Logical modelling and analysis of cellular regulatory networks with GINsim 2.9.3

Wassim Abou-Jaoudé[1,2,3], Denis Thieffry[1,2,3,*]

[1] Institut de Biologie de l'Ecole Normale Supérieure (IBENS), Paris, France. [2] CNRS UMR 8197, Paris, France. [3] INSERM UMR 1024, Paris, France.

Email: wassim@biologie.ens.fr, thieffry@ens.fr;

*Corresponding author

Abstract

The logical formalism is well adapted to model large biological networks, for which detailed kinetic data are scarce. This tutorial focuses on a well-established qualitative (logical) framework for the modelling of regulatory networks. Relying on GINsim, a software implementing this logical formalism, we guide the reader step by step towards the definition and the analysis of a simple model of the mammalian p53-Mdm2 network.

Keywords

Biological networks, logical modelling, qualitative analysis, regulatory circuit, regulatory graph, state transition graph, p53-Mdm2 network.

1 Introduction

The logical formalism is becoming increasingly popular to model cellular networks [1]. Here, we focus on the framework developed by René Thomas and colleagues, which includes the use of multilevel variables when functionally justified, along with sophisticate logical rules or parameters [2, 3]. This approach has been applied to the study of a wide range of networks controlling, for example, the lysis-lysogeny decision of the bacteriophage λ [4], the specification of flower organs in arabidopsis [5], the segmentation of drosophila embryo [6–8], the formation of compartment in drosophila imaginal disks [9, 10], drosophila egg shell patterning [11], cell cycle in mammals and yeast [12, 13], the differentiation of T-helper lymphocytes [14, 15], neural differentiation [16], as well as cell fate decisions in tumors [17–19], etc. Our goal here is to introduce this modelling framework and the use of the current version of the software GINsim, which implements this formalism. The chapter is organised as follows. Section 2 provides practical information on GINsim and file formats. General guidelines to define and analyse logical models are given in Section 3. Next, in Section 4, we proceed with the construction and analysis of a logical model for a network centered around p53 and Mdm2 proteins and their role in DNA damage repair. The chapter ends with an outlook section.

2 Material

2.1 GINsim

GINsim software supports the definition, the simulation and the analysis of regulatory graphs, based on the (multilevel) logical formalism. Developed in Java, GINsim is platform-independent and only requires a recent Java Virtual Machine.

GINsim is freely available for academic users and is available from the GINsim website (`http://ginsim.org`), along with documentation and a model repository. We will use the development release GINsim-2.9.3 for this tutorial. To get started with GINsim, download the Java ARchive (JAR file), from the download section of GINsim website. To launch GINsim, double-click on the file icon or use the command: `java -jar GINsim-#version.jar` (java options are available, e.g. to increase the size of RAM to be used).

You are now ready to define a logical regulatory graph, or to open a pre-existing logical model and analyse its dynamical properties.

2.2 File formats

To store logical regulatory graphs and state transition graphs, including graphical attributes, GINsim relies on a specific XML file format called GINML. In addition to these flat files (with extension `.ginml`), ZIP archives can be generated (extension `.zginml`), including files containing definitions or mutants, initial states, and other simulation parameters (files `mutant`, `initialState`, `reg2dyn_parameters`, respectively), in addition to the flat file containing the

model (file `regulatoryGraph.ginml`).

GINsim further supplies facilities for export in a number of file formats, including the new SBML qual format [20] (see Note 1).

3 Logical regulatory graph definition and analysis

In this section, we introduce the different steps to follow to define a logical model of a regulatory network and to analyse its dynamical properties. These definitions are applied to the p53-Mdm2 network in the Section 4.

3.1 Defining a logical regulatory graph

The first step in the delineation of a logical model of a regulatory network is the definition of a *logical regulatory graph*, where the nodes represent regulatory components (genes, proteins, etc.), whereas the arcs represent regulatory interactions.

1. Define a set of regulatory components $\{g_0, \ldots, g_n\}$. To each component g_i ($i \in \{0, \ldots n\}$), associate a *maximal level* max_i, defining a range of functional qualitative levels $\{0, \ldots max_i\}$ for g_i.

2. Define a set of arcs between nodes. An arc represents a regulatory effect of its *source* node onto its *target* node. For each arc, assign a *threshold* $\theta \in \{1, \ldots max_i\}$ (max_i being the maximal level of g_i, the source of the arc). This threshold indicates the level of g_i at which the regulatory effect on g_j, the target of the interaction, occurs. Further, assign a sign in

$\{+, -, ?, \pm\}$, where $+$ denotes an activation, $-$ an inhibition, ? an unknown sign, and $\pm$ a dual interaction (*i.e.*, an interaction with positive or negative effect depending on the presence of cofactors).

When the maximal level max_i of a node g_i is higher than 1, an outgoing arc pointing from g_i towards a node g_j may be composed of several interactions (*i.e.*, there is a multi-arc from g_i to g_j), denoting situations where the regulatory effect of g_i on g_j is different depending on the level of g_i (see Note 2).

3. In addition, one must define the rules governing the evolution of the regulatory component levels. These rules can be specified in the form of *Boolean formulae* or of *logical parameters* (see Note 3 for the definitions of logical parameters, basal levels and default values in GINsim). Here, we will use Boolean formulae (i.e. connecting literals and the Boolean operators NOT, AND, and OR) to specify the conditions on the regulators enabling the activation of each component.

3.2 Dynamical simulations

The dynamical behaviour of a logical regulatory graph is also defined as a graph, called *state transition graph*. In a *state transition graph*, each node represents a *state* of the model, i.e. a vector s with definite component levels (s_i, the ith component of s, is the level of g_i in state s). The arcs represent transitions between states. One core function of GINsim is the automatic construction of this graph based on the predefined logical regulatory graph and rules. To use

this function judiciously, it is important to understand the principles underlying this construction, which are outlined hereafter. Afterwards, several options to analyse the dynamics of logical regulatory graphs are presented.

A each state s, a specific combination of interactions are active (i.e. those for which the source levels are above the corresponding thresholds). These active interactions may enforce changes of component levels, depending on the logical rules. For each component g_i, if its current level s_i is different from the corresponding target value in state s, there is a call to update the level of g_i towards this target level. Several such components can be called for update at a given state. Two main strategies are then commonly used. Under *synchronous updating*, all concerned components change their levels simultaneously in a unique transition towards a single next state. In contrast, *asynchronous updating* generates a successor state for each component called for updating. If a state involves k updating calls, it will thus have k successors, and each successor state will differ from it by the level of a single component (see Note 4 for additional explanations). The introduction of priority classes allows to define more subtle updating schedules (see Note 5 and [12]).

