour of a protein is actually shared with such mutants, for instance a binding domain far from any site of mutation will quite likely retain its usual functionality. This poses a further serious challenge to modelling since mutated proteins therefore duplicate large chunks of an already highly combinatorial wild-type network, while also potentially adding interactions.

To tackle these issues, we introduce a syntactic extension of Kappa that allows the definition of a structured space of agents. Agents can either be declared *ab initio* or derived from existing agents in a manner reminiscent of object-oriented programming (particularly the prototype-based approach). In the latter case, the new agent can gain, lose, rename, mutate or duplicate sites of the agent from which it is derived. This organizes the space of agents hierarchically and thus enables us to write *generic* rules that mention agents that have many descendants in the hierarchy. These generic rules act as shorthand for sets of normal Kappa rules; they capture behaviours *shared* by splice variants (e.g. p46, p52 and p66 Shc), genetically related proteins (e.g. ERK1 and ERK2) or mutated proteins. In particular, the conciseness of generic rules enables us to write and analyze large Kappa models far more easily. We illustrate this with a small generic rule set (15 rules) for the erbB receptor network that, once expanded into Kappa, has over 300 rules and which grows considerably larger still if we add in drug interventions and mutated erbB agents.

In summary, our agent hierarchy allows us to write large models in a comfortable way, to navigate the perturbation space of the model (ligands, mutations and drugs) and investigate the consequences of chosen perturbations, i.e. those for which we have experimental data, with the static and causal analyses of Kappa. This is particularly interesting for mutational perturbations as these enable us to reproduce, *in numero*, biochemical experiments that employ engineered mutations. In this way, our rules—a formalization of the *consensus* pathway assembled by many biochemical experiments—can be tested by checking, *in numero*, whether perturbing them with mutated agents—representing the engineered mutations—matches those experimental results. Of course, this procedure can never "prove" that a rule is correct but it can be used to reject rules that lead to behaviour incompatible with experimental results. It can also point to the existence of missing links in a model if it throws up false negatives with respect to the experimental data, e.g. it predicts some but not all experimentally observed phosphorylated sites. In other words, it enables us to put our assumptions under the microscope and verify that the consensus wild-type pathway behaves as expected when subjected to perturbations—and if it doesn't, we will need to change our consensus model.

Contribution and relation to existing work Rule-based modelling is one branch of a rich literature based on the idea of representing proteins and their interactions as concurrent processes, thereby viewing a signalling network as a kind of massively distributed system. This was initially expressed in the formalism of π-calculus [7, 8] but, since then, a number of variants of π-calculus [9, 10] and of other languages for distributed systems [11–17] have also been proposed for representing various aspects of biological processes, notably the importance of causality and compartments.

Rule-based modelling, rather like the BWB/BlenX system [18], was developed out of these ideas but, instead of being based on some prior formalism for general distributed systems, is a domain-specific modelling language for biological processes. Our language Kappa is particularly closely related to BioNetGen (BNG) [19]. Although the original aims of BNG were rather different—it was conceived as a language for describing systems of ODEs in a higher-level fashion, rather than as a modelling language in its own right—the two approaches have much in common and, in particular, our agent hierarchy proposal would work just as well in BNG as in Kappa.

Despite these many advances, to the best of our knowledge none of the above-cited approaches, including Kappa and BNG, can deal with *all* the potential sources of combinatorial explosion in signalling models. Our extension of Kappa with agent hierarchies directly addresses this problem in the specific context of rule-based modelling. Given that mutating agents, via small changes in their sites and thus interaction capabilities, is central to our proposal, it would be interesting to investigate the possible connections of this work with the recent use of mutations on the structure of BlenX programs in order to evolve networks via genetic algorithms [20]. However, it should be stressed that our work was originally intended to facilitate the construction (and documentation) of large models in a way that makes explicit any underlying uniformities, rather than in directly enabling an evolutionary analysis of networks.

2 Kappa and agent variants

A Kappa [21] model consists of a collection of concrete agents and rules. Each agent, or more properly agent type, has a *name*, an associated set of *sites*, each with an optional internal state, and a *copy number*. An atomic rule falls into one of five classes—a binding between two agents, an unbinding, the modification of an agent, the creation of an agent or the deletion of an agent—but a rule can also be non-atomic, combining several actions.

Given a Kappa model, its *contact map*, which is computed statically from the rules, specifies which agents can bind and on which sites. (See e.g. Figs. 1, 4.) On the other hand its *influence map*, also computed statically, specifies the causal relations of activation and inhibition between rules, that is to say a rule activates (inhibits) another if its application may add (subtract) from the set of instances of the other one. We will make use of the static analysis of rule accessibility [22] which identifies whether a rule is dead, i.e. cannot be applied, or is potentially applicable; in the latter case, we will use the story sampler [23] to extract, from stochastic simulations [24] of the model, the chains of rule firings that can lead to an actual application of the rule. If we find such a story, this confirms that the static analysis didn't produce a false positive.

The concrete syntax we use to present agents, agent variants and rules should be self-explanatory (although we stress that it can be formalized). One key thing to remember, as said earlier, is that in the definition of a rule one has the option of not mentioning some sites of an agent. In situations where agents have up to a dozen different sites (e.g. the members of the EGF receptor family), this is key to obtaining concise models. This, combined with the ability to mention generic agents, allows us to express enough uniformities for also obtaining concise descriptions of perturbed models.

2.1 Agent variants

A variant on an agent always introduces a new name and can arise in several different ways: it can lose or mutate an existing site, gain a new site or rename/duplicate an existing site. To represent these possibilities formally, we need only introduce two perturbation operations on agents, one to add a site, the other to replace a site with a set of sites. The latter operation subsumes site deletion (by replacing a site with the empty set), site renaming (replacing with a singleton set) and duplication.

For example,

```
%gen: A(s,t)
%gen: B = A[+u s\{} t\{t1,t2}]
```

declares the agent A with sites s and t and derives from it an agent B with sites t1, t2 and u. This defines a tree of *agent variants*; most nodes of the tree are labelled ‘gen’ for generic but leaves of the tree can be labelled ‘conc’ for concrete which signals that that agent can be used in a Kappa model. Note that we have a second tree structure that traces site linkages: any site can be traced back to either a site addition or to a site declared ab initio; and conversely, following the linkages the other way, a site in agent A maps to a set of sites in any given descendant agent B (empty if the site has been deleted, singleton if it has just been renamed). This is important for compiling generic rules into a bona fide Kappa model.

