

Rule-based modelling and tunable resolution

Russ Harmer

▶ **To cite this version:**

Russ Harmer. Rule-based modelling and tunable resolution. EPTCS, 2009, 9, pp.65-72. hal-00694488

HAL Id: hal-00694488
https://hal.science/hal-00694488

Submitted on 4 May 2012

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

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

Rule-based modelling and tunable resolution

Russ Harmer
PPS, CNRS & Université Paris Diderot–Paris 7

We investigate the use of an extension of rule-based modelling for cellular signalling to create a structured space of model variants. This enables the incremental development of rule sets that start from simple mechanisms and which, by a gradual increase in agent and rule resolution, evolve into more detailed descriptions.

1 Rule-based modelling

Cell signalling networks involve many binding/unbinding interactions combined with post-translational modifications (PTMs) between a large number of primarily proteins. As is by now well-known [1], this gives rise to a combinatorial explosion in the number of possible molecular species and so in the number of reactions required to define a model. For example, a protein with ten phosphorylation sites already has, in isolation, 2^{10} possible states which would mean 1024 distinct molecular species. A protein-protein interaction (PPI) depending on just one of those sites should therefore be independent of the states of the other nine; this would require 2^9 reactions, one for each possible state of those nine sites.

This problem can be solved by using *rules* that act upon *partially specified* molecular species; so a rule determines an entire family of reactions. In our example, this replaces the 2^9 reactions with a *single* rule that mentions only the site of interest (and the other nine sites not at all). Rule-based modelling makes explicit the notion of *agent*—and, as such, is sometimes called agent-based modelling—that is left implicit in traditional reaction-based models. In that setting, an "agent" is implicitly a combination of molecular species just as, conversely, a rule is a combination of reactions. A further refinement in this setting is that an agent has an *interface*, a set of *sites* through which it can bind other agents and which can have internal *states* that represent, typically, the presence or otherwise of PTMs such as phosphorylation that may affect a protein's behaviour.

In this paper, we use the rule-based language Kappa [2] as our starting point. In Kappa, there are three principal kinds of rule: bindings, unbindings and modifications. We use binding rules to create complexes by explicitly connecting agents together via their sites; a given site on an agent can be bound to at most one other site at a time. In the concrete syntax of Kappa, the two sites involved in a bond are identified by a shared (but otherwise unimportant) number prefixed by `!`:

```
C(r),C(l) -> C(r!0),C(l!0)
```

So `C(r!0),C(l!0)` is nothing more than a textual syntax for the site graph [3] consisting of two nodes—both labelled `C`, one showing a site `r`, the other a site `l`—connected by a bond between those sites. In the natural way, unbinding rules undo previous bindings; they typically split a complex into two disjoint pieces (but not if the complex were held together by a double bond, for example). In the concrete syntax of Kappa, an internal state of a site is a text string prefixed by `~`. Modification rules simply change such internal states. For example, the following rule says that the site `s` of agent `S` is modified from state `u` to state `p`—but only when this site `s` is bound to the site `k` of agent `E`.

```
E(k!1), S(s~u!1) -> E(k!1), S(s~p!1)
```

S. Barry Cooper & Vincent Danos (eds.): Fifth Workshop on
Developments in Computational Models—Computational Models From Nature
EPTCS 9, 2009, pp. 65–72, doi:10.4204/EPTCS.9.8

1.1 Agent variants

Signalling proteins are generally organized by biologists into family groups (kinases, phosphatases, small G-proteins, *etc.*) then smaller, more precise subgroups (tyrosine kinases, MAPK phosphatases, Ras, *etc.*), ending with the individual proteins (erbB2, MKP-3, H-Ras, *etc.*). Closely related proteins often share many of their interactions and this causes a second combinatorial explosion due to the need to enumerate many *instances* of essentially the same rule, *e.g.* if four related agents `A1-4` can each bind any of three further agents `B1-3`, we have to write each instance of this binding explicitly (so 12 rules in all).