For models of moderate size, one can choose to generate the complete *state transition graph*,considering all possible initial states. Alternatively, one can build a state transition graph for specific initial state(s).

In such state transition graphs, it is relatively easy to determine the stable states, defined as nodes with no outgoing arcs, as well as more complex attractors, defined as terminal maximal strongly connected components, denoting an

oscillatory behaviour.

Beyond the identification of stable states and of more complex attractors, we are particularly interested in knowing which attractors can be reached from specific initial conditions. Such questions can be addressed by verifying the existence of *pathway(s)* (i.e. sequences of transitions) from initial states to attractor states.

3.3 Strongly Connected Components Graph

To ease the analysis of the dynamics of a model, one can compress the state transition graphs into a graph of Strongly Connected Components (SCC). The SCC graph is an acyclic graph generated on the basis of the original one, such that each strongly connected component of the original graph is compressed in a single node of the SCC graph. Interestingly, the resulting SCC graph preserves the reachability properties of the original graph.

3.4 Hierarchical Transition Graph

In many situations, the SCC graph results only in a moderate compression of STG. To augment this compression and ease the interpretation of the dynamics, we have recently introduced another acyclic graph, called *Hierarchical Transition Graph* (HTG), which further merges linear chains of states (in addition to cycles) into single nodes [21]. The resulting graph preserves stable states and other important dynamical properties, but do not fully conserve reachability properties.

3.5 Circuit analysis

Regulatory circuits are responsible for the emergence of dynamical properties, such as multistationarity or sustained oscillations (see Note 6). In this respect, GINsim implements specific algorithms to:

1. Identify all the circuits of a regulatory graph (possibly considering constraints such as maximum length, consideration or exclusion of some components, etc.).

2. Determine the functionality contexts of these circuits, using a computational method presented in [22].

3.6 Reduction of logical models

When models increase in size, it become quickly difficult to cope with the size of the corresponding STG. One solution consists in simplifying or reducing the model before simulation. In this respect, GINsim implements a method to reduce a model on the fly, i.e. just before the simulation. The modeler can specify the nodes to be reduced, and the logical rules associated with their targets are then recomputed taking into account the (indirect) effects of their regulators. This construction of reduced models preserves crucial dynamical properties of the original model, including stable states and more complex attractors [23].

3.7 Definition of perturbations

Common perturbations are easily specified within the logical framework:

- A gene knock-down is specified by driving and constraining the level of the corresponding regulatory node to the value 0.
- Ectopic expression is specified by driving and constraining the level of the corresponding regulatory component to its highest value (or possibly to a range of values greater than zero, in the case of a multi-valued node).
- Multiple perturbations can be defined by combining several such constraints.
- More subtle perturbations can be defined by more sophisticate rewriting of component rules (*i.e.* to change the effect of a given regulatory arc).

4 Application to the p53-Mdm2 network

Here, we illustrate the use of GINsim through the construction and the analysis of a logical model for the p53-Mdm2 regulatory network.

4.1 p53-mdm2 network in mammals

The transcriptional factor p53 plays an essential role in the control of cell proliferation in mammals by regulating a large number of genes involved notably in growth arrest, DNA repair, or apoptosis. Its level is tightly regulated by the ubiquitin ligase Mdm2. More precisely, nuclear Mdm2 down-regulates the level of active p53, both by accelerating p53 degradation through ubiquitination and by blocking the transcriptional activity of p53. In return, p53 activates Mdm2 transcription and down-regulates the level of nuclear Mdm2 by inhibiting Mdm2

nuclear translocation through inactivation of the kinase Akt. Finally, high levels of p53 promote damage repair by inducing the synthesis of DNA repair proteins.

Given its key role in DNA repair and cell fate control, this network has been modeled by various groups using different formalisms, including ordinary differential equations [24], stochastic models [25], as well as the logical framework [26].

Here, we rely on the logical model presented in [26], which encompasses the following components: the protein p53; the ubiquitin ligase Mdm2 in the cytoplasm; the ubiquitin ligase Mdm2 in the nucleus; and DNA damage (see Figure 1).

4.2 Defining the logical regulatory graph

The set of instructions listed below defines a logical model, which can be saved in the GINML format as described in Section 2.2. To edit a graph, use the toolbox located just on the top of the main graphic window (below the scrolling menus). Passing slowly with the mouse on each of the editing tools pops a message explaining what this tool does. The "E" tool enables further edition of a pre-existing node or arc. The garbage can enables the deletion of a pre-existing arc or tool. Clicking once on one of the remaining tools activates it and enables the drawing of one node or one arc. Clicking twice on one of these tools will enable the drawing of several nodes or arcs without clicking again on the relevant tool.

1. Add the four nodes corresponding to p53, Mdm2cyt, Mdm2nuc and DNAdam,

specifying their names and maximal levels as defined in Table 1. The easiest way is to first double-click on the component addition tool and draw the four nodes, and then double-click on the Edition ("E") tool to change their names and maximal levels. Note that a node can also be ticked as input, but this is not relevant here, as all nodes are regulated by at least one other node of the network. Figure 1 illustrates this step.

2. Add the arcs between the nodes as defined in Figure 1, specifying their associated signs and thresholds, as list in the Table 2. Figure 2 illustrates this step. Note 2 provides further information about the possibility to associate different signs and value intervals with one arc.

3. For each node (select the node to edit it), specify the logical rules listed in Table 3. For this, you need to select a node and then Formulae in the scrolling menu at the bottom left of the GINsim window. Figure 3 illustrates this step.

   Alternatively, one can define logical parameters using the graphical menu, after selecting Parameters with the aforementioned scrolling menu. This implies the selection of a relevant set of interactions and clicking on the left arrow to add the corresponding parameter in the parameter list (see Note 3 for default values).

   The third option of this menu enables the user to enter textual annotations (bottom-right panel) or hyperlinks to relevant database entries (bottom-middle panel).

Note that the definition of adequate logical rules or parameters is necessary to ensure the desired effects of each interaction on the target nodes.

4. Change the graphical appearance of nodes and arcs of the graph at your convenience. For this, select the object, node or arc, and the Style tab. You can either change the default style or define your own styles for both graph nodes and arcs.

5. Selecting the Modelling Attributes tab, with no object selected in the main window, verify that the order of the variables is: p53, Mdm2cyt, Mdm2nuc, DNAdam. if this is not the case, modify the component order accordingly, using the arrows close to the component list at the bottom-left of the tab.

