xxx
xxx
Advance Access Publication Date: Day Month Year
Original paper

Systems biology

rxncon 2.0: a language for executable molecular systems biology

J.C. Romers [1,*] and M. Krantz [1,*]

[1] Humboldt Universität zu Berlin, Theoretical Biophysics, 10115 Berlin, Germany.

* To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Advances in the experimental state of the art have provided us and continue to provide us with more and more detailed information regarding the mechanisms underlying signal transduction phenomena in living cells. Simultaneously, progress in modelling techniques and computer hardware enable simulations of ever larger models. We present *rxncon*, a modelling language composed of precise and concise statements describing the experimental knowledge of signalling network.
Results: We provide the syntax and semantics of *rxncon*, the reaction-contingency language.
Availability: http://github.com/rxncon
Contact: jesper.romers@hu-berlin.de
Contact: marcus.krantz@biologie.hu-berlin.de
Supplementary information:n/a online.

1 Introduction

The cellular regulatory networks monitor the state of a cell and its surroundings, and control key cellular processes such as metabolism, cell division and apoptosis. One of the main challenges of systems biology is to provide a mechanistic (as opposed to phenomenological) understanding of these networks in terms of their elementary building blocks: the reactions between and states of biological molecules.

Mechanistic understanding requires collection and integration of knowledge. This in turn requires a language in which these tasks are natural. Such a language, *rxncon*, is presented here. Its statements consist of the *appropriate abstractions* to naturally describe the mechanistic building blocks responsible for cellular signalling: *elemental reactions*, that describe independent reaction events, and their *contingencies*, the necessary molecular context consisting of interactions and modifications. The elemental reactions create and destroy states, which in their turn make up the contingencies. In this sense, the language is very close to *experiment*, since every statement corresponds to an experimental fact.

Scalability is a fundamental problem in the description of cellular networks (Hlavacek *et al.*, 2003), in particular when aiming for genome-scale models (GSM). The actual problem is twofold, first in the model formulation, and second in the model execution: even when the formulation of a model does not run into scalability issues, the execution or simulation might still be infeasible. The state-of-the-art genome-scale models are metabolic models (Thiele and Palsson, 2010). In fact, all genome-scale models to date are metabolic GSMs, and the success of the field relies on the characteristics of the metabolic networks. The metabolic networks transfer mass via metabolic conversion and transport reactions. Metabolites in different compartments are considered different components and hence they are not considered to carry any further internal structure in the form of states. Instead, metabolic reactions create and destroy metabolites, making the reaction mutually exclusive at the level of individual metabolites. The absence of combinatorial interplay between reactions is one of the key features of metabolic networks, as it pre-empts the combinatorial complexity that constitutes the principal challenge in both modelling and empirical analysis of signal transduction networks.

Cellular regulatory networks require a different approach than metabolic networks. The regulatory networks process information, which is encoded primarily in state changes of components, with no or limited or local transfer of mass only. Consequently, the assumptions underlying the metabolic modelling approaches are not valid or useful in describing cellular regulation. The critical difference is that most components can undergo multiple distinct reactions, each of which changes a site-specific state (e.g. modification at specific residue, ligand binding at specific domain), and that most of these state changes can be combined.

The problems posed by combinatorial complexity are well-known (Hlavacek and Faeder, 2009; Rother *et al.*, 2013) and overcoming them is one of the challenges in the field. The approach followed by *rxncon* is one of macrostates: instead of enumerating all possible combinations of states generated by the reaction events (the so-called microstates), we "trace out" the degrees of freedom that are unknown to us anyway. Rule-based modelling, which adheres to the "don't care, don't write" principle (Faeder *et al.*, 2005, 2009; Danos and Laneve, 2004), follows a similar approach. Indeed, in coming work we present the translation of *rxncon* systems to rule-based models (in preparation).

Furthermore *rxncon* is *composable*. This means that single reaction events can be added to a system by adding single statements and

without touching all previous statements. Similarly, as knowledge about a signalling network progresses, more accurate domain or residue information can be provided without having to completely start from scratch. This facilitates *iterative* model building and *cooperative* efforts by multiple groups.

