Supplementary information

Overall structure

`sbml-diff` is implemented as a python package which receives SBML models as strings, and produces output in the DOT language specifying how the requested diagram should be drawn. The command-line program is a python script that provides a wrapper around this package.

The web interface accessible at http://sysos.eng.ox.ac.uk/tebio/upload is also implemented in Python, using the Flask microframework. When files are uploaded through the form, they are saved to a directory on the web-server. The `sbml-diff` package is used to perform the comparison, and the resulting DOT output is saved to disk. Graphviz is then used to produce images in both PNG and PDF formats.

Representing events with multiple assignments

If an event includes multiple assignment elements, it is represented as a rectangle containing a separate parallelogram node representing each assignment, and a single diamond node representing the trigger statement:

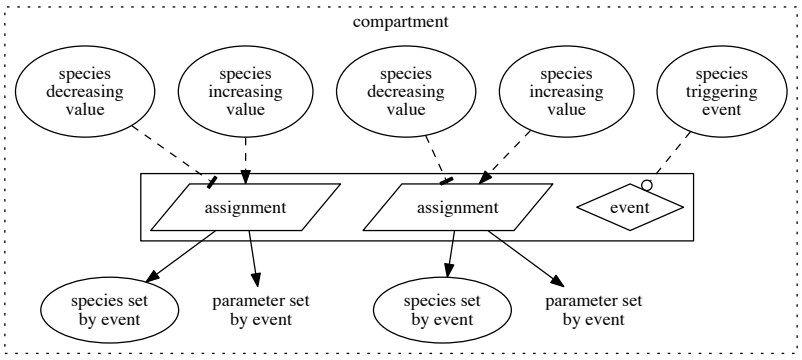


Figure 1: Representation of an event containing two assignment elements

Multiple representations of a single model

Whilst the paper focuses mostly on comparison between multiple models, `sbml-diff` can also be applied to a single model. BioModels entry BIOMD0000000012 is a model of the repressilator. `sbml-diff` can provide several views of this: the full view (Figure 2), makes it clear that the parameters `beta` and `alpha0` have values assigned by rules, but are not otherwise used; the full view with rules hidden (Figure 3) shows the structure of the chemical reactions more clearly; and the cartoon view (Figure 4) further clarifies the structure of the genetic network.