6. Save your model using the Save option of File menu.

4.3 Dynamical analysis

The simulation of a logical model defined as described above results in a state transition graph (which can be saved as a GINML file). The simulation settings (initial states, updating schemes and perturbations) can also be saved in the archive `zginml` (cf. Section 2.2).

Let us first consider the construction of the *asynchronous dynamics*:

1. Select Run Simulation in the Tools menu. This opens a panel enabling the construction of the dynamics.

The two Configure buttons and the two neighboring scrolling menus on the top enable the definition and the selection of model perturbations and reductions (see below). The bottom left window enables the definition and the recording of different parameter settings, which greatly facilitate the reproduction of results.

Regarding the construction strategy, a scrolling menu enables the choice between the generation of a State Transition Graph (STG), its compression into a Strongly Connected Components (SCC), or its further compression into a Hierarchical Transition Graph (HTG). Using another scrolling menu, the user can chose between synchronous or asynchronous updating, or define or select predefined priority classes (see Note 5).

Before clicking the Run button, verify that the default settings are as specified in Figure 4: asynchronous updating, no priority, no mutant selected, no initial state specified.

2. Display the state transition graph obtained (Figure 5). In fact, when the STG is small, it is automatically displayed. Note that the scrolling menus propose different options, including path search functions, etc.

   In the default level layout, the nodes with no incoming arc are placed at the top, whereas the nodes with no outgoing arc (stable states) are placed at the bottom. Stable states are further emphasised with different graphical attributes (here an ellipse). In this new window, you can re-arrange the nodes, change the graphical settings (clicking on the Style tap), and check

outgoing transitions by selecting a state, as shown in Figure 5.

The state 0100 (i.e. with Mdm2cyt ON and the other three components OFF) is selected, from which three unitary transitions are enabled by the rules: increase of Mdm2nuc from 0 to 1, decrease of Mdm2cyt from 1 to 0, and increase of p53 from 0 to 1. The selected state and its three successor states are shown in the bottom panel. It is possible to follow a transition path by clicking on a little rightward arrow on the left, which switches the selection to the corresponding state. When the selected state also connects to predecessors states, these are also shown, preceded by leftward arrows.

Note that we have obtained a unique stable state, (0010) (following the order defined above, this vector states that p53=0, Mdm2cyt=0, Mdm2nuc=1 and DNAdam=0), which corresponds to the rest state (low level of p53 and no DNA damage).

3. For comparison, let's build the STG using the synchronous updating strategy (selecting Synchronous with the scrolling menu in the Priority Class selection area shown in Figure 4. The resulting STG is shown in in Figure 6 Naturally, the stable state (0010) is preserved (bottom left) but we now obtain two cyclic attractors (bottom middle and right). Single and multiple transitions are denoted by solid and dotted arcs, respectively. For example, the selected state 0101 is leading to state 1011 through simultaneous changes of p53, Mdm2cyt and Mdm2nuc, as shown in the bottom panel (blue cells). Such behavior is not realistic from a biological point of view!

4. In the Subsection 3.3, we have seen that the STG can be compressed to better visualize its structure. A first compression consists in building the graph of strongly connected components (SCC). This can been done after the generation of the STG, by selecting the Construct SCC Graph function from the Tools scrolling menu.

   Alternatively, one can directly generate the SCC graph by selecting the corresponding option with the Construction Strategy scrolling menu. Figure 7 shows the resulting SCC graph, with all other simulation parameters left (in particular with asynchronous updating) identical to those shown in Figure 4.

   Now, we clearly see that there are two transient cyclic components, denoted by ct, followed by # and the number of states included in the cycle. The first cycle (ct#9) corresponds to large amplitude p53 oscillations in the presence of DNAdam (with p53 oscillating between the levels 0 and 2). The second cycle (ct#6) corresponds to smaller amplitude p53 oscillations in the absence of DNAdam (with p53 oscillating between the levels 1 and 2). In both cycles, Mdm2cyt and Mdm2nuc also oscillate.

5. As mentioned in the Subsection 3.4, we can further compress the dynamics using the Hierarchical Transition Graph (HTG) representation. This is achieved by selecting the corresponding option with the Construction Strategy scrolling menu. Figure 8 shows the resulting HTG, with all other simulation parameters left identical as shown in Figure 4. Although relatively modest in this case (from eleven nodes for the SCC graph to six

nodes for the HTG), this compression can be much more impressive in cases where we have limited oscillations and many alternative dynamical pathways (see e.g. [19, 21]).

Further analyses can be performed:

5. Using the Stable States option of the Tools menu of the main window, verify that the unique stable state of this model is indeed (0010) (see Figure 9); this calculation bypasses the construction of the STG, which is particularly useful for large models.

6. Define a mutant corresponding to an ectopic expression of DNAdam (see Figure 10). Such perturbations can be encoded before the computation of stable states or of a state transition graph. Verify that the resting stable state (0010) is not stable anymore for this perturbation. Can you find what is the new attractor for this perturbation?

7. Analyse Circuits from the Tools scrolling menu can be used to verify that the regulatory graph contains four circuits, among which three are functional (*i.e.* have a non-empty functionality contexts). For each functional circuit, one can verify its sign (depending on the rules) and its functionality context. As shown in the Figure 11, the positive circuit defined by the cross inhibitions between p53 and Mdm2nuc is functional when Mdm2cyt=DNAdam= 0. Indeed, the inhibition of Mdm2nuc by p53 is not functional in the presence of Mdm2cyt or DNAdam.

8. For large networks, it might be useful to reduce the model before per-

forming a simulation or other kind of dynamical analysis. Although our application is of moderate size, let's illustrate the use of GINsim model reduction functionality. Selecting the Reduce model option in the Tools scrolling menu launches the reduction interface. Select the component Mdm2cyt for reduction, as shown in Figure 12. Hitting the Run button generates a logical model encompassing only the three remaining components. Note that Mdm2nuc is now the target of a dual interaction from p53. The logical rule associated with Mdm2nuc has been modified to take into account the former indirect effect of p53 through Mdm2cyt. However, now that a reduction has been defined, you can select it when launching a simulation or computing stable states, without generating the reduced graph. Perform a complete asynchronous simulation and verify that the number of states is now lower by a factor of two (12 states instead of 24) compared to Figure 5.

5 Outlook

The logical formalism is particularly useful to model regulatory networks for which precise quantitative information is barely available, or yet to have a first glance of the dynamical properties of a complex model.

For this tutorial, we have considered a network of moderate size and we have followed the different steps enabling the delineation of a consistent logical model. Although relatively small, this model already displays relatively complex dynamics, including several transient oscillatory patterns and a stable state. It

further served as a reference to illustrate advanced functions, such as model reduction or regulatory circuit analysis.