Finally, the language is *compilable* to executable models. In this work we establish the formal semantics of the language, but in other works we discuss the translation into a bipartite Boolean model (Thieme *et al.*, 2017) and into a rule-based model (in preparation).

Taken together, we present a scalable, composable language for describing cellular signal transduction processes that contains the appropriate abstractions to make a direct connection with experimental knowledge and which is compilable to executable, simulatable models.

2 Syntax and semantics of the *rxncon* language

A *rxncon* system can be thought of as a compendium of knowledge about the mechanistic processes that underlie cellular signal transduction phenomena. The language, which we formalize below, consists of a collection of statements enumerating the biochemical reactions and their contingencies, the context in which these reactions take place.

Since each statement considers either a *reaction* or a *contingency*, each individual statement is an experimentally verifiable fact about the signalling network, that can be annotated with literature sources and further details. These statements are independent: the reactions only denote which property of a molecule (phosphorylation residue, binding domain) changes, without having to resort to a microstate description, which is inherently unscalable.

In these sections, we give a formal definition of the *rxncon* language. We use Backus-Naur Form (BNF) (Backus, 1959; Naur, 1961) definitions to describe syntactically correct *rxncon* statements, and show how the BNF products map to different semantic concepts. These concepts in turn map to classes in the code of our implementation of the language. We will sometimes refer to a property p of an object by writing $\langle \text{object} \rangle.p$: by this we mean that part of the BNF product with the name p. We use the Kleene star * to denote zero or more of a particular item and the symbol + for one or more of a particular item.

2.1 Specs

The central building block of the *rxncon* language is a molecule specification or *spec*, of which the BNF definition is given in (1). They appear as elements of reaction and state statements, in which they are used to specify properties of molecules.

$$\begin{aligned}
\langle \text{Spec} \rangle &\models \langle \text{Component} \rangle\ [\text{`@'}\ \langle \text{StructureIndex} \rangle]\ [\text{`_'}\ \langle \text{Locus} \rangle] \qquad (1)\\
\langle \text{Component} \rangle &\models \langle \text{Protein} \rangle \mid \langle \text{mRNA} \rangle \mid \langle \text{Gene} \rangle\\
\langle \text{StructureIndex} \rangle &\models \text{`0'} \mid \text{`1'} \mid \text{`2'} \mid \ldots\\
\langle \text{Protein} \rangle &\models \textit{Protein name}\\
\langle \text{mRNA} \rangle &\models \langle \text{Protein} \rangle\ \text{`mRNA'}\\
\langle \text{Gene} \rangle &\models \langle \text{Protein} \rangle\ \text{`Gene'}\\
\langle \text{Locus} \rangle &\models \text{`['}\ [\langle \text{Domain} \rangle]\ [\text{`('}\langle \text{Residue} \rangle\text{`)'}]\ \text{`]'}\\
\langle \text{Domain} \rangle &\models \textit{Domain name}\\
\langle \text{Residue} \rangle &\models \textit{Residue name}
\end{aligned}$$

The required *Component* denotes the particular protein, gene or mRNA that is referred to. Protein names are composed of alphanumeric characters, but have to start with a letter and not end in –Gene or –mRNA, which automatically refer to the gene or mRNA molecule corresponding to the protein. This one-to-one-to-one relation makes implementation of reactions that rely on the central dogma, *i.e.* translations and transcriptions, straightforward.

The optional *Locus* points to a location on a molecule, in order of increased resolution: to a *domain* or a *residue*. Domains can contain residues. This construction allows one to accurately reflect the detail of experimental knowledge: *e.g.* one might not know the precise residue at which a protein needs to be phosphorylated in order for a certain reaction to be possible, but only the domain on which the phosphorylation lives. If a residue is specified, the spec's *resolution* is "at the residue level", and similar for domain. If no Locus is provided, the spec's resolution is "at the component level".