Mutation of a site is represented by the compound operation of deleting the original site and, if desired, adding a new site to “replace” it. If the desired result of the mutation is simply the loss of certain wild-type interactions, the loss of the site is enough and no such new site need be added; but sometimes mutations result in new interactions becoming possible in which case we would need to introduce a new site in order to write the new rules expressing the novel interactions of the mutated agent, e.g. the tyrosine kinase inhibitor erlotinib binds to the L858R mutated erbB1 with much higher affinity than to the wild-type receptor.

2.2 A first example

Let us make this more concrete with an example extracted from a larger model of the MAPK cascade. We start with two basic agent types, MAP2K and MAPK, from which we would like to derive some more specific agent types. Our first declarations introduce the starting agents:

```
%gen: MAP2K(D,S~u,ST~u)
%gen: MAPK(CD,T~u,Y~u)
```

Formally, these declarations play a role analogous to that of the axioms in any formal language and, as in that kind of setting, we use them as the starting point to introduce more subtle objects. In this case, we wish to consider the three common kinds of MAPK protein—ERKs, JNKs and p38s—and their respective MAP2K upstream activators—MEKs, JNKKs and p38 kinases.

To do this, we first introduce three variants of MAPK and three of MAP2K:

```
%gen: ERK = MAPK[+FXFP]
%gen: JNK = MAPK
%gen: p38 = MAPK
```

Note that, while ERK gains a new site FXFP, an ERK-specific binding site for immediate early gene products such as Fos and Jun [25], JNK and p38 simply inherit the sites of MAPK without making any changes. As we will see shortly, the introduction of these three variants allows us to express concisely the specificity of binding between these three distinct families of MAPKs and their cognate upstream activators. Note also that these three agents are still generic as they represent families of proteins: ERK covers two proteins (ERK1 and ERK2), JNK covers three (JNK1, JNK2 and JNK3) and p38 covers four (p38alpha/beta/gamma/delta); and several of those proteins have multiple splice variants.

We formalize this by a further layer of variants:

```
%conc: ERK1 = ERK[T\{T202} Y\{Y204}]
%conc: ERK2 = ERK[T\{T185} Y\{Y187}]
```

We show only the case of ERK1 and ERK2 as those of JNK and p38 are completely analogous. Recall that we use the 'conc' tag (rather than 'gen') to make explicit the fact that ERK1 and ERK2 are concrete, not generic, agents and, as such, can be used in a Kappa model.

Note that we have renamed (via singleton duplications) the sites of ERK to include specific information about the exact residue numbers of their phosphorylatable sites; this is not essential, of course, but does illustrate the documentary power of agent variants over and above their role of structuring the space of agents.

We must also introduce generic and concrete variants of MAP2K. Each variant covers two proteins: MEK1 and MEK2 for MEK; MEK4 and MEK7 for JNKK; and MEK3 and MEK6 for p38K. (Again, for the sake of simplicity, we only show the concrete variants of MEK.)

```
%gen: MEK = MAP2K
%gen: JNKK = MAP2K
%gen: p38K = MAP2K

%conc: MEK1 = MEK[S\{S218} ST\{S222}]
%conc: MEK2 = MEK[S\{S222} ST\{S226}]
```

Already, the simple fact of hierarchically structuring the agents under consideration yields a useful object in its own right that documents, in a completely formal way, a significant amount of biological knowledge (about exactly how related proteins relate to each other) that can easily be found in several online databases but which, in that medium, remains informal and purely descriptive, whereas, in this formalized setting, has already been subjected to an initial step of processing and structuring. It also includes a convenient documentation of the specific sites of interest, e.g. the precise identities of phosphorylation sites, that are otherwise rather cumbersome to keep track of.

Moreover, the creation of this agent hierarchy also facilitates the process of writing rules by enabling us to write them at the appropriately generic level. It eases the cognitive burden of writing rules by exposing clearly the similarities and differences between various agent types. More concretely, it allows us to avoid writing essentially the same rule many times for closely related agents and, as such, also eliminates the risk of forgetting cases (a very common mistake when developing large rule sets). We turn to this in the next subsection where we will complete the MAPK example.

2.3 Generic rules

We have seen how we can structure agents hierarchically with concrete agents at the leaves and generic agents above them. In this context, a normal (or *concrete*) Kappa rule is a rule that only mentions concrete agents. A *generic* rule is syntactically just like a normal rule but mentions one or more generic agents. The purpose of such a rule is to be expanded into a set of concrete rules by replacing each generic agent G in the rule with all appropriate concrete agents C below it in the hierarchy. However, this expansion is modulated by the changes made to G's sites in C; notably, if site s of G is deleted in C, then no rule testing the existence of s can instantiate G to C. And we must also use the site linkages between C and G to deal with any renaming and duplication of G's sites in C. So, were we to write the single generic rule

```
MAP2K(D), MAPK(CD) <-> MAP2K(D!0), MAPK(CD!0)
```

this would "incorrectly", i.e. not as we wish, expand to a collection of concrete rules where all concrete descendants (in the agent hierarchy) of MAP2Ks can bind with all concrete descendants of MAPKs, e.g. JNK2 could bind ERK1. This is the reason why, in the previous section, we introduced a second layer of generic agents—ERK, JNK, p38; MEK, JNKK, p38K. Given that, we can write the following three generic rules that properly respect the desired specificity of binding between MAP2Ks and MAPKs.

```
MEK(D), ERK(CD) <-> MEK(D!0), ERK(CD!0)
JNKK(D), JNK(CD) <-> JNKK(D!0), JNK(CD!0)
p38K(D), p38(CD) <-> p38K(D!0), p38(CD!0)
```

These three generic rules expand to eighteen concrete rules if we take ERK1/2, JNK1/2/3 and p38$\alpha/\beta/\gamma/\delta$ as concrete agents. If we included the many splice variants of the JNKs and p38s, the *same* three generic rules would expand to over thirty concrete rules. This illustrates the flexibility of our approach whereby a given generic rule can expand differentially depending on the background of concrete agent variants we select. In particular, a single rule set can be seen as existing at many levels of detail—and this is easily tunable by the modeller as a function of his/her current needs.