To tackle this problem, we have recently [4] proposed an extension of rule-based modelling where, in addition to rules, the modeller also defines an *agent hierarchy*. This specifies how new agents are defined from existing ones by small modifications of their sites: a site of the parent can be deleted, renamed or duplicated in the child. The agent hierarchy is rooted by agents declared *ab initio*. The descendants of an agent are called its *variants*; by default, leaves are *concrete* agents and the others are *generic* agents.

Rules can mention agents and sites at any level of this hierarchy; a rule is *generic* unless it involves only concrete agents and sites. A generic rule generally has many concrete *instantations*: a concrete agent `A` inherits a rule *r* mentioning one of its ancestors `B` provided that no site of `B` mentioned in *r* has been deleted in `A`; if one of `B`'s sites has been duplicated in `A`, the rule *r* is duplicated with one instance for each duplicate. An agent hierarchy plus a set of rules thus determines, via this *compilation* procedure, a typically much larger set of concrete rules. For example, we could define a new agent `T` by duplicating the site `r` of the agent `C` above:

```
T = C[r\{r1 r2}]
```

The above binding rule for `C` becomes two binding rules for `T`: one for the sites `l` and `r1`, the other for `l` and `r2`. As a consequence, whereas `C` builds up linear polymers, `T` builds (binary) branching polymers.

This framework allows a clear uniform treatment of: agents representing products of closely related genes, *e.g.* ERK1 and ERK2; agents representing splice variants of a single gene, *e.g.* p66, p52 and p46 Shc; and agents representing mutated forms of proteins, *e.g.* the L858R substitution of EGFR. An agent hierarchy can be thought of as an abstract, *i.e.* not necessarily factual, phylogenetic tree; as such, in [4], agent hierarchies were restricted to be trees. In this paper, we weaken this to DAGs; hierarchies need not therefore correspond to authentic phylogenetic trees, but we will see that this affords pragmatically useful additional expressive power.

1.2 Agent resolution

There is a subtle tension in representing biological knowledge. On the one hand, a number of online databases (*e.g.* UniProt, HPRD) document, in great detail, the sequences, domains, modifiable residues and interactions of proteins but do not provide any formalization of this knowledge. On the other hand, formal models rarely document proteins but instead provide an executable representation of their interactions, *e.g.* as ODEs. Indeed, it might not even be desirable to represent all information in a model, *e.g.* it might suffice to consider all products of the same gene (or even several related genes) as a single entity in order to illustrate a desired point.

The rule-based approach takes a first step towards resolving this issue. A rule is a formalization of a little nugget of biological knowledge, *e.g.* 'the SH2 domain of Grb2 binds the phosphorylated Tyr317 residue of p52 Shc'. As such, a rule-based model is essentially a *self-documenting collection of facts* that is, moreover, directly executable [5]. This contrasts with most reaction-based models which, while executable, disperse facts across multiple reactions as a consequence of the need to fully specify all molecular species, *cf.* the dispersion of agents across molecular species.

The introduction of an agent hierarchy significantly extends the documentary power of the rule-based approach. First of all, the very definition of a hierarchy already collates a significant amount of biological knowledge and can serve as useful documentation for families of proteins. More importantly, an agent hierarchy is written down once and for all and yet, in conjunction with a single generic rule set, can be used to generate multiple concrete rule sets just *by varying the choice of concrete agents*. (Recall that, by default, these are the leaves of the hierarchy.)

For example, the above rule codifying Grb2's binding to p52 Shc actually applies to all three splice variants of Shc and to two distinct residues of each Shc variant, although the precise numbering of the residues is different in each. We can easily express this with a simple hierarchy of Shc agents and a single generic rule:

```
Shc(PTB,YXNX~u,SH2)
Grb2(SH3n,SH2,SH3c)

Shc(YXNX~p), Grb2(SH2) -> Shc(YXNX~p!0), Grb2(SH2!0)

p66 = Shc[YXNX\{Y349 Y427}]
p52 = Shc[YXNX\{Y239 Y317}]
p46 = Shc[YXNX\{Y194 Y272}]
```

If `p66`, `p52` and `p46` are taken to be the concrete agents (as they are by default), the generic rule compiles into six concrete rules, two for each variant of Shc. However, if we specify `Shc` to be concrete, the "generic" rule is actually concrete (it is its own compilation).