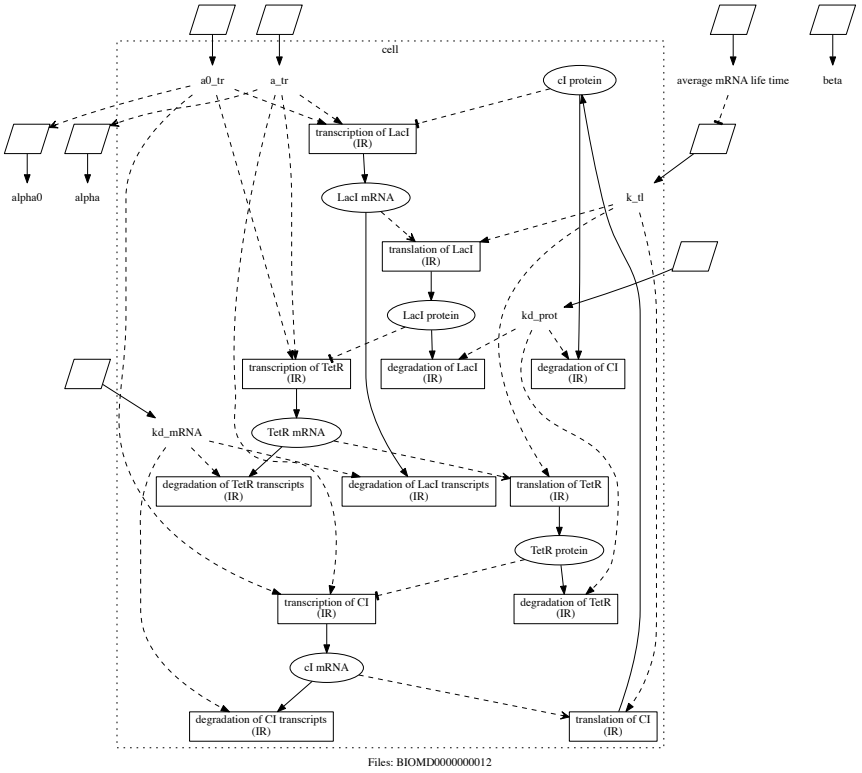
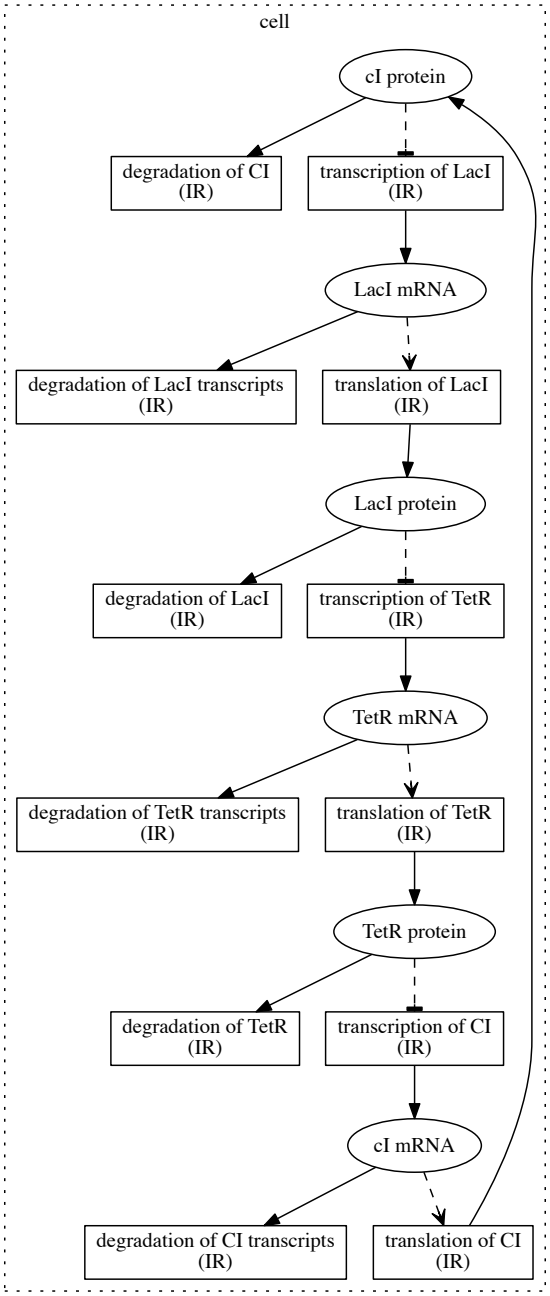


Figure 2: It can be seen that the parameters `beta` and `alpha0` have values assigned by rules, but are not otherwise used



Files: BIOMD0000000012

Figure 3: In this case, hiding the rules (with the `--hide-rules` command-line flag), reveals the structure of the chemical reactions more clearly

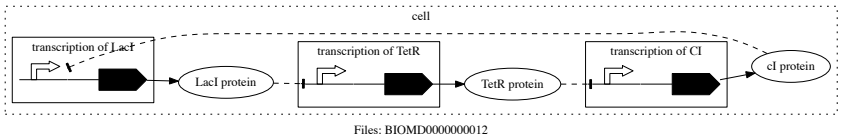


Figure 4: The cartoon view with rules hidden represents less of the model's mathematical structure, but shows the structure of the genetic network more clearly)

Determination of arrow direction

In an SBML model, a `reaction` can have a `listOfModifiers` element, listing species that appears in the kinetic rate formula of a reaction, but do not appear in the list of reactant or products. SBML itself does not specify a way to record whether this interaction is activatory or inhibitory, but it is possible for the modeler to explicitly indicate the role of the modifier species by annotating it using the appropriate term from the Systems Biology Ontology: either SBO:0000459 (stimulator), SBO:0000020 (inhibitor), or one of their more-specific child-terms. However, such annotations are often lacking from models. For example, of the 612 curated models in the 2016-05-10 release of the BioModels database, few contain any annotations indicating the type of modifier: 7 use SBO:0000020 (inhibitor), 2 use SBO:0000206 (competitive inhibitor), 2 use SBO:0000207 (non-competitive inhibitor), 5 use SBO:0000459 (stimulator), 20 use SBO:0000461 (essential activator), and 3 use SBO:0000462 (non-essential activator). Additionally, there are 'shorthand' languages that facilitate writing SBML models by hand (such as Antimony and SBML-shorthand) which do not support annotations with SBO terms.