There is, however, an associated cost, over and above the obvious need to recompile one's generic rules, when changing the level of detail of a model: under certain circumstances, this will lead to a degradation in the performance of stochastic simulation. The reason for this is that the cost of an event in the simulator depends, in part, on the maximum outdegree of the "wake-up map", a graph derived from the rule set which keeps track of which rules are reactivated when a rule fires [24]. In the worst-case scenario, our generic rule expansion causes a "blow up" of the wake-up map with concomitant degradation in the simulator's performance.

More generally, our mechanism of using an agent hierarchy and generic rules to generate a concrete rule set allows the Kappa modeller a finer control of the granularity of his/her rules. Consider for example an agent A that can bind two agents, B1 or B2, and that binding with either is sufficient (and necessary) for A to bind a further agent C. To express this in Kappa, we would have to write two rules for A binding C; one for the case of B1, the other for B2. This isn't too bad—but if we have not two but a large number of activating ligands of A, it rapidly becomes tedious and error-prone to write the rule sets. By using a generic agent B, representing the class of A-activating ligands, we write just one generic rule that covers all cases (albeit requiring recompilation after the addition of new concrete descendants of B). Or to put it another way, we think of the generic agent B as generating a coarse-graining of the model's molecular species that no longer distinguishes between the various concrete descendants of B (i.e. B1, B2, etc).

With more complex agent hierarchies, one can express further, more subtle coarse-graining effects such as the MAP2K-MAPK binding specificity example above. However, it should be admitted that the example of MAPK is particularly conducive to a treatment of this kind (which is why we use it as our initial example!) and that not all signalling pathways exhibit the same degree of sharing of structure found here, as expressed by the highly generic nature of the rules. This in itself is a useful aspect of our language extension in that it enables us to recognize, formally, the fact that a pathway is highly generic or, on the contrary, particularly obtuse and dependent on many specific details. Indeed, the purpose of this extension is not to obtain a maximal "compression" of a concrete rule set into as few generic rules as possible; rather it is to illuminate the structure of a model by expressing it at an appropriate level of abstraction.

3 The perturbation space

Now that we have shown, with the MAPK example, how our parsimonious language extension enables rapid development of large rule sets via the mechanism of generic rules, let us turn to the main problem of interest here which is to build realistic models incorporating multiple erbB ligands and receptors, mutated forms of those receptors and monoclonal antibodies (mAbs) and tyrosine kinase inhibitors (TKIs) targeting those receptors. Unlike the previous MAPK model where the use of agent variants was convenient but hardly indispensable, in this case it would be a nightmarish process to write the rules directly in Kappa. As we will see, the use of agent variants not only helps to structure the model in a human-understandable manner, it also radically tames the combinatorial explosions caused by having multiple ligands and receptors and by the introduction of mutations.

We first define our agent hierarchy. It has two roots, erbB for the receptors and erbL for the ligands, each with four children.

```
%gen: erbB(L,CR,N,atp,AS,C,Y~u)
%gen: erbBL(L)
```

The next layer of agents splits the space of ligands into four, each with a different repertoire of receptors to which it binds.

```
%gen: erbBL1 = erbBL
%gen: erbBL14 = erbBL
%gen: erbBL34 = erbBL
%gen: erbBL4 = erbBL
```

Note that a hierarchical presentation of a model has a degree of intensionality and, in particular, is of course not unique—indeed, the compiled model is actually a presentation of itself. This begs the remark that a presentation is both a way to achieve compactness of description and to document knowledge about relationships between agents that *disappears* in the compilation process.

We also need variants for the four erbB receptors. Note that we introduce them as generic agents and only later specialize them as wild-types and mutant ones (only erbB1-WT is shown here).

```
%gen: erbB1 = erbB[Y\{Y1016, Y1092, Y1110, Y1172, Y1197}]
%gen: erbB2 = erbB[L\{}]
%gen: erbB3 = erbB[N\{}]
%gen: erbB4 = erbB
%conc: erbB1_WT = erbB1
```

To keep our presentation uncluttered, we have only shown the full repertoire of phosphorylation sites for erbB1; in the full model, the other receptors also have a similar complement of Y sites. Note though that erbB2 loses the site L and erbB3 the site N. Finally, let us note the concrete ligands. In what follows, we will in fact only consider EGF and HRG.

```
%conc: EGF = erbBL1
%conc: TGFalpha = erbBL1
%conc: AR = erbBL1

%conc: BTC = erbBL14
%conc: HB-EGF = erbBL14
%conc: ER = erbBL14

%conc: HRG = erbBL34
%conc: NRG2 = erbBL34

%conc: NRG3 = erbBL4
%conc: NRG4 = erbBL4
```

3.1 The consensus model

We build our consensus model on the basis of a conservative reading of the literature; see e.g.[26, 27]. Specifically, we consider that ligands bind monomer receptors which can then externally (on the trans-side of the plasma membrane) dimerize; this in turn enables the formation of asymmetric dimers that lead to receptor binding on the cis-side and cross-phosphorylation.

```
erbBL1(L), erbB1(L,CR) -> erbBL1(L!0), erbB1(L!0,CR)
erbBL14(L), erbB1(L,CR) -> erbBL14(L!0), erbB1(L!0,CR)
erbBL14(L), erbB4(L,CR) -> erbBL14(L!0), erbB4(L!0,CR)
erbBL34(L), erbB3(L,CR) -> erbBL34(L!0), erbB3(L!0,CR)
erbBL34(L), erbB4(L,CR) -> erbBL34(L!0), erbB4(L!0,CR)
erbBL4(L), erbB4(L,CR) -> erbBL4(L!0), erbB4(L!0,CR)
```

These six generic rules expand into a significant number of concrete Kappa rules in a manner that depends on the level of detail requested in the identities of ligands. For example, although there are three ligands of type erbBL1 and three of type erbBL14, the two ligands of type erbBL34 actually exist in multiple splice variants, as do those of type erbB4. In any case, at the very least these six generic rules give rise to fifteen concrete rules; and, of course, we also need the unbinding rule:

```
erbBL(L!0), erbB(L!0) -> erbBL(L), erbB(L)
```

Note that this generic unbinding rule will generate concrete rules that will never apply, e.g. a descendant of erbBL1 unbinding erbB3. However, these dead rules are detected by our static analysis and so can be removed (if desired) from the generated rule set.

```
erbBL(L!1), erbB(L!1,CR), erbBL(L!2), erbB(L!2,CR) -> \
 erbBL(L!1), erbB(L!1,CR!0), erbBL(L!2), erbB(L!2,CR!0)
erbBL(L!1), erbB(L!1,CR), erbB2(CR) -> \
 erbBL(L!1), erbB(L!1,CR!0), erbB2(CR!0)
```

The first of these generic rules deals with most of the cases of (external) dimerization. The only difficulty comes from the fact that erbB2 has no ligand binding site L and, as such, cannot ever match the generic erbB agent in that rule which mentions L. For this reason, we must explicitly include the second rule that covers the case of erbB2 dimerizing with a different erbB receptor type; we do not consider erbB2 homodimerization. These three rules generate a very large number (well over 150) of concrete rules due not only to the fact that the erbB generic agent is capable of multiple matches but also because the concrete erbB agents can bind multiple ligand agents.