Larger molecular complexes that can appear in contingencies might have multiple subunits containing the same molecule. In such a case, there is an ambiguity when combining different contingencies based solely on the component names of the molecules. To work around this problem, we introduce an additional *Structure index*, a number unique for each molecule.

Specs have a superset / subset relation amongst each other. The spec A is a subset of a spec B if

- A's and B's Component and StructureIndex match, and
- A's resolution is equal or higher than B's, and
- the Locus information in B that is not empty coincides with that in A.

The spec A is a superset of a spec B if B is a subset of A. Trivially a spec is its own superset and its own subset.

2.2 States

States correspond to independent *observable quantities*, such as protein's phosphorylation or bond to another protein. What is called "state" in the literature often refers to the fully specified microstate of a molecule. A *rxncon* state is a macroscopic state: except for the information on *e.g.* a phosphorylation, all other information is ignored (or "traced out" in statistical physics parlance).

In this section we discuss the different properties that states can have, and the different classes of states that appear in the *rxncon* language.

States belong to a certain *Class*. Currently we distinguish six classes in *rxncon*, see Table 1. *Modifications* such as `A_[(r)]-{p}` denote a modification of a particular residue, such as phosphorylations. *Interactions* such as `A_[a]--B_[b]` describe bound states between different molecules. *SelfInteractions* such as `A_[x]--[y]` describe bound states within the same molecule. *EmptyBindings* such as `A_[x]--0` describe an unbound (empty) binding domain on a molecule. *Inputs* such as `[Turgor]` describe a macroscopic input signal that cannot be localised on a single molecule. The special *FullyNeutral* state will be discussed below.

We distinguish states that are *located* on a single molecule, Modifications, SelfInteractions and EmptyBindings, ones that are located on a pair of molecules, Interactions, and non-localizable Inputs.

States are built up of zero or more specs or loci and inherit the notion of resolution from them. Each class of state has for every spec or Locus an associated *elemental resolution*. If every spec and locus that appears in a state is at its elemental resolution, the state itself is referred to as an *elemental state*.

States inherit the superset / subset relation from the specs they contain. A state S_1 is a subset of a state S_2 if

- they belong to the same class, and
- all non-spec properties coincide, and
- all specs in S_1 are subsets of the specs in S_2.

For classes of states that contain more than one spec, we consider the meaning of two states to coincide under permutation of the specs, *i.e.* `A_[a]--B_[b]` is equivalent to `B_[b]--A_[a]`.

There exists a notion of *mutual exclusivity* of states: the same residue on the same molecule cannot simultaneously be in the phosphorylated and unmodified form. An overview of which states are mutually exclusive with which can be found in Table 1. Note that elementarity of states is assumed here.

Every state has one or more "neutral" counterparts, for Modifications this is a Modification with the neutral Modifier, and for (Self)Interactions the appropriate EmptyBindings. Reactions that synthesise components mostly do so in a fully neutral combination of states, the FullyNeutralState which we denote by "0". This state is in fact a shorthand for the combination of all the neutral states for a particular component, see Section 2.6.

2.3 Syntax of reactions

The states we have seen in the previous section are created and destroyed by *elemental Reactions*. The syntax, which is presented in (2), contains two specs and a ReactionType.

$$\begin{aligned} \langle \text{Reaction} \rangle &\models \langle \text{Spec} \rangle \text{ '\_' } \langle \text{ReactionType} \rangle \text{ '\_' } \langle \text{Spec} \rangle \quad (2) \\ \langle \text{ReactionType} \rangle &\models \text{'p+'} \mid \text{'p-'} \mid \text{'ppi+'} \mid \ldots \end{aligned}$$

For a (non-exhaustive, but representative) list of *rxncon* Reactions, see Table 2. The skeleton rule that determines the semantics is explained in the following section.