As mentioned in the introduction, various logical models for different cellular processes have been proposed during the last decades. Many of these models are available in the repository included along with GINsim on the dedicated website (http://ginsim.org). The interested reader can thus download the model of his choice and play with it, for example to reproduce some of the results reported in the corresponding publication.

6 Notes

1. GINsim allows the user to export logical regulatory graphs as well as state transition graphs towards various formats, facilitating the use of other softwares:

    - SBML-qual, the qualitative extension of the popular model exchange format [20].
    - MaBoSS, a C++ software for simulating continuous/discrete time Markov processes, applied on a Boolean networks (`https://maboss.curie.fr/`).
    - BoolSim (`http://www.vital-it.ch/software/genYsis/`).
    - GNA, a software for the piecewise linear modelling of regulatory networks (`http://ibis.inrialpes.fr/article122.html`).
    - NuSMV, a symbolic model-checking tool (`http://nusmv.fbk.eu/`).

- Integrated Net Analyzer (INA) supporting the analysis of Place/ Transition Nets (Petri Nets) and Coloured Petri nets (`http://www2.informatik.hu-berlin.de/~starke/ina.html`).
- Snoopy, a tool to design and animate hierarchical graphs, among others Petri nets (`http://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Snoopy`.
- Graphviz, an open source graph visualization software offering main graph layout programs (`http://www.graphviz.org/`).
- BioLayout Express 3D, a tool for the visualization and analysis of biological networks (`http://www.biolayout.org`).
- Cytoscape, a popular open source software platform for visualizing molecular interaction networks (`http://www.cytoscape.org/`).
- Scalable Vector Graphics (SVG) format, an XML standard for describing two-dimensional graphics (`http://www.w3.org/Graphics/SVG/`).

2. In the non-Boolean case, a gene may have several distinct effects on another gene, depending on its activity level. In this case, only one arc is drawn, encompassing multiple interactions, each with its own threshold. An interaction is then active when the level of the source is above its threshold, and below that of the next interaction. Logical regulatory graphs may encompass multi-arcs, composed of different interactions having different effects. To each interaction, an interval specifying the range

of the source levels for which the interaction occurs must be specified. Intervals assigned to interactions with the same source and target must obviously be disjoint. Moreover, a sign should be specified (positive, negative or dual) for each regulatory effect.

3. For each node g, each combination of incoming interactions defines a logical parameter. This includes the case where no interaction acts on g (when every regulator value of g is below the first threshold of the arc pointing towards g). The logical parameter related to this case is called the *basal level* of g_i, *i.e.* the level of g_i in the absence of any specific activation. Considering that many parameters are usually null, zero is the default value in GINsim (*i.e.* a parameter that is not explicitly assigned takes value zero).

4. Transitions between states of the state transition graphs amount to the update of one (in the asynchronous case) or several (in the synchronous case) components. In any case, the update (increase or decrease) of a component is unitary (current value $+1$ or -1). Obviously, this remark applies only for multi-valued components (for which the maximal level is greater than 1).

5. Priority classes allow to refine the updating schemes applied to construct the state transition graphs [12]. GINsim users can group components into different classes and assign a priority rank to each of them. In case of concurrent updating transitions (*i.e.* calls for level changes for several

components in the same state), GINsim updates the gene(s) belonging to the class with the highest ranking. For each priority class, the user can further specify the desired updating assumption, which then determines the treatment of concurrent transition calls inside that class. When several classes have the same rank, concurrent transitions are treated under an asynchronous assumption (no priority).

6. A regulatory circuit is defined as a sequence of interactions forming a simple closed directed path. The sign of a circuit is given by the product of the signs of its interactions. Consequently, a circuit is positive if it has an even number of inhibitions, it is negative otherwise. R. Thomas proposed that positive circuits are necessary to generate multistationarity, whereas negative circuits are necessary to generate stable oscillations (see [27] and references therein). External regulators might prevent the functioning of a circuit imbedded in a more complex network. Reference [22] presents a method to determine the *functionality context* of a circuit in terms of constraints on the levels of its external regulator. A circuit functionality context can be interpreted as the part of the state space where the circuit is functional, *i.e.* generates the expected dynamical property [28].

References

1. Naldi A, Monteiro P, Mussel C, Kestler H, Thieffry D, Xenarios I, Saez-Rodriguez J, Helikar T, Chaouiya C: **Cooperative development of logical modelling standards and tools with CoLoMoTo**. *Bioinformatics* in revision.

2. Thomas R: **Regulatory networks seen as asynchronous automata: a logical description**. *Journal of Theoretical Biology* 1991, **153**:1–23.

3. Thomas R, Thieffry D, Kaufman M: **Dynamical behaviour of biological regulatory networks I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state**. *Bulletin of Mathematical Biology* 1995, **57**:247–76.

4. Thieffry D, Thomas R: **Dynamical Behaviour of Biological Regulatory Networks, II. Immunity Control in Bacteriophage Lambda.** *Bulletin of Mathematical Biology* 1995, **57**(2):277–97.

5. Mendoza L, Thieffry D, Alvarez-Builla E: **Dynamics of the genetic regulatory network for Arabidopsis thaliana flower morphogenesis**. *Journal of Theoretical Biology* 1998, **193**:307–19.

6. Sánchez L, Thieffry D: **A logical analysis of the Drosophila gap-gene system**. *Journal of Theoretical Biology* 2001, **211**(2):115–41.

7. Sánchez L, Thieffry D: **15. Segmenting the fly embryo: a logical analysis of the pair-rule cross- regulatory module**. *Journal of Theoretical Biology* 2003, **224**:517–37.

8. Sánchez L, Chaouiya C, Thieffry D: **Segmenting the fly embryo: a logical analysis of the segment polarity cross-regulatory module**. *International Journal of Developmental Biology* 2008, **52**(8):1059–75.

9. González A, Chaouiya C, Thieffry D: **Dynamical analysis of the regulatory network defining the dorsal-ventral boundary of the Drosophila wing imaginal disc**. *Genetics* 2006, **174**:1625–34.

10. González A, Chaouiya C, Thieffry D: **Logical modelling of the role of the Hh pathway in the patterning of the** *Drosophila* **wing disc**. *Bioinformatics* 2008, **24**:i234–40.

11. Fauré A, Vreede BMI, Sucena E, Chaouiya C: **A discrete model of Drosophila eggshell patterning reveals cell-autonomous and juxtacrine effects**. *PLoS Computational Biology* 2014, **10**(3):e1003527.