We next have the rule for internal (or asymmetric) dimer formation. Here, a receptor binds its (external) dimer partner on a second site; this dimer is asymmetric because the bond is made between the N site of one receptor and the C site of the other. In a dimer not containing erbB3, this asymmetric dimer can flip states; this is the second rule.

```
erbB(CR!1,N,C), erbB(CR!1,C) -> erbB(CR!1,N!0,C), erbB(CR!1,C!0)
erbB(CR!1,N!2,C), erbB(CR!1,N,C!2) -> erbB(CR!1,N,C!3), erbB(CR!1,N!3,C)
```

The final rule is for trans phosphorylation of one erbB receptor by its dimer partner. In an asymmetric dimer, the receptor bound on site C is the *activator* whereas the receptor bound on N is the *activated*. It is thus the activator that gets phosphorylated.

```
erbB(N!1,atp,AS), erbB(C!1,Y~u) -> erbB(N!1,atp,AS), erbB(C!1,Y~p)
```

This one generic rule expands into many concrete rules for two independent reasons. Firstly, each erbB agent can be multiply instantiated: the first erbB can be anything but erbB3 (whose N site was deleted) and the second can be any of the four erbBs. Secondly, each receptor duplicates the site Y, so we get one concrete rule per duplicand. In the context of the purely wild-type model, neither the atp nor the AS site plays any role. This is because the rule tests only for the existence of these sites (which always succeeds) and that they are both unbound (which also always succeeds since we have no rules for binding to either of them). However, as we will see shortly, these two sites do play an important role once we take drug interventions and mutations into account.

We can neatly summarize the model so far with its contact map (Fig. 1).

3.2 Ligand perturbations

The erbB receptor network clearly has a lot of flexibility in its response to ligands. In particular, the receptor dimers that form depend on the available ligands and the presented receptors. In addition, erbB3 has compromised capability to form asymmetric dimers: it can activate the catalytic activity of its dimer partner but cannot be activated by it. This phenomenon adds yet another layer of subtlety to erbB receptor activation. For example, in a cell line expressing erbB2, erbB3 and erbB4, one would expect HRG to promote phosphorylation of erbB2, via erbB2:erbB4 dimers, as well as phosphorylation of erbB3 and erbB4. On the other hand, were erbB4 not expressed, one would expect only erbB3 phosphorylation, via erbB2:erbB3 dimers. However, this kind of reasoning rapidly becomes highly complicated, particularly in the presence of multiple ligands, and we would like some way of deducing, from the rule set and a choice of expression levels of ligands and receptors, which receptors get phosphorylated (and, in some cases, on which sites).

We can do this using static analysis of the rule set. We first write dummy rules that detect typical molecular species of interest, e.g.

```
erbB2(Y~p) -> erbB2(Y~p)
```

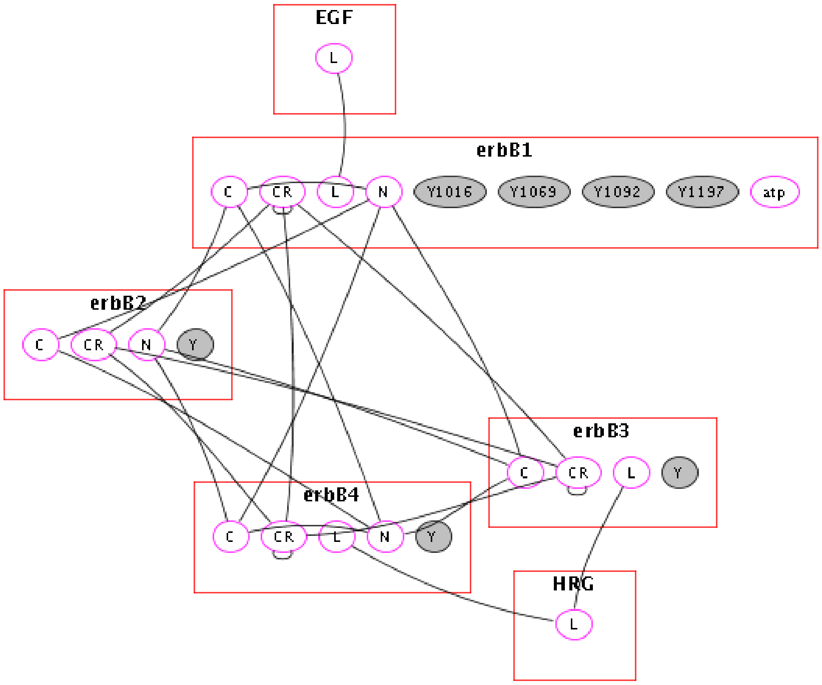

Fig. 1. *Contact map of the consensus model: each concrete agent is represented once with all its sites; possible bindings are indicated by an edge joining two nodes, modifiable sites are indicated in grey. Only a restricted subset of known EGFR receptors ligands is shown, namely EGF and HRG.*

We then ask the static analyser whether or not our dummy rules can fire. It responds in one of two ways: either a categorical 'no' or a tentative 'yes'. In the case of a 'no', we know (since the static analysis never produces false negatives) that our rule set cannot create the molecular species in question—starting from the declared initial solution. In the case of a 'yes', we have no certainly (since the analysis *can* give false positives) that the species can arise, but also no proof that it cannot. In an attempt to confirm the 'yes', we then use the story sampler to search for pathways leading to a dummy rule; if (at least) one exists, we have proof that the species can arise.