2.4 Semantics of reactions: skeleton rules

Several languages, such as BNGL Faeder *et al.* (2009) and Kappa (Danos *et al.*, 2007) exist to formulate rule-based models. Here we briefly define the skeleton rule language: a simple language that is used to define the semantics of the *rxncon* reactions in terms of previously introduced *rxncon* concepts.

$$\begin{aligned} \langle \text{SkeletonRule} \rangle &\models \langle \text{Terms} \rangle \text{ '->' } \langle \text{Terms} \rangle \\ \langle \text{Terms} \rangle &\models \langle \text{Term} \rangle \mid \langle \text{Term} \rangle \text{ '+' } \langle \text{Terms} \rangle \\ \langle \text{Term} \rangle &\models \langle \text{Components} \rangle \text{ '\#' } [\langle \text{States} \rangle] \\ \langle \text{Components} \rangle &\models \langle \text{Component} \rangle \mid \langle \text{Component} \rangle \text{ '!' } \langle \text{Component} \rangle \\ \langle \text{States} \rangle &\models \langle \text{State} \rangle \mid \langle \text{State} \rangle \text{ '!' } \langle \text{States} \rangle \end{aligned}$$

The rule describes a transition between a number of terms at its left-hand side into a number of terms at its right-hand side. Every term consists of (*i*) one or more *Components*, which are connected in a complex and (*ii*) zero or more elemental states. The latter define the internal state of the molecules in the complex.

Given a Reaction and its skeleton rule, one can define the notions of *production*, *consumption*, *synthesis* and *degradation* of states by Reactions, where RHS and LHS refer to the right-hand side and left-hand side of the corresponding skeleton rule:

- a state is produced by a reaction if it appears on the RHS, not on the LHS, but the component carrying the state does appear on the LHS,
- a state is consumed by a reaction if it appears on the LHS, not on the RHS, but the component carrying the state does appear in the RHS,
- a state is synthesised by a reaction if it appears on the RHS, and the component carrying the state does not appear on the LHS,
- a state is degraded by a reaction if no state mutually exclusive with it appears on the LHS, and the component carrying the state does not appear on the RHS.

2.5 Contingencies

The context for reaction events is given by *contingencies*, see (3). These are (Boolean combinations of) states that influence the reaction events.

$$\begin{aligned} \langle \text{Contingency} \rangle &\models \langle \text{Subject} \rangle \text{ ',' } \langle \text{Verb} \rangle \text{ ',' } \langle \text{Object} \rangle \quad (3) \\ \langle \text{Subject} \rangle &\models \langle \text{Reaction} \rangle \\ &\quad \mid \text{'<'} \langle \text{BooleanContingency} \rangle \text{ '>'} \\ &\quad \mid \text{'['} \langle \text{Output} \rangle \text{ ']'} \\ \langle \text{BooleanContingency} \rangle &\models \textit{Boolean contingency name} \\ \langle \text{Output} \rangle &\models \textit{Output name} \\ \langle \text{Verb} \rangle &\models \langle \text{BooleanOperator} \rangle \mid \langle \text{ContingencyType} \rangle \\ \langle \text{BooleanOperator} \rangle &\models \text{'AND'} \mid \text{'OR'} \mid \text{'NOT'} \\ \langle \text{ContingencyType} \rangle &\models \text{'!'} \mid \text{'x'} \mid \text{'?'} \mid \text{'0'} \mid \text{'k+'} \mid \text{'k-'} \\ \langle \text{Object} \rangle &\models \langle \text{State} \rangle \\ &\quad \mid \langle \text{BooleanContingency} \rangle \langle \text{StructureEquivalences} \rangle * \\ \langle \text{StructureEquivalence} \rangle &\models \text{'\#'} \langle \text{NamespacedComponent} \rangle \text{ '='} \langle \text{NamespacedComponent} \rangle \\ \langle \text{NamespacedComponent} \rangle &\models [\langle \text{Namespace} \rangle] \langle \text{Component} \rangle \langle \text{StructureIndex} \rangle \\ \langle \text{Namespace} \rangle &\models \langle \text{BooleanContingency} \rangle \text{ '.'} \\ &\quad \mid \langle \text{BooleanContingency} \rangle \text{ '.'} \langle \text{Namespace} \rangle \end{aligned}$$

We distinguish *strict contingencies*, with ContingencyType "!" for an absolute requirement and ContingencyType "x" for an absolute inhibition, and *quantitative contingencies*, with ContingencyType "k+" representing a positive contribution to the reaction rate and ContingencyType "k-" a negative contribution.