12. Fauré A, Naldi A, Chaouiya C, Thieffry D: **Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle**. *Bioinformatics* 2006, **22**(14):124–31.

13. Fauré A, Naldi A, Lopez F, Chaouiya C, Ciliberto A, Thieffry D: **Modular Logical Modelling of the Budding Yeast Cell Cycle**. *Molecular Biosystems* 2009, **5**:1787–96.

14. Naldi A, Carneiro J, Chaouiya C, Thieffry D: **Diversity and plasticity of Th cell types predicted from regulatory network modelling**. *PLoS Computational Biology* 2010, **6**:e1000912.

15. Abou-Jaoude W, Monteiro P, Naldi A, Grandclaudon M, Soumelis V, Chaouiya C, Thieffry D: **Model checking to assess T-helper cell plasticity**. *Frontiers in Bioengineering and Biotechnology* in press.

16. Coolen M, Thieffry D, Drivenes O, Becker T, Bally-Cuif L: **miR-9 controls the timing of neurogenesis through the direct inhibition of antagonistic factors**. *Developmental Cell* 2012, **22**:1052–64.

17. Sahin O, Frohlich H, Lobke C, Korf U, Burmester S, Majety M, Mattern J, Schupp I, Chaouiya C and Thieffry D, Poustka A, Wiemann S, Beissbarth T, D A: **Modeling ERBB receptor-regulated G1/S transition to find targets for de novo trastuzumab resistance**. *BMC Systems Biology* 2009, **3**:1.

18. Calzone L, Tournier L, Fourquet S, Thieffry D, Zhivotovsky B, Barillot E, Zinovyev A: **Mathematical Modelling of Cell-Fate Decision in Response to Death Receptor Engagement**. *PLoS Computational Biology* 2010, **6**:e1000702.

19. Grieco L, Calzone L, Bernard-Pierrot I, Radvanyi F, Kahn-Perlès B, Thieffry D: **Integrative Modelling of the Influence of MAPK Network on Cancer Cell Fate Decision**. *PLoS Computational Biology* 2013, **9**(10):21003286.

20. Chaouiya C, Bérenguier D, Keating SM, Naldi A, van Iersel MP, Rodriguez N, Dräger A, Büchel F, Cokelaer T, Kowal B, Wicks B, Gonçalves E, Dorier J, Page M, Monteiro PT, von Kamp A, Xenarios I, de Jong H, Hucka M, Klamt S, Thieffry D, Le Novère N, Saez-Rodriguez J, Helikar T: **SBML qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools**. *BMC Systems Biology* 2013, **7**:135.

21. Bérenguier D, Chaouiya C, Monteiro PT, Naldi A, Remy E, Thieffry D, Tichit L: **Dynamical modeling and analysis of large cellular regulatory networks**. *Chaos* 2013, **23**(2):025114.

22. Naldi A, Thieffry D, Chaouiya C: **Decision Diagrams for the Representation of Logical Models of Regulatory networks**. *Lecture Notes in Computer Sciences* 2007, **4695**:233–47.

23. Naldi A, Remy E, Thieffry D, Chaouiya C: **Dynamically consistent reduction of logical regulatory graphs**. *Theoretical Computer Science* 2011, **412**(21):2207–2218.

24. Ciliberto A, Novak B, Tyson JJ: **Steady states and oscillations in the p53/Mdm2 network**. *Cell Cycle* 2005, **4**(3):488–93.

25. Puszynski K, Hat B, Lipniacki T: **Oscillations and bistability in the stochastic model of p53 regulation**. *Journal of Theoretical Biology* 2008, **254**(2):452–65.

26. Abou-Jaoudé W, Ouattara D, Kaufman M: **From structure to dynamics: frequency tuning in the p53-Mdm2 network I. Logical approach**. *Journal of Theoretical Biology* 2009, **258**(4):561–77.

27. Thieffry D: **Dynamical roles of biological regulatory circuits.** *Briefings in Bioinformatics* 2007, **8**(4):220–5.

28. Comet JP, Noual M, Richard A, Aracena J, Calzone L, Demongeot J, Kaufman M, Naldi A, Snoussi EH, Thieffry D: **On circuit functionality in Boolean networks**. *Bulletin of Mathematical Biology* 2013, **75**(6):906–19.

Tables

Table 1: **Components and maximal levels for the p53-Mdm2 model.**

Components	Maximal levels
p53	2
Mdm2cyt	1
Mdm2nuc	1
DNAdam	1

Table 2: **Interactions and corresponding activity ranges for the p53-Mdm2 model**.

Sources	Targets	Source activity ranges
p53	Mdm2cyt	$[2, 2]$
	Mdm2nuc	$[1, 2]$
	DNAdam	$[1, 2]$
Mdm2cyt	Mdm2nuc	$[1, 1]$
Mdm2nuc	p53	$[1, 1]$
DNAdam	DNAdam	$[1, 1]$
	Mdm2nuc	$[1, 1]$

Table 3: **Logical rules for the p53-Mdm2 model.** This table lists the conditions enabling the activation of each component (up to level one in the case of a Boolean component, potentially up to higher levels for multilevel components, as for p53 here). These conditions are defined in terms of Boolean expression using the NOT, AND and OR Boolean operators (denoted by !, & and | in GINsim, respectively).

Components	Target levels	Boolean rules
p53	2	!Mdm2nuc
Mdm2cyt	1	p53 : 2
Mdm2nuc	1	Mdm2cyt | (!p53 & !DNAdam)
DNAdam	1	DNAdam & !p53

Figures

Figure 1: **GINsim main window displaying p53-Mdm2 logical regulatory graph.** The menu at the top displays five titles. These file scrolling menu provides access to classical file management options, to an option for merging the current graph with another one, as well as to facilities to export the regulatory towards various formats. The central area displays the regulatory graph, while the lower panel contains two tabs: the Modelling Attributes tab (selected here) and the Style tab, both in relation with to the selected graph component, here p53. The Edit button on the top is selected and emphasized with a green contour, enabling the edition of the attributes of the selected node, including its id and name, its maximal level (Max, here set to 2), and also the insertion of annotations in the form of free text (bottom right panel) or of links to relevant database entries (bottom middle panel).