This means that a modeller can transparently shift between different levels of *agent resolution*: if certain details are unimportant, they can be filtered out by designating the concrete agents to be higher up in the hierarchy; if it turns out later that those details actually are important, the preceding design choice can be reversed simply by moving the concrete fringe back down and recompiling the generic rules. Usually, such design choices are hard-wired into the details of a model and cannot just be undone without forcing wholesale changes to the model. The compilation of a generic rule set with respect to a particular choice of concrete agents therefore gives modellers the ability to *tune* the level of agent resolution for a given model; indeed, a generic rule set is not one model but many—parametrized by the placement of the concrete agents in the hierarchy.

1.3 Rule resolution

We now turn to the related notion of *rule resolution* by which we mean the extent to which (generic) rules are restricted to only certain subfamilies of agents. For example, a highly generic rule for the docking between MAP2Ks and MAPKs would be perfectly sufficient to study typical properties of an isolated MAPK cascade:

```
MAP2K(D,AS,atp,S~u,ST~u,cat~n)
MAPK(CD,AS,atp,T~u,Y~u,cat~n)

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

(Note that this docking rule is reversible and so is actually two rules: docking and undocking.)

If we then wished to turn our attention to the behaviour of particular human MAPK cascades, we might extend the hierarchy along the following lines.

```
ERKK = MAP2K
JNKK = MAP2K
p38K = MAP2K
ERK = MAPK
JNK = MAPK
p38 = MAPK

ERKK(D), ERK(CD) <-> ERKK(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)
```

Note that the cascade has become three parallel cascades. The original docking rule would now generate far too many concrete rules—most of which would violate the desired binding specificity of `ERKK` for `ERK`, `JNKK` for `JNK` and `p38K` for `p38`—so we have replaced it with more specific *instantiations* that "insulate" the cascades from each other. This notion of (generic) rule instantiation is perfectly well-defined, even if we replace a generic agent by another generic agent below it in the hierarchy. For example, if we further extended our hierarchy with

```
MEK1 = ERKK
MEK2 = ERKK
```

the instantiation of `MAP2K` by `ERKK` and `MAPK` by `ERK` would yield the now-generic rule

```
ERKK(D), ERK(CD) <-> ERKK(D!0), ERK(CD!0).
```

Rule instantiation is thus a sort of *partial* compilation of generic rules into concrete rules. It is also analogous to rule *refinement* [3]; indeed, in many agent hierarchies, all variants of some root agent can be seen, in pure rule-based style, as instances of the same agent in different states (*e.g.* via "private" sites with internal states encoding the agent's "real" name in the agent hierarchy). The process of rule instantiation then corresponds to refinement of that rule to accept only certain states (*i.e.* certain descendants) of the agent [V. Danos, private communication]. Of course, the special case of instantiating rules with only concrete agents corresponds exactly to the rule compilation described in [4] (and outlined in §1.1).

It is important to note that, in the previous section, the agent hierarchy was varied (in the choice of concrete agents) while the generic rules remained fixed. This permits the modeller to produce a range of rule sets that vary in their level of agent detail. In this section, however, it is the rules that are varied (by instantiation) while the hierarchy remains unchanged. Instead of tuning the agent resolution of the induced concrete rule set, this tunes the *rule resolution* and leads to a more carefully generated (and typically much smaller) concrete rule set. We thus have two independent axes of tunable resolution. In the next section, we examine real examples of this coming from the interactions between MAPKs and their phosphatases and upstream activators respectively.

Context This work belongs to a large and growing literature based on the approach, initiated by Regev [6], of using process-theoretic ideas (from concurrency theory) to represent proteins and their networks of interactions. While very promising, this approach is not always straightforward to use as models rapidly grow in size and complexity, becoming unwieldy and hard to maintain and further develop; furthermore, biological knowledge is increasing all the time, so a realistic modelling framework must be sufficiently intuitive, yet flexible enough to accommodate regular updates; cf. [7]. The ability to tune agent and/or rule resolution addresses this by allowing for incremental development of models: instead of having to work out and express everything in one go, a first approximation can be made which will be subsequently refined by extending agent hierarchies and/or instantiating generic rules to get more precise descriptions.

2 Tunable resolution

2.1 MAPK–MAPK phosphatase interaction

MAPK phosphatases come in three basic flavours: S/T- (serine/threonine) specific, Y- (tyrosine) specific and dually specific phosphatases. We can organize this into a simple V-shaped (DAG) hierarchy:

```
MAPKSTP(AS,cat~n)
MAPKYP(AS,cat~n)

MAPKSTP(AS,cat~y), MAPK(T~p,Y) -> MAPKSTP(AS!0,cat~y), MAPK(T~p!0,Y) @ b (u)
MAPKSTP(AS!0), MAPK(T!0) -> MAPKSTP(AS), MAPK(T)
MAPKSTP(AS!0,cat~y), MAPK(T~p!0) -> MAPKSTP(AS!0,cat~y), MAPK(T~u!0)

MAPKYP(AS,cat~y), MAPK(T,Y~p) -> MAPKYP(AS!0,cat~y), MAPK(T,Y~p!0) @ b (u)
MAPKYP(AS!0), MAPK(Y!0) -> MAPKYP(AS), MAPK(Y)
MAPKYP(AS!0,cat~y), MAPK(Y~p!0) -> MAPKYP(AS!0,cat~y), MAPK(T~u!0)

MKP = MAPKSTP[+KIM]
MKP = MAPKYP[+KIM]

MKP(KIM), MAPK(CD) <-> MKP(KIM!0), MAPK(CD!0)
```

Note that `MKP` has two distinct definitions; it is an *alias* for its two parents. This means it inherits from both and so acquires the necessary rules to modify `MAPK` on both `T` and `Y`. It has also acquired a new site `KIM` which is used to dock with `MAPK`s. The rules are textbook examples of enzyme-substrate interaction; the only subtlety lies in their kinetic rates (`b` and `u`). In rule-based modelling, a binary rule, while usually causing a binary reaction (with rate `b`), may also provoke a *unary* reaction: the two agents binding may *already be bound* on other sites in which case `u` (units s^{-1}) is the corresponding first-order rate constant.

2.2 Increasing resolution

Let us now extend the hierarchy to add some additional agents and generic rules:

```
HePTP = MAPKYP
ERKP = MPK
JNKP = MPK
p38P = MPK

ERKP(KIM), ERK(CD) <-> ERKP(KIM!0), ERK(CD!0)
JNKP(KIM), JNK(CD) <-> JNKP(KIM!0), p38(CD!0)
p38P(KIM), JNK(CD) <-> p38P(KIM!0), p38(CD!0)

MKP3 = ERKP
MKP1 = ERKP
MKP1 = JNKP
MKP1 = p38P
DUSP5 = JNKP[cat~y]
DUSP5 = p38P[cat~y]
```