Finally, the ContingencyTypes "0" and "?" denote *no effect* respectively *unknown effect*.

When all of a reaction's contingencies are satisfied, the signalling network is considered to be in a state that can accomodate the reaction. For a reaction to be considered active, the network needs to be in this state, the reaction's reactants need to be present and its sources (the states it targets for consumption) need to be present.

Contingencies inherit the notion of elementarity from the states they contain: if all states are elemental, the contingency is elemental and otherwise not.

2.5.1 Satisfiability of contingencies

Since contingencies can form Boolean expressions of states, it is important that they are satisfiable. The reference implementation of *rxncon* is linked to picoSAT (Biere, 2008), an industrial-strength satisfiability solver.

Every contingency can (and will, in practice) be expanded into an elemental contingency (see Section 2.6). It is therefore sufficient to consider satisfiability of elemental contingencies. However, not every naively obtained solution to a Boolean expression over states is a valid solution: some states are mutually exclusive with one another, and are therefore not allowed.

Furthermore, a contingency needs to be *connected* to the reactants: if it refers to a molecule that is not one of the reactants there needs to be at least one path from the reactants to that molecule over bond states to be valid. This in particular becomes an issue when translating a *rxncon* system to rules in a rule-based model (in preparation). A Boolean contingency is *satisfiable* if it has at least one solution that contains no mutually exclusive states and is connected.

2.5.2 Structured indices and boolean contingencies

In many cases, the name of a molecule might not be sufficient to uniquely identify it in a complex, which is solved by adding structure indices to specs. The *rxncon* reference implementation has an algorithm to find

Table 1. Classes of states in the rxncon language. We denote the BNF definition, the resolution of the respective specs and loci at which the state becomes elemental, the states with which they are mutually exclusive and their neutral counterparts.

Class	BNF definition	BNF definition part	Elemental resolution	Mutually exclusive with	Neutral state(s)
Modification	⟨Substrate⟩ '-{' ⟨Modifier⟩ '}'	⟨Substrate⟩ ⊨ ⟨Spec⟩ ⟨Modifier⟩ ⊨ 0 | p | ...	Residue n/a	• Modification with equal ⟨Substrate⟩, different ⟨Modifier⟩	⟨Substrate⟩ '-{0}'
Interaction	⟨FirstMol⟩ '--' ⟨SecondMol⟩	⟨FirstMol⟩ ⊨ ⟨Spec⟩ ⟨SecondMol⟩ ⊨ ⟨Spec⟩	Domain Domain	• All Interaction, SelfInteraction, and EmptyBinding mutually exclusive if any domain matches	⟨FirstMol⟩ '--0' and ⟨SecondMol⟩ '--0'
SelfInteraction	⟨FirstDom⟩ '--' ⟨SecondDom⟩	⟨FirstDom⟩ ⊨ ⟨Spec⟩ ⟨SecondDom⟩ ⊨ ⟨Locus⟩	Domain Domain		⟨FirstDom⟩ '--0' and ⟨FirstDom⟩.*component*'_'⟨SecondDom⟩ '--0'
EmptyBinding	⟨Mol⟩ '--0'	⟨Mol⟩ ⊨ ⟨Spec⟩	Domain		
Input	'[' ⟨Name⟩ ']'	⟨Name⟩ ⊨ *Input State name*	n/a	• Inputs are not mutually exclusive	
FullyNeutral	0	n/a	n/a	• n/a (expanded to other states)	

Table 2. Classes of Reactions in the rxncon language. Here we denote the specs appearing in the Reaction as computer-code style variables `$x`, `$y` to allow a "method call" syntax on them which extracts a particular part of their BNF expression. For the domain-specific language expressing the rules, please see the text.