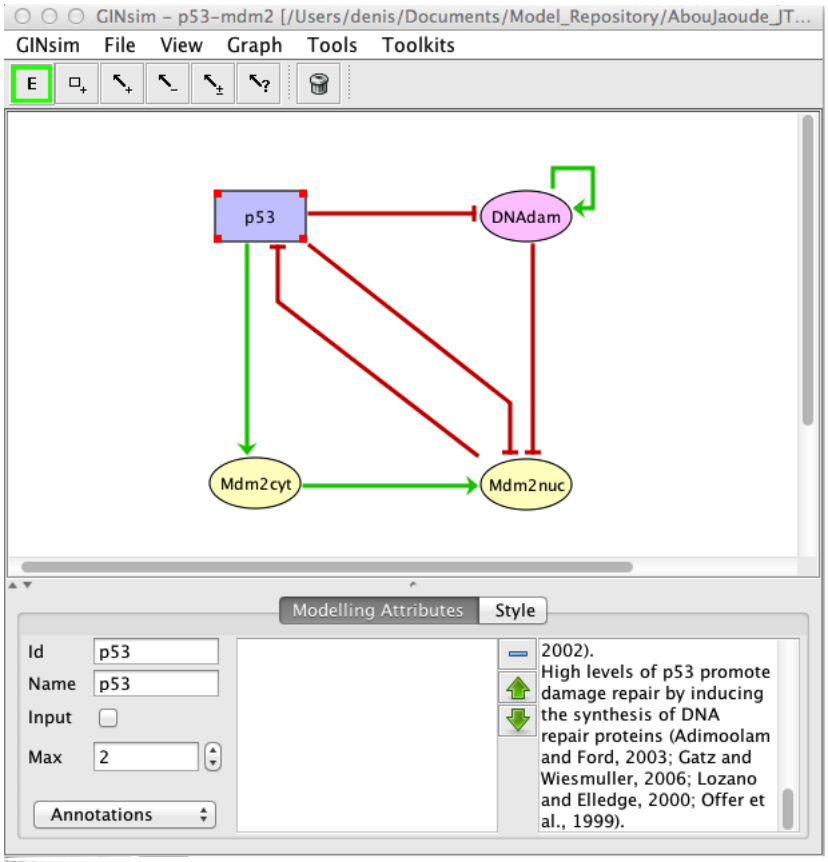

Figure 2: **Regulatory arcs management in GINsim**. To add an arc, the corresponding arc button must be pushed (push twice to add multiple arcs in a row), allowing the drawing of an arc between a source component and its target. Once an arc has been defined, it can be further edit by selecting it along with the Edition button. The sign of an interaction and the threshold(s) are defined with the Modelling Attributes tab, as shown here for the positive arc from p53 onto Mdm2cyt. Note that this arc has been defined as positive and associated with a threshold level 2, as shown in the bottom-left panel. (see also Note 2).

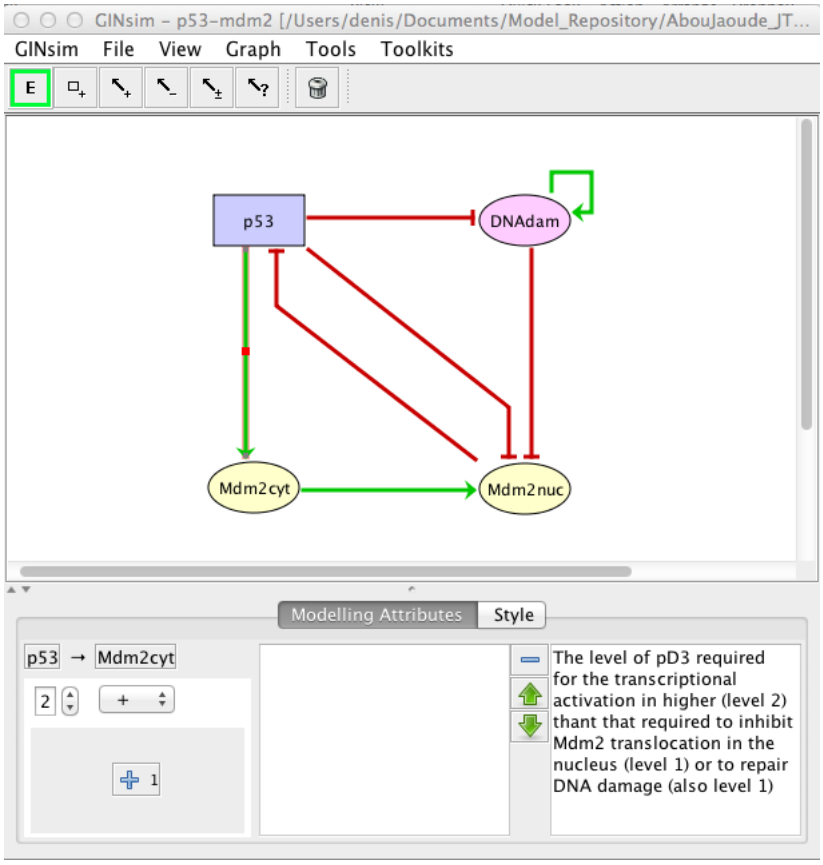

Figure 3: **Defining logical rules for regulatory components**. This screenshot shows the Modelling Attributes associated with the selected node DNAdam. The maximal level is set to 1. After selecting Formulae with the bottom-left scrolling menu, the user can define formulae by clicking on the little arrows in the main bottom panel. The target value (here set to 1 per default) can be changed in the case of a multilevel component. By clicking on the E button, one can enter a formula, using literals (these should be exactly match the IDs of components regulating the selected node, i.e. p53 or DNAdam in the present case) and the Boolean operators !, & and |, denoting NOT, AND and OR, respectively (following the usual priority rule; parenthesis can be used to define complex formulae). Note that several rows can be used in association with a single target values; these rows are then combined with OR operators. Here, the formula DNAdam & !p53 associated with the target value 1 implies that DNAdam will be maintained at a level 1 if already present, but only in the absence of p53.