```
MKP3(KIM!1,cat~n), ERK(CD!1) -> MKP3(KIM!1,cat~y), ERK(CD!1)
MKP1(KIM!1,cat~n), MAPK(CD!1) -> MKP1(KIM!1,cat~y), MAPK(CD!1)
```

Note that these three new docking rules are instantiations of the generic MKP/MAPK docking; note also that HePTP does not acquire a docking rule.

The DAG structure naturally *encodes* the observed facts [8] that MKPs have complex, overlapping binding specificities—but does not *explain* how. Some MPKs specifically bind ERK, others specifically bind JNK/p38 and others still bind all three kinds of MAPKs. In reality, this variety of binding specificity is brought about by there being (at least) two distinct possible binding sites on the MAPK and two or three on the MKP; a more detailed hierarchy could therefore be defined that expresses this additional detail. This would constitute a better formal expression of the actual molecular mechanism used by MKPs and MAPKs to dock, not just the functional consequences of that mechanism that the DAG encodes; but we must leave the construction of that more detailed model, and comparison with the present model, for future work.

The agent DUSP5 adopts a new default internal state y for its site cat whereas MKP3 and MKP1 acquire this state only after docking, presumably via an allosteric mechanism [8]. This reflects the fact that the enzyme-substrate interaction between either MKP1 or MKP3 and MAPKs only occurs as a monomolecular reaction (within an MKP−MAPK complex); whereas DUSP5 can additionally engage in docking-independent enzyme-substrate interaction.

It would obviously be possible, although cumbersome, to transform this hierarchy into a tree by the usual algorithm that expands any DAG to a tree; but it would be much more interesting to attempt to build a tree hierarchy that accurate reflects the actual phylogeny of the MKP protein family. It is to be hoped that the corresponding rule set would be fairly generic since one would expect close phylogenetic relatives to share many interactions; but this, too, we must leave for future investigation.

In summary, we have navigated from a starting, "simple" rule set that is generated by taking MKP to be a concrete agent and using the completely generic docking rule, to a "detailed" rule set generated by taking MKP as a generic agent with several distinct concrete children and the three more specific instantiations as docking rules. Note that the agent hierarchy itself *does not change*; only our choice of concrete agents with respect to which we compile. Analogously, the rule set *does not change* either; it is our choice of rule instantiations that varies. In other words, these two very different rule sets actually coexist within a single model; and we can choose between them simply by picking the appropriate agent and rule resolutions and compiling down to concrete rules.

There is a sense in which we can consider the detailed rule set as the result of an *evolution* of the simple rule set. A generic rule can either *adapt* to the introduction of new agent variants, *e.g.* the docking rule that is replaced by its three instantiations, or be *conserved* across the new agents, *e.g.* the enzyme-substrate rules for MKPs that do not take any account of the refinement of MKP or MAPK into subfamilies. With hindsight, the erbB rule set developed in [4] can also be seen in this way as an evolution from a single ligand-receptor interaction to a rich network of ligand families and receptors with complex, overlapping binding specificities.

2.3 Distributive vs. processive activation

As a final illustration, we outline how the evolution of rule sets can even modify the ways in which rules can be applied so as to change the molecular mechanism they represent. We begin with the MAP2K and MAPK agents from §1.3 and add the standard enzyme-substrate rules precisely as for MKP.

```
MAP2K(AS,cat~y), MAPK(T~u,Y) -> MAP2K(AS!0,cat~y), MAPK(T~u!0,Y) @ b (u)
MAP2K(AS!0), MAPK(T!0) -> MAP2K(AS), MAPK(T)
MAP2K(atp,AS!0,cat~y), MAPK(T~u!0) -> MAP2K(atp,AS!0,cat~y), MAPK(T~p!0)