Reaction	Class	`$x`	`$y`	Base *rxncon*-intermediate rule
`$x_p+_$y`	Phosphorylation	Protein, Component	Protein, Residue	`$x# + $y#$y-{0} -> $x# + $y#$y-{p}`
`$x_p-_$y`	Dephosphorylation	Protein, Component	Protein, Residue	`$x# + $y#$y-{p} -> $x# + $y#$y-{0}`
`$x_ub+_$y`	Ubiquitination	Protein, Component	Protein, Residue	`$x# + $y#$y-{0} -> $x# + $y#$y-{ub}`
`$x_ap+_$y`	Auto-phosphorylation	Protein, Component	Protein, Residue	`$y#$y-{0} -> $y#$y-{p}`
`$x_ap-_$y`	Auto-dephosphorylation	Protein, Component	Protein, Residue	`$y#$y-{p} -> $y#$y-{0}`
`$x_ppi+_$y`	Protein-protein-bind	Protein, Domain	Protein, Domain	`$x#$x--0 + $y#$y--0 -> $x!$y#$x--$y`
`$x_ppi-_$y`	Protein-protein-unbind	Protein, Domain	Protein, Domain	`$x!$y#$x--$y -> $x#$x--0 + $y#$y--0`
`$x_pt_$y`	Phosphotransfer	Protein, Residue	Protein, Residue	`$x#$x-{p} + $y#$y-{0} -> $x#$x-{0} + $y#$y-{p}`
`$x_trsc_$y`	Transcription	Protein, Component	Gene, Component	`$x# + $y# -> $x# + $y# + $y.mrna#0`
`$x_trsl_$y`	Translation	Protein, Component	mRNA, Component	`$x# + $y# -> $x# + $y# + $y.protein#0`

reasonable default structure indices if none are supplied, and internally every spec in the contingency list carries a structure index once the *rxncon* system has been constructed.

When one defines contingencies that contain Boolean expressions or nested Booleans (Boolean contingencies containing Boolean contingencies), there is an additional ambiguity. The structure indices of a Boolean contingency live in a namespace that is labelled by the name of that particular Boolean contingency. Within that namespace every structure index is well-defined, but one has to map the indices within the namespace of the Boolean contingency to the subject namespace. This applies to contingencies that have a reaction as their subject as well as contingencies that themselves have a Boolean contingency as their subject: when one combines multiple Boolean contingencies, the namespaces have to be merged to obtain an unambiguous labelling.

The following rules apply:

- for monomolecular reactions, the reactant has structure index 0,
- for bimolecular reactions, the reactants have indices 0 and 1,
- when a contingency (with a reaction or a Boolean contingency as its subject) has a Boolean contingency as its object, a *structure equivalence* has to be supplied. This equivalence relation establishes which (Component, StructureIndex) pairs in the subject namespace map to which (Component, StructureIndex) pairs in the object namespace. As an example, the equivalence `#A@0=A@2` means that the component `A@0` in the subject's namespace refers to the same molecule as `A@2` in the Boolean contingency's namespace.

2.6 *rxncon* system

A full *rxncon* system is a set of one or more Reactions and zero or more Contingencies, see (4).

$$\langle \text{RxnConSystem} \rangle \models \langle \text{Reaction} \rangle + \langle \text{Contingency} \rangle * \quad (4)$$

After reading a *rxncon* system, one first *finalizes* the system an then validates it.

The finalization concerns (1) the expansion of non-elemental contingencies and (2) the structuring of non-structured contingencies. The first happens in two places. It is possible to formulate contingencies in terms of non-elemental states, whereas Reactions by definition only produce, consume, synthesise and degrade elemental states. To handle this mismatch, every non-elemental state appearing in a contingency becomes a Boolean 'OR' complex carrying the name of the non-elemental contingency.

Furthermore, the FullyNeutral state needs to be expanded. This state appears in synthesis reactions and is useful since it is a property of the entire *rxncon* system what the neutral state for a component exactly is.

Finally, a validation takes place. Not every *rxncon* system is internally consistent. However, this can only be decided after finalisation. The validation checks that

- there are no elemental states appearing in the contingencies that are not produced, consumed, synthesised or degraded by elemental reactions,
- there are no reactions that are the subject of contingencies that are not in the list of reactions,
- there are no unsatisfiable contingencies.