For example, the static analysis shows that, with our rule set, erbB2 phosphorylation cannot take place (categorical 'no') under the following conditions:

- HRG only; erbB2 and erbB3 only [erbB3 cannot phosphorylate erbB2]
- HRG only; erbB1, erbB2 and erbB3 only [erbB1 cannot bind HRG]

whereas it can potentially take place (tentative 'yes') under the following conditions:

- HRG only; erbB2, erbB3 and erbB4 only
- EGF and HRG; erbB1, erbB2 and erbB3 only.

To confirm this claim, we ask for stories leading to the appropriate observables. In both cases, we find indeed a story leading to phosphorylated erbB2 which confirms that the static analysis did not give us false positives (Figs. 2, 3).

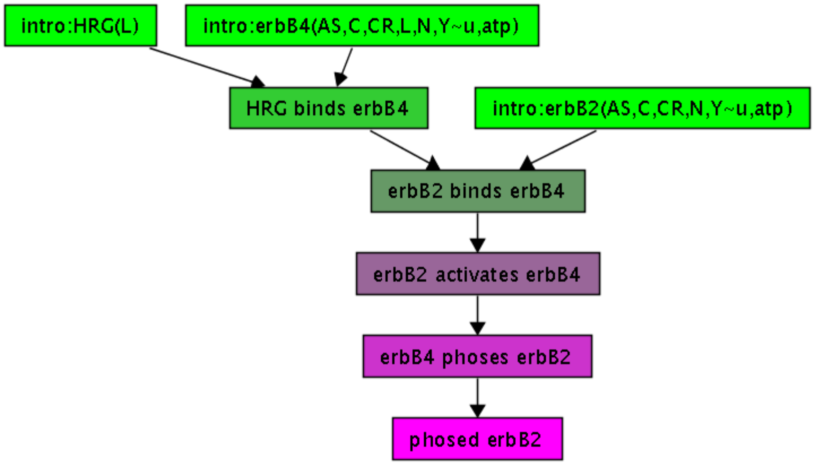

Fig. 2. *Story leading to erbB2 phosphorylation by erbB4.*

This combination of static analysis and story sampling enables a powerful model development process where, starting from a consensus, perhaps overly restrictive, rule set, we investigate which observables of interest can arise under which conditions. We then compare these predictions to experimental data in order to judge the accuracy and completeness of the model. If experimental data conflicts with the results of our analysis, this means one of two things: the consensus model either has fatal flaws or missing links. A 'fatal flaw' means that certain experimentally unobservable species can be generated by the rule set; in other words, that the mechanism described by the rules makes unwarranted assumptions. A 'missing link' corresponds to the more likely situation where an experimentally observed species remains inaccessible with our consensus model; this implies that the rule set lacks certain necessary rules.

For example, in the above discussion, we noted that, in our rule set with erbB1, erbB2 and erbB3 only, HRG stimulation leads only to erbB3 phosphorylation; whereas the combination of EGF and HRG leads to phosphorylation of all three receptors. This constitutes an experimentally refutable prediction. In the event of such a refutation, e.g. we observe erbB1 phosphorylation upon HRG stimulation, we could freely postulate various new rules, check that they do indeed open up the possibility of erbB1 phosphorylation and then compare and contrast their effects on other observables. If a new rule creates a 'fatal flaw', we can discount it; but in general this may still leave us with a choice between several proposed new mechanisms. To decide between these would require us to find a new, experimentally refutable prediction and do the experiment (or find it in the literature).

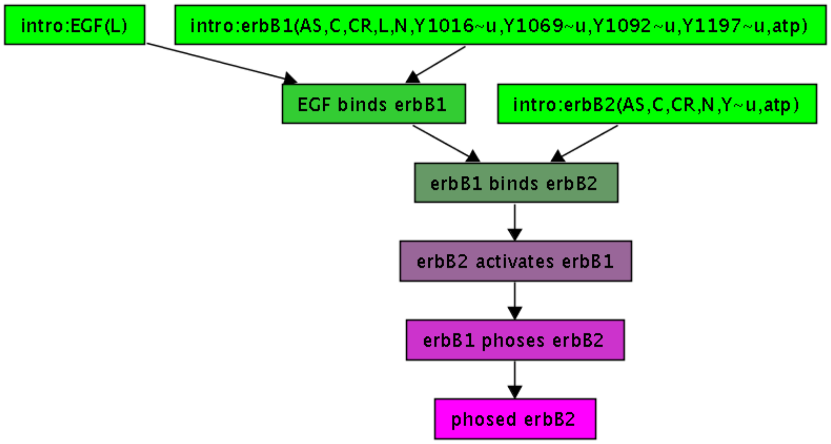

Fig. 3. *Story leading to erbB2 phosphorylation by erbB1.*

We stress that this remains a human-directed model development process—we do not consider automatically generated rules in any form—but one in which variant mechanisms can be built and evaluated in an organized fashion.

3.3 Drug perturbations

In recent years, particularly with the realization that deregulated erbB signalling contributes to the development of multiple cancers, much research has focussed on finding ways of blocking the activity of this family of receptors via drug intervention. To date, two broad classes of drug have been developed: monoclonal antibodies (mAbs) and tyrosine kinase inhibitors (TKIs). Antibodies typically act as classical competitive inhibitors that exert their function by binding cell surface receptors in a way that physically obstructs their usual ligands from binding. On the other hand, TKIs behave as classical non-competitive inhibitors of kinase activity that do not prevent substrate binding but instead block the ATP binding site of the kinase domain, thus preventing substrate phosphorylation.

As an illustration of the ease with which we can incorporate these kinds of pharmaceutical intervention in our modelling framework, we include rules for C225 (cetuximab, a mAb) binding to erbB1's site `L`, ZD1839 (gefitinib, a TKI) binding to erbB1's `atp` site and 4D5 (trastuzumab, another mAb) binding to erbB2's dimerization site `CR`.

```
C225(L), erbB1(L) -> C225(L!0), erbB1(L!0)
ZD1839(L), erbB1(atp) -> C225(L!0), erbB1(atp!0)
4D5(L), erbB2(CR) -> C225(L!0), erbB2(CR!0)
```

As each of these molecules has a *reversible* inhibitory effect on the respective receptor, we also need the accompanying unbinding rules:

```
C225(L!0), erbB1(L!0) -> C225(L), erbB1(L)
ZD1839(L!0), erbB1(atp!0) -> ZD1839(L), erbB1(atp)
4D5(L!0), erbB2(CR!0) -> 4D5(L), erbB2(CR)
```

The contact map of the system (Fig. 4) including these inhibitors makes it clear that the antibodies (C225 and 4D5) act as competitive inhibitors: C225 competes with EGF for the ligand binding site of erbB1 and 4D5 competes for the dimerization binding site of erbB2.