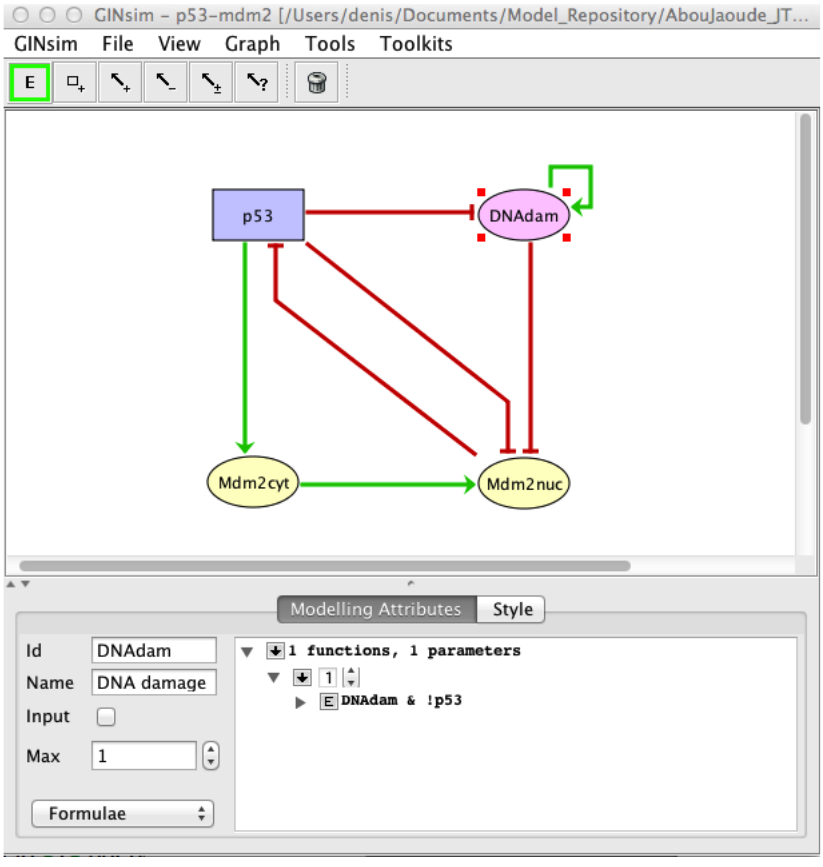

Figure 4: **Launching of the construction of a state transition graph.** This panel is obtained when selecting Run simulation from the Tools scrolling menu in GINsim main window. The default simulation settings are shown, i.e. the construction of State Transition Graph using the asynchronous updating, with no selected initial state (meaning that all states are considered in the simulation). Hitting the Run button will generate the corresponding State transition Graph in a new window (see Figure 5).

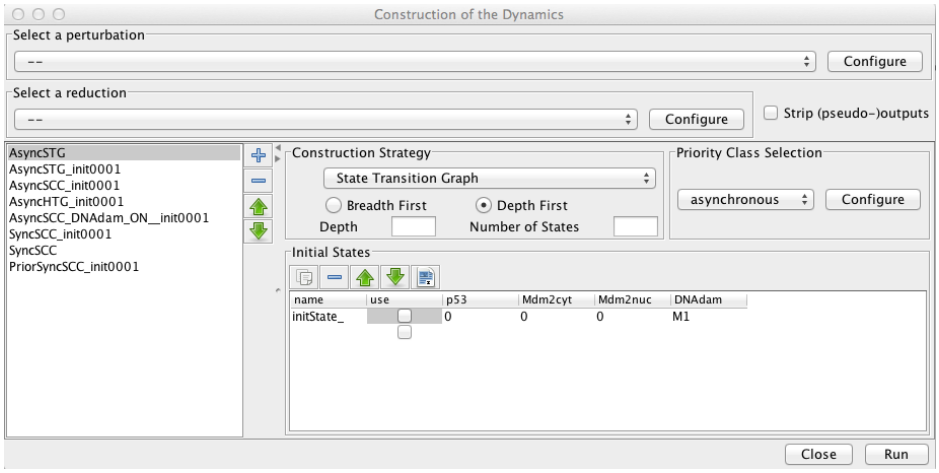

Figure 5: **Asynchronous state transition graph for the p53-Mdm2 model.** Asynchronous state transition graph generated with the simulation parameters shown in Figure 4, including the stable state (0010) laying at the bottom. The selected state (0100) is shown in the bottom panel, with its successors.

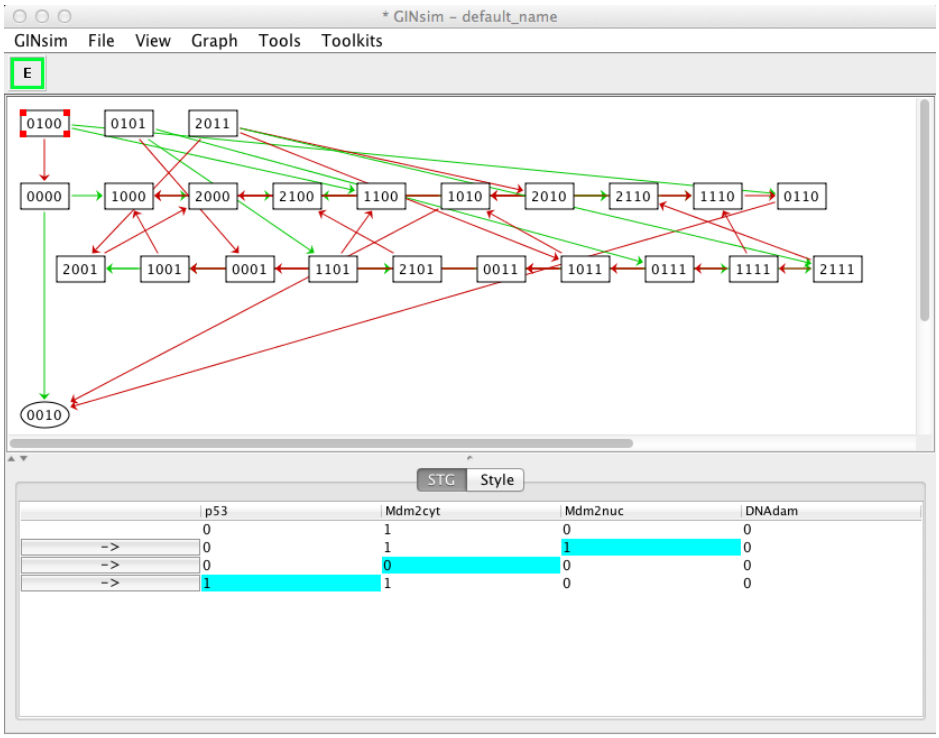

Figure 6: **Synchronous state transition graph for the p53-Mdm2 model.** The STG generated with the simulation parameters shown in Figure 4, but using the synchronous updating scheme. The STG is composed of three non connected subgraphs. On the left, we find back the resting stable state 0010, which can be reached from 14 other states. On the right, we see that the synchronous updating of our model further generates two two-states cyclic attractors, which can be reached from three or two other states respectively. Solid and dotted arrows denote single and multiple transitions, respectively.