3 Examples

Consider the *rxncon* system with the following reactions

```
A_[a]_ppi+_B_[b]                    (5)
A_[a]_ppi+_C_[c]
C_p+_A_[(r1)]
D_p+_A_[(r2)]
```

This system describes two forward protein-protein-interaction reactions, between `A` and `B`, and `A` and `C` respectively, and two modification reactions: `C` and `D` phosphorylating `A` at two different residues.

In what follows we present several aspects of the *rxncon* language that have been introduced in the main body of this work, by studying the same set of reactions with different sets of contingencies.

3.1 Non-elemental contingency

Contingencies may contain non-elemental states, for example a phosphorylation requirement without specificity as to which residue is modified. This state should be expanded into a disjunction of (possibly multiple) elemental state(s), to enable further analysis or simulation. Which elemental states are candidates for this is a system-wide property however, so this expansion can only be performed once the entire system is known.

Consider system (5) with the contingency

```
A_[a]_ppi+_B_[b],!,A@0-{p}
```

This contingency is non-elemental and gets expanded into

```
A_[a]_ppi+_B_[b],!,<A-{p}>#A@0=A@0
<A-{p}>,OR,A@0_[(r1)]-{p}           (6)
<A-{p}>,OR,A@0_[(r2)]-{p}
```

3.2 Non-satisfiable contingency

Boolean contingencies may contain statements that are not satisfiable. Consider the contingency

```
C_p+_A_[(r1)],!,<DOUBLEBOND>#A@1=A@0#C@0=C@2
<DOUBLEBOND>,!,A@0_[a]--B@1_[b]
<DOUBLEBOND>,!,A@0_[a]--C@2_[c]
```

This contingency has one solution, namely `A@1_[a]--B@3_[b]` and `A@1_[a]--C@0_[c]` both true (note that we have lifted the structural indices out of the Boolean and into the reaction's namespace). These two states are mutually exclusive with one another, since the `A@1_[a]` binding domain cannot be doubly-bound. This statement should therefore be rejected.

3.3 Nested boolean contingency

In larger systems, it is useful to reuse certain larger Boolean contingencies. For example the same complex or multiple phosphorylation might be a precondition for several reactions. This generalizes to reusing Boolean contingencies within Boolean contingencies. Consider the following set of contingencies

```
A_[a]_ppi+_B_[b],!,<A-PHOS-BOUND>#A@0=A@1#B@1=B@0   (7)
<A-PHOS-BOUND>,AND,<A-{p}>#A@1=A@0                  (8)
<A-PHOS-BOUND>,AND,A@1_[a]--B@0_[b]                 (9)
```

Here we reuse the `<A-{p}>` contingency (6). Within the namespace of the reaction, `A` has structure index 0 since it is the first reactant and `B` index 1 since it is the second. The equivalence statement in (7) states that this `A` coincides with `<A-PHOS-BOUND>`'s `A@1`, which by (8) coincides with `<A-{p}>`'s `A@0`. The `B@0` defined in coincides with `B@1` in the reaction.

4 Discussion and conclusion

We have presented the syntax and semantics of *rxncon*, the reaction-contingency language for the description of cellular signalling processes. As it stands, the language is suited for knowledge consolidation and standardization. However, in upcoming work we will present the translation of *rxncon* systems to both qualitative bipartite Boolean models (Thieme *et al.*, 2017) and quantitative rule-based models (in preparation). Both have their domain of applicability and strengths. Boolean simulations require no knowledge about the functional form of reaction rate laws, reaction constants and relative concentrations – the type of quantitative knowledge that is often lacking. As it turns out, the functionality of signalling networks is often not dependent on such details which makes the Boolean models excellent territory for initial model validation. Rule-based modelling (Faeder *et al.*, 2009) is a very natural fit for *rxncon*: both approaches adhere to a form of the "don't care: don't write" principle in which information regarding the state of reactants that is unknown or unimportant is left out of the description.