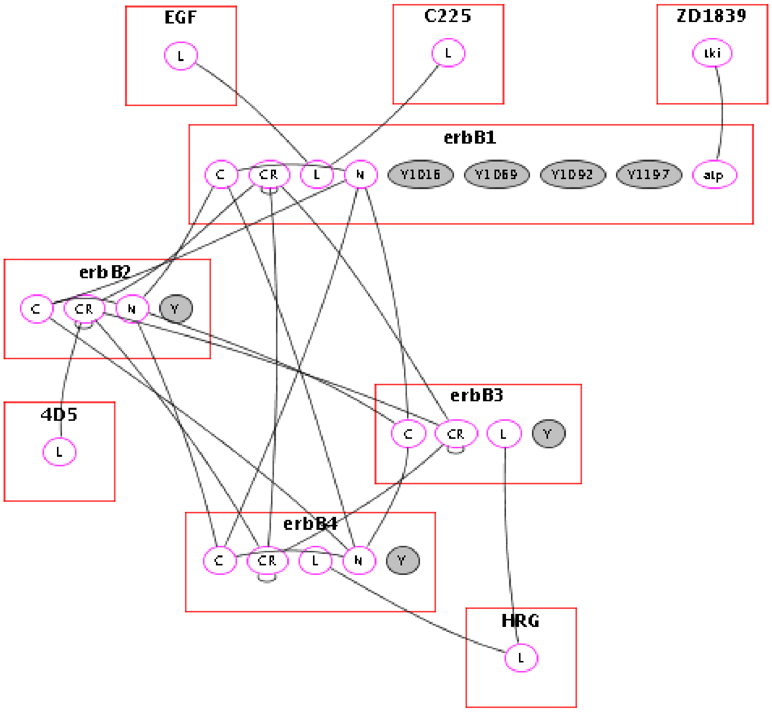

Fig. 4. *Contact map of the consensus model with antibodies (C225 and 4D5) as competitive inhibitors, and ZD1839 as non-competitive inhibitor.*

The inhibitory effect of ZD1839 shows up only in the influence map: the rule binding ZD1839 to erbB1 inhibits all modification rules (concretely, the phosphorylations) dependent on erbB1. This is because ZD1839 binds to the site atp of erbB1 which must be free in order for erbB1 to modify its dimer partner. The presence of ZD1839 thus frustrates, without completely preventing, erbB1-dependent phosphorylation. We will return to this later.

It should be noted that, in our examples of inhibitors, all agents are concrete. But, in general, drugs would also be organized, much like natural ligands, by an appropriate agent hierarchy.

3.4 The uses of mutational perturbations

So far, we have seen how the use of agent variants allows us to organize agents hierarchically and thus write generic rules at a convenient level of granularity. In particular, this facilitates the development of models with families of related proteins, or proteins with multiple splice variants, that have overlapping functionality. However, agent variants also enable the treatment of mutated agents which likewise share a lot of the functionality of their wild-type cousins but which also potentially lose some of that functionality and/or gain new functionality.

This has two immediate applications. Firstly, it allows us to build models with a mixture of wild-type and mutated agents in order to investigate (statically or numerically) the consequences of mutations. This is particularly interesting in the context of models, as described previously, that also include drug interventions.

More subtly, it also allows us to cast a critical eye over the assumptions we make in building our wild-type model. After all, a lot of the experimental data from which consensus pathways have been deduced comes from mutation experiments. These typically eliminate one or more phosphorylation sites in a protein and investigate which, if any, pathways suffer from this perturbation. However, such data can be difficult to interpret and the deduced wild-type interactions may be incorrect.

For example, as explained in [28], one experiment showed that expressing a kinase-dead mutant of PI3K inhibited Ras activation upon EGF stimulation; this led the authors to propose a role for PI3K in activating Ras. But then, a second study demonstrated that a constitutively active (and membrane associated) mutant of PI3K did *not* promote Ras activation, which contradicted the conclusions of the first study. In the end, it turned out [28] that PI3K actually *inhibits* Ras *de*activation; so PI3K sensitizes Ras for activation but cannot by itself actually activate it. In more details: PI3K promotes Gab1 recruitment to the membrane which, on EGF stimulation, strongly recruits Shp2 to the membrane; Shp2 is a tyrosine phosphatase that dephoses the phospho-tyrosine binding sites for RasGAP (and for PI3K!) on erbB receptors and Gab1. So Shp2 inhibits RasGAP recruitment to the membrane which indirectly aids Ras activation (Sos which activates Ras has an easier job). This gives a measure of the daunting complexity of inferring a protein network, and as a consequence a measure of how helpful a methodology such as the one we illustrate here can prove.

Indeed, using agent variants, we can express the kinds of (artificially) mutated proteins used in biochemical studies and so replay numerically such experiments. We can therefore detect, *in numero*, if the hypothesized wild-type network is in fact incorrect, e.g. if we had a model for Ras activation including a rule for ‘PI3K activates Ras’, we would have been able to predict that a constitutively active PI3K mutant would activate Ras; the fact that, experimentally, this is not observed means that that rule must be wrong. This kind of perturbational analysis is not just useful for postdictive verification of inferences, it is also a discipline to build a model upon such data, and to build further data to refute predictions; this could be particularly interesting if two plausible molecular mechanisms (candidate consensus pathways) made divergent predictions.

3.5 Testing the wild-type model

In two recent papers, Kuriyan and coworkers have developed a conceptual model of erbB receptor acivation that depends on the formation of an asymmetric dimer [27, **?**]. We have used this when writing the above rules for the wild-type erbB network in the previous section. They developed their model using a combination of structural and mutational data and provide convincing evidence of its correctness by cotransfecting various artifical erbB constructs that lack one or more of the N, C and AS sites.