MAP2K(AS,cat~y), MAPK(T,Y~u) -> MAP2K(AS!0,cat~y), MAPK(T,Y~u!0) @ b (u)
MAP2K(AS!0), MAPK(Y!0) -> MAP2K(AS), MAPK(Y)
MAP2K(atp,AS!0,cat~y), MAPK(Y~u!0) -> MAP2K(atp,AS!0,cat~y), MAPK(Y~p!0)
```

In isolation, these rules implement what biochemists call a *distributive* dual phosphorylation mechanism: the enzyme must unbind its substrate after each modification, so two independent enzyme-substrate interactions are needed for a doubly-phosphorylated substrate to come about. This mechanism is often found in real MAPK cascades, particularly in (relatively speaking) simple organisms such as xenopus oocytes [9], and gives rise to a sharply ultrasensitive dose-response curve.

If we now reincorporate the simple docking rule from §1.3,

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

the above enzyme-substrate interaction rules can be applied to either docked or undocked agents. In particular, it is possible for a `MAP2K` to dock with a `MAPK` and modify *both* its sites before undocking. This implements what biochemists call a *processive* dual phosphorylation. Such a mechanism typically produces a less sharp does-response curve as, unlike in the distributive case, doubly-phosphorylated substrate can start to appear immediately.

Note that, as in the `MKP` example, we have left the enzyme-substrate rules unchanged, *i.e.* we consider them to be conserved—and hence promiscuous—across all variants of `MAP2K` and `MAPK`. Alternatively, we could have imposed binding specificity at the level of these rules by instantiating them with the more specific agents (`ERKK`, `JNKK` and `p38K` from §1.3). However, it seems more plausible from an evolutionary point of view that fine-tuned binding specificity arises from the appearance of docking sites not directly implicated in the enzyme mechanism—as "unlucky" mutations in this mechanism might risk destroying the protein's catalytic activity. In any case, the choices we make of when to instantiate (*i.e.* adapt) and when not to (*i.e.* remain conserved) determine a rudimentary formal description (realistic or not) of the evolution of a rule set.

3 Discussion

We have presented a framework for rule-based modelling where multiple (concrete) models can be extracted from a single generic rule set and associated agent hierarchy. As such, it allows modellers to navigate around a space of related (concrete) models, selecting appropriate levels of detail according to the question in hand.

In particular, this allows for the incremental development of models where one starts from a rather simple-minded mechanism and gradually refines this into a detailed model by increasing agent and/or rule resolution, *e.g.* the MAPK model described in this paper (MAP2K, MAPK and MKP). As a bonus, the use of generic rules also greatly speeds up the process of model construction, not least because as one is no longer drowning in a sea of concrete rules.

More conceptually, we view this process of "unveiling" a detailed rule set from a modest one—by gradual increasing agent and rule resolution—as the first steps towards a formalization of what it means to "evolve" a rule set. For example, the MAP2K/MAPK rule set starts as a rudimentary enzyme-substrate interaction in a single cascade activated by a distributive mechanism, which evolves into multiple cascades activated by a docking-mediated processive mechanism.

The next step, as briefly mentioned above, would be to produce a more phylogenetically-accurate agent hierarchy, accompanied by a more precise generic rule set, in an attempt to explain how, for example, MKP/MAPK docking specificity is implemented at the molecular level. In order to do this, we might need to move to a deeper level of representation of proteins: rather than representing a protein as a single agent, instead consider individual domains as agents and proteins as unbreakable complexes, connected by a backbone. This representation would enable separate hierarchies for domains, not just proteins, which would afford far more subtle possibilities for protein construction by recombination of elements of these hierarchies.

Acknowledgements My thanks to Vincent Danos, Jérôme Feret, Walter Fontana and Jean Krivine for numerous discussions related to this work.

References

[1] 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.

[2] 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.

[3] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling, symmetries, refinements. *Lecture Notes in Computer Science*, 5054:103–122, 2008.

[4] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling and model perturbation. Submitted to Transactions on Computational Systems Biology, 2009.

[5] V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable Simulation of Cellular Signaling Networks. *Lecture Notes in Computer Science*, 4807:139–157, 2007.

[6] A. Regev and E. Shapiro. Cells as computation. *Nature*, 419, September 2002.

[7] M.L. Guerriero, J.K. Heath, and C. Priami. An Automated Translation from a Narrative Language for Biological Modelling into Process Algebra. *Lecture Notes in Computer Science*, 4695:136–151, 2007.

[8] D.M. Owens and S.M. Keyse. Differential regulation of MAP kinase signalling by dual-specificity protein phosphatases. *Oncogene*, 26(22):3203–3213, 2007.

[9] J.E. Ferrell and R.R. Bhatt. Mechanistic studies of the dual phosphorylation of mitogen-activated protein kinase. *Journal of Biological Chemistry*, 272(30):19008–19016, 1997.