This work provides a major upgrade to the previous version of the *rxncon* language (Tiger *et al.*, 2012). The notion of resolution of specs and thereby states is novel, and the elementarity of states and their mutual exclusivity was not considered in detail in the previous release. Still, even without these concepts, the language performed well (Flöttmann *et al.*, 2013). This leads one to think that natural processes are rather robust with regards to changes in detail.

Concluding, we have paved the way for genome-scale modelling of signal transduction networks in living cells. As mechanistic understanding of these systems grows so will the applications, in particular in the medical field where many diseases have been shown to be related to malfunctioning networks (Hanahan and Weinberg, 2011; López-Otín *et al.*, 2013). Several theoretical challenges remain: in upcoming work we present the precise translation from *rxncon* to qualitative Boolean and quantitative rule-based models. Furthermore several elements are still missing in the language that are crucial for (mammalian) signalling processes, in particular *localisation* and *allele effects*. These will be subject of further study.

Acknowledgements

The authors thank Janina Linnik, Mikołaj Rybiński and Alexander Bockmayr for fruitful discussions. Without them this work would not have been possible.

Funding

This work was supported by the German Federal Ministry of Education and Research as e:Bio Cellemental (FKZ0316193, to MK).

References

Backus, J. W. (1959). The syntax and semantics of the proposed international algebraic language of the Zurch ACM-GAMM Conference. In *Information Processing: Proceedings of the International Conference on Information Processing, Paris*, pages 125–132. UNESCO.

Biere, A. (2008). Picosat essentials. *JSAT*, **4**, 75–97.

Danos, V. and Laneve, C. (2004). Formal molecular biology. *Theoretical Computer Science*, **325**(1), 69 – 110.

Danos, V., Feret, J., Fontana, W., Harmer, R., and Krivine, J. (2007). *Rule-Based Modelling of Cellular Signalling*, pages 17–41. Springer Berlin Heidelberg, Berlin, Heidelberg.

Faeder, J. R., Blinov, M. L., Goldstein, B., and Hlavacek, W. S. (2005). Rule-based modeling of biochemical networks. *Complexity*, **10**(4), 22–41.

Faeder, J. R., Blinov, M. L., and Hlavacek, W. S. (2009). *Rule-Based Modeling of Biochemical Systems with BioNetGen*, pages 113–167. Humana Press, Totowa, NJ.

Flöttmann, M., Krause, F., Klipp, E., and Krantz, M. (2013). Reaction-contingency based bipartite boolean modelling. *BMC Systems Biology*, **7**(1), 58.

Hanahan, D. and Weinberg, R. A. (2011). Hallmarks of cancer: The next generation. *Cell*, **144**(5), 646–674.

Hlavacek, W. S. and Faeder, J. R. (2009). The complexity of cell signaling and the need for a new mechanics. *Science Signaling*, **2**(81), pe46–pe46.

Hlavacek, W. S., Faeder, J. R., Blinov, M. L., Perelson, A. S., and Goldstein, B. (2003). The complexity of complexes in signal transduction. *Biotechnology and Bioengineering*, **84**(7), 783–794.

López-Otín, C., Blasco, M. A., Partridge, L., Serrano, M., and Kroemer, G. (2013). *Cell*, (6).

Naur, P. (1961). A course of algol 60 programming. *ALGOL Bull.*, (Sup 9), 1–38.

Rother, M., Munzner, U., Thieme, S., and Krantz, M. (2013). Information content and scalability in signal transduction network reconstruction formats. *Mol. BioSyst.*, **9**, 1993–2004.

Thiele, I. and Palsson, B. O. (2010). A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nat. Protocols*, **5**(1), 93–121.

Thieme, S., Romers, J., Münzner, U., and Krantz, M. (2017). Bipartite boolean modelling - a method for mechanistic simulation & validation of large-scale signal transduction networks.

Tiger, C.-F., Krause, F., Cedersund, G., Palmér, R., Klipp, E., Hohmann, S., Kitano, H., and Krantz, M. (2012). A framework for mapping, visualisation and automatic model creation of signal-transduction networks. *Molecular Systems Biology*, **8**(1).