We can use agent variants, in combination with static analysis, to reproduce these kinds of results *in numero*. For example, kinase-dead erbB1 is obtained by defining a variant of erbB1 with the AS site deleted; this agent can no longer phosphorylate its dimer partner. Similarly, we can also introduce variants that delete either the N or the C site instead of, or in addition to, the AS site.

```
%conc: erbB1_KD = erbB1[AS/{}]
%conc: erbB1_noN = erbB1[N/{}]
%conc: erbB1_noC = erbB1[C/{}]
%conc: erbB1_KDnoN = erbB1[AS/{} ; N/{}]
%conc: erbB1_KDnoC = erbB1[AS/{} ; C/{}]
```

These agents inherit all rules from wild-type erbB1 that do not mention the sites that they lack. So erbB1_KD can freely form asymmetric dimers but phosphorylates nothing, whereas erbB1_noN and erbB1_noC are partially compromised in their ability to form asymmetric dimers: the former can activate its partner and get phosphorylated, but cannot be activated and phosphorylate its partner; the latter can be activated by its partner and phosphorylate it, but cannot activate its partner and get phosphorylated.

We can now use static analysis, as in the previous section, to analyze the consequences of coexpressing pairs of these variant agents. We do this by checking the accessibility of the rules

```
erbB1(N!1), erbB1(C!1) -> erbB1(N!1), erbB1(C!1)
erbB1(Y1197~p) -> erbB1(Y1197~p)
```

(that respectively detect the possibility of an asymmetric dimer forming and an erbB1 receptor becoming phosphorylated) in an initial solution that includes EGF and a choice of any (one or) two of the erbB1 variants. In particular, we can recapitulate the results of [27] (see their Fig. 6; we use the same combination numbers) in completely automatic fashion:

1. wild-type erbB1 only: asymmetric homodimer accessible; phosphorylation accessible
2. erbB1_KD only: asymmetric homodimer accessible; phosphorylation inaccessible
3. erbB1_KD & erbB1_noN: one asymmetric heterodimer accessible; phosphorylation inaccessible
4. erbB1_KD & erbB1_noC: one asymmetric heterodimer accessible; phosphorylation accessible
5. erbB1_KDnoC only: asymmetric homodimer inaccessible; phosphorylation inaccessible
6. erbB1_KDnoC & erbB1_noN: one asymmetric heterodimer accessible; phosphorylation inaccessible
7. erbB1_KDnoC & erbB1_noC: asymmetric heterodimer inaccessible; phosphorylation inaccessible
8. erbB1_KDnoN only: asymmetric homodimer inaccessible; phosphorylation inaccessible
9. erbB1_KDnoN & erbB1_noN: asymmetric heterodimer inaccessible; phosphorylation inaccessible
10. erbB1_KDnoN & erbB1_noC: one asymmetric heterodimer accesible; phosphorylation accessible

In [27], this had to be done by hand, a task that soon begins to get rather subtle, particularly if you want to consider doubly-mutated agents and/or coexpression of more than two receptor constructs at a time. It is thus very useful to be able to express this situation in Kappa and rely on static analysis to detect the impossibility/possibility of phosphorylation. Moreover, if the static analysis announces that phosphorylation is not impossible, we can, as above, use the story sampler to search for ways in which this can actually take place. Again, in some cases, this is easy to do by hand but, beyond a certain degree of complexity, it is highly desirable to have an automatic method in order to avoid making mistakes.

These results demonstrate that our consensus model is indeed compatible with the experimental data of Kuriyan et al. and, as such, it passes the test. This comforts us, for now, in our choice of rules but of course provides no guarantee that future experimental data will not invalidate some of them.

3.6 The limits of perturbation testing

We mentioned earlier that our language extension shields the modeller from the underlying rule set generated by generic rules. However, we should say that this is only true *qualitatively*—if we wish to manipulate the rate constants of our model in such a way that different concrete instantiations of one generic rule get different kinetics, this can only be done by examining and modifying directly the generated rule set.

More generally, the modelling methodology advocated above based on static analysis cannot be used to gauge the effect of perturbations, such as drugs, that restrict, but don't outlaw, the application of other rules. Or, to put it another way, a perturbation that operates entirely at the level of kinetics is undetectable by this method. We would however expect to observe the effects of such perturbations during stochastic simulation and/or story sampling. Indeed, it would be straightforward to observe the inhibition of erbB1's kinase activity by tracking that rule's activity in the absence and presence of drugs. More ambitiously, we could compare the relative strengths of each erbB's kinase activity and the way in which that is disturbed by drugs that target only one receptor; this would require running the story sampler many times at many time points to get a statistical picture of the model's activity profile over time. We leave this for future work.

4 Conclusions

Even if wild-type pathways are obviously central to a systemic view of molecular biology, modelling is not just about these. It is equally important to be able to navigate the space of derivatives of a model for two complementary reasons. Firstly, one needs to understand diseased conditions as natural perturbations of the wild-type; secondly, one also needs to represent synthetic perturbations (by genetic knock-outs, domain truncations, point mutations, etc) because they are key in the inference of the wild-type. This is a formidable challenge because the space of such model perturbations introduces a second kind of combinatorial explosion. The well-studied example of the EGF receptor family (see §3) is a powerful illustration of this fact. Now, we have to do something if we want our modelling vessel to stay afloat in the sea of perturbations. In other words, just as the passage from reactions to rules tames the first binding-caused explosion, we have to find a mechanism to tame what one might call the perturbation-caused explosion.

The fact that Kappa describes molecular interactions at the level of domain binding and modification seems a good start, since this is the granularity at which the engineering of perturbations in protein networks actually happens (e.g. Y to A mutations that disable a modification). But to tackle our representation problem, we need another ingredient, namely a syntactic extension of Kappa that enables a clean, uniform treatment of protein families, splice variants and mutated proteins. This is what we have proposed here. The idea is to structure agents hierarchically so that rules can be expressed at an appropriate level of abstraction, as generic rules, which are then automatically compiled into pure Kappa. This eases the pain (and pitfalls) of writing large rule sets (indeed the modeller has no need to ever look at the resulting concrete rule set, unless he/she wishes to modify its rate constants), and as we wanted, this give means to navigate their perturbation space.

Of course there is no magic: to work around the explosive generativity of wild-type pathways we capture postulated regularities by using rules (if the universe of reactions were lacking any regularity no method could describe them anyway, a rather grim perspective for systems biology); to work around the second source of complexity, again we capture regularities of another kind, namely that much of the wild-type behaviour of a protein is actually shared with its mutants and isoforms.