In the absence of explicit information, the simplest (and most conservative) approach is to simply draw arrows with generic arrowheads, indicating that the effect of the interaction is unknown. However, such ambiguity would considerably reduce the usefulness of the diagrams, and so we attempt to determine which arrowhead to draw by examining the reaction's `kineticLaw`. The heuristic used may occasionally give misleading results[1], but we believe that the benefit of being able to correctly draw arrowheads in most cases outweighs the risk of sometimes getting it wrong.

A naive approach would be recursively transverse the tree of arithmetic operations, keeping track of whether the expression at each stage is a monotonically increasing, monotonically decreasing, constant, or non-monotonic function of the modifier species, assuming that all parameter values and concentrations are non-negative. Such a procedure works for some functions, such as the Hill repression function: as x and n are non-negative, x^n is a monotonically increasing function of x, so $(K_m + x^n)$ is a monotonically increasing function of x, and $1/(K_m + x^n)$ is a monotonically decreasing

[1] The arrowhead may be misleading if the arrow direction depends on the value of parameters and the SBML file does not set the `value` attribute on each parameter, or if the `kineticLaw` is not monotonic. In the latter case, the arrowhead is correct for *some* values of the concentration vector, but not for *all*.

function of x. However, such an approach fails for the commonly encountered Hill activation function $x^n/(K_m + x^n)$: both the numerator and denominator of this fraction are monotonically increasing functions of x, so this is an indeterminate case. To determine whether this is an increasing or decreasing function, we would need to differentiate symbolically, and test whether the resulting expression $(nK_m x^{n-1}/(K_m + x^n)^2)$ has a constant sign for all non-negative parameter values and concentrations[2]. Whilst the first task can be easily achieved in Python using the SymPy library, the second is more challenging and SymPy's `Assumptions` module is not currently sufficiently powerful for this task.

As it is difficult to write a program that can verify whether an arbitrary input function is monotonic we do not attempt to do this, and instead simply assume the kinetic-law is a monotonic function of each modifier species. Most of the commonly chosen functions are monotonic (e.g. linear, Hill activation, Hill repression, Michaelis-Menten). There are times when such an assumption may be mistaken[3], but such cases are relatively unusual.

Under such an assumption, it is sufficient to evaluate the kinetic-law at two values of the modifier species concentration, and the choice of these values will not affect the determined direction.

It is also necessary to choose values for the parameters (and other concentrations). However, the choice of these values could affect the determined direction: for example, if the Hill coefficient is negative, then the Hill activation function becomes monotonically decreasing, rather than monotonically increasing. Fortunately, this choice is not important in many of the most common cases: provided that all parameters and concentrations are positive, Hill activation, Hill repression and Michaelis-Menten kinetic laws are monotonically increasing, decreasing and increasing (respectively) functions, regardless of the precise numerical values.

If the model specifies the value of a parameter (or initial concentration of a species), we use that numerical value. Otherwise, we arbitrarily use a value of one. The kinetic law is then compared at two arbitrarily chosen values of the modifier's species concentration (0.1 and 1).

If a numerical error (e.g. a division-by-zero error) is encountered during one of the evaluations of the `kineticLaw`, the interaction is treated as indeterminate and represented by a diamond arrowhead.

For a monotonic `kineticLaw`, this method is equivalent to testing the sign of a finite difference approximation to the Jacobian of the system at a single point.

[2] Or, if parameter values are known, that it has a constant sign for all concentrations

[3] For example, the 'species concentration' might actually represent cell density, and a component of growth medium may increase growth rate when provided at low concentration, but decrease the growth rate when at high concentration due to toxicity or osmotic effects