We have shown that this strategy works well with our EGF example, as we were able to neatly set a wild-type model together with a selection of derivatives. With this model in place, one can bring the usual analysis tools of Kappa to bear on the rule set. As we have shown further, even in the absence of quantitative information about rates and copy numbers, one can obtain qualitative predictions about the induced perturbed behaviours and thus support on a full-scale the traditional informal inferences that are commonplace in the experimental investigation of protein networks.

References

1. B.N. Kholodenko, O.V. Demin, G. Moehren, and J.B. Hoek. Quantification of Short Term Signaling by the Epidermal Growth Factor Receptor. *J. Biol. Chem.*, 274(42):30169–30181, 1999.
2. A. Kiyatkin, E. Aksamitiene, N.I. Markevich, N.M. Borisov, J.B. Hoek, and B.N. Kholodenko. Scaffolding protein GAB1 sustains epidermal growth factor-induced mitogenic and survival signaling by multiple positive feedback loops. *J. Biol. Chem.*, 281:19925–19938, 2006.
3. Richard J. Orton, Oliver E. Sturm, Vladislav Vyshemirsky, Muffy Calder, David R. Gilbert, and Walter Kolch. Computational modelling of the receptor tyrosine kinase activated MAPK pathway. *Biochemical Journal*, 392(2):249–261, 2005.
4. Birgit Schoeberl, Claudia Eichler-Jonsson, Ernst-Dieter Gilles, and Gertraud Müller. Computational modeling of the dynamics of the map kinase cascade activated by surface and internalized EGF receptors. *Nature Biotechnology*, 20:370–375, 2002.
5. W.S. Hlavacek, J.R. Faeder, M.L. Blinov, R.G. Posner, M. Hucka, and W. Fontana. Rules for Modeling Signal-Transduction Systems. *Science's STKE*, 2006(344), 2006.
6. S. Maslov and I. Ispolatov. Propagation of large concentration changes in reversible protein-binding networks. *Proceedings of the National Academy of Sciences*, 104(34):13655, 2007.
7. A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 6, pages 459–470, Singapore, 2001. World Scientific Press.
8. A. Regev and E. Shapiro. Cells as computation. *Nature*, 419, September 2002.
9. Corrado Priami, Aviv Regev, Ehud Shapiro, and William Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 2001.
10. C. Baldi, Pierpaolo Degano, and Corrado Priami. Causal π-calculus for biochemical modeling. In *Proceedings of the AI*IA Workshop on BioInformatics 2002*, pages 69–72, 2002.
11. C. Priami and P. Quaglia. Beta Binders for Biological Interactions. In *Computational Methods in Systems Biology: International Conference CMSB 2004, Paris, France, May 26-28, 2004: Revised Selected Papers*. Springer, 2005.
12. L. Cardelli. Brane Calculi Interactions of Biological Membranes. In *Computational Methods in Systems Biology: International Conference CMSB 2004, Paris, France, May 26-28, 2004: Revised Selected Papers*. Springer, 2005.
13. A. Regev, E.M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. BioAmbients: an abstraction for biological compartments. *Theoretical Computer Science*, 325:141–167, 2004.
14. M. John, R. Ewald, and A.M. Uhrmacher. A Spatial Extension to the π Calculus. *Electronic Notes in Theoretical Computer Science*, 194(3):133–148, 2008.

15. M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. *Lecture Notes in Computer Science*, 4230:1–23, 2006.
16. F. Ciocchetta and J. Hillston. Bio-PEPA: an extension of the process algebra PEPA for biochemical networks. *Electronic Notes in Theoretical Computer Science*, 194(3):103–117, 2008.
17. L. Calzone, F. Fages, and S. Soliman. BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, 2006.
18. L. Dematte, C. Priami, and A. Romanel. The BlenX language: a tutorial. *Lecture Notes in Computer Science*, 5016, 2008.
19. M.L. Blinov, J.R. Faeder, and W.S. Hlavacek. BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20:3289–3292, 2004.
20. L. Dematté, C. Priami, A. Romanel, and O. Soyer. Evolving BlenX programs to simulate the evolution of biological networks. *Theoretical Computer Science*, 408(1):83–96, 2008.
21. V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.
22. V. Danos, J. Feret, W. Fontana, and J. Krivine. Abstract Interpretation of Cellular Signalling Networks. *Lecture Notes in Computer Science*, 4905:83–97, 2008.
23. V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-Based Modelling of Cellular Signalling. *Lecture Notes in Computer Science*, 4703:17–41, 2007.
24. V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable Simulation of Cellular Signaling Networks. *Lecture Notes in Computer Science*, 4807:139–157, 2007.
25. LO Murphy, S. Smith, RH Chen, DC Fingar, and J. Blenis. Molecular interpretation of ERK signal duration by immediate early gene products. *Nat Cell Biol*, 4(8):556–64, 2002.
26. A.W. Burgess, H.S. Cho, C. Eigenbrot, K.M. Ferguson, T.P.J. Garrett, D.J. Leahy, M.A. Lemmon, M.X. Sliwkowski, C.W. Ward, and S. Yokoyama. An Open-and-Shut Case? Recent Insights into the Activation of EGF/ErbB Receptors. *Molecular Cell*, 12(3):541–552, 2003.
27. X. Zhang, J. Gureasko, K. Shen, P.A. Cole, and J. Kuriyan. An Allosteric Mechanism for Activation of the Kinase Domain of Epidermal Growth Factor Receptor. *Cell*, 125(6):1137–1149, 2006.
28. C. Sampaio, M. Dance, A. Montagner, T. Edouard, N. Malet, B. Perret, A. Yart, JP Salles, and P. Raynal. Signal strength dictates phosphoinositide 3-kinase contribution to Ras/extracellular signal-regulated kinase 1 and 2 activation via differential Gab1/Shp2 recruitment: consequences for resistance to epidermal growth factor receptor inhibition. *Mol Cell Biol*, 28(2):587–600, 2008.
29. X. Zhang, KA Pickin, R. Bose, N. Jura, PA Cole, and J. Kuriyan. Inhibition of the EGF receptor by binding of MIG6 to an activating kinase domain interface. *Nature*, 450(7170):741, 2007.