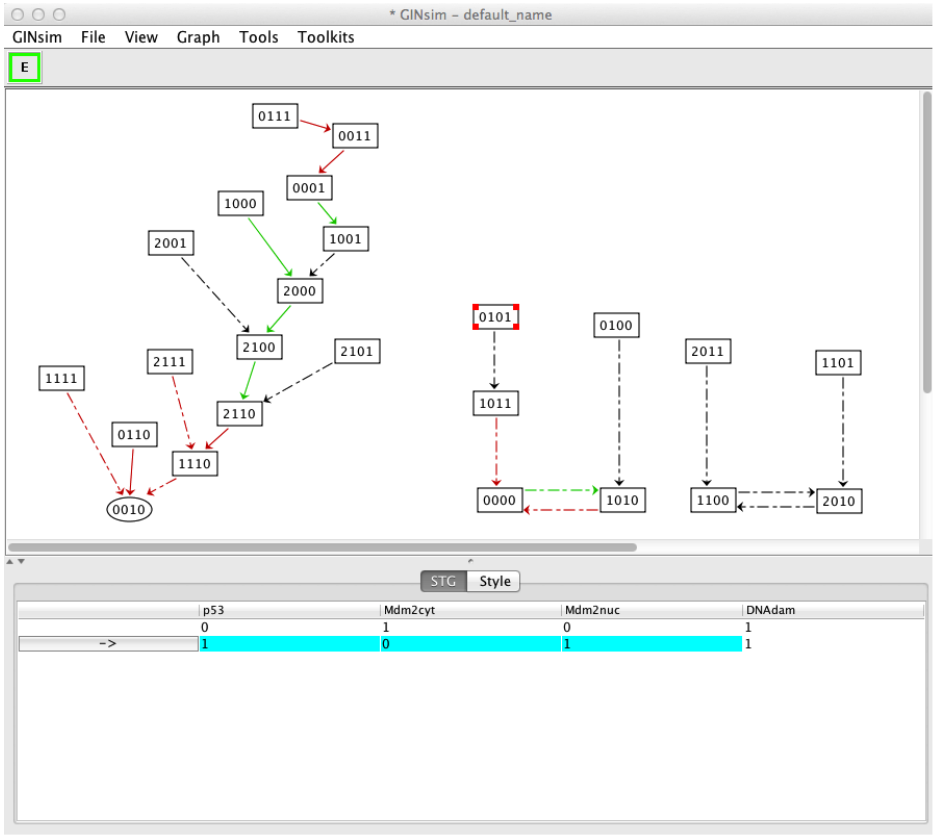

Figure 7: **Strongly connected component graph.** The graph of strongly connected components for the complete asynchronous dynamics of the p53-Mdm2 model is shown. It has been obtained by selecting the construction of Strongly Connected Component Graph in the corresponding scrolling menu when launching the simulation. The layout has been slightly manually improved. The three states shown at the top (in green) have only outgoing arcs (i- means that the corresponding states irreversibly traversed). The blue nodes correspond to non trivial strongly components (ct stands for cyclic transient component; the number 9 or 6 following the # denotes the number of states from the STG grouped in the corresponding SCC). None of these two non trivial SCC corresponds to attractors as one can escape them following one of the outdoing arcs. The SCC ct#6 is selected and its composition is shown in the bottom panel. The $*$ denotes all possible values for the corresponding components (here two for Mdm2nuc: 0 and 1). This SCC thus contains six states, all with DNAdam OFF. The unique attractor (a stable state) is shown in red at the bottom (ss- stands for stable state), which corresponds to the same resting stable state as shown in Figure 5.

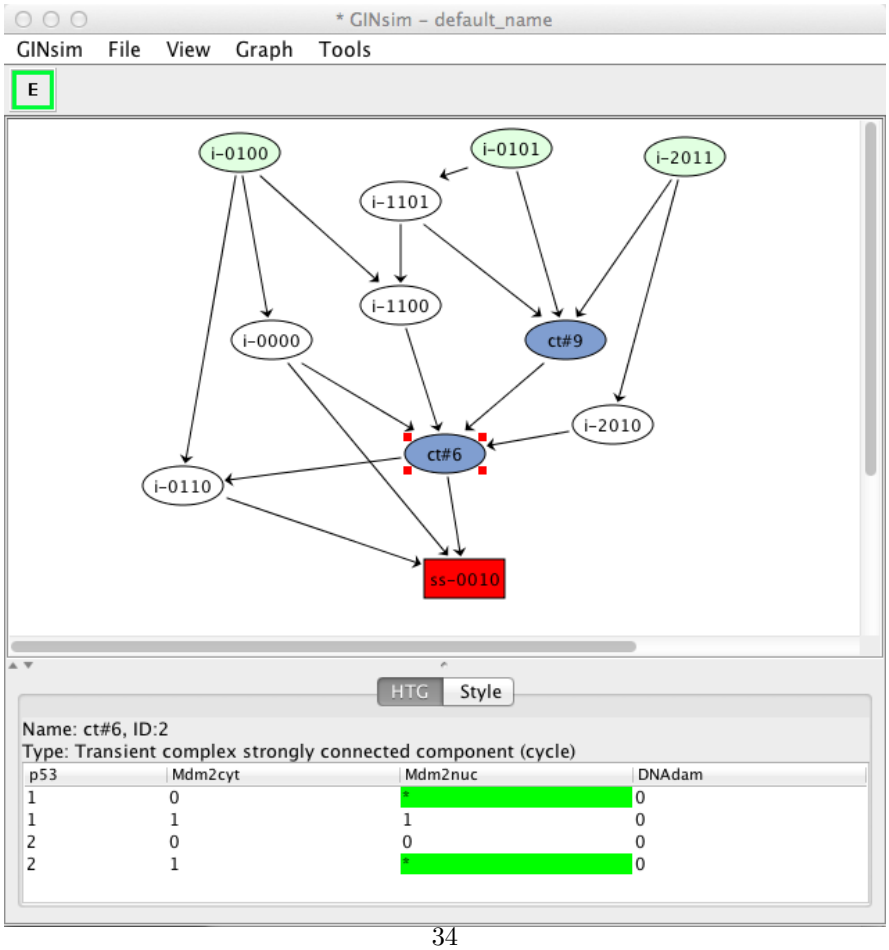

Figure 8: **Hierarchical transition graph.** The hierarchical transition graph for the complete asynchronous dynamics of the p53-Mdm2 model is shown. It has been obtained by selecting the construction of Hierarchical Transition Graph in the corresponding scrolling menu when launching the simulation. The layout has been slightly manually improved. The green node at the top has been selected and contains three states shown in the bottom panel. They can each lead to the cyclic component ct#9 or to a set of four transient state denoted by i#4. The blue nodes correspond to the two non trivial strongly components, and the the unique stable state is shown in red at the bottom, as in Figure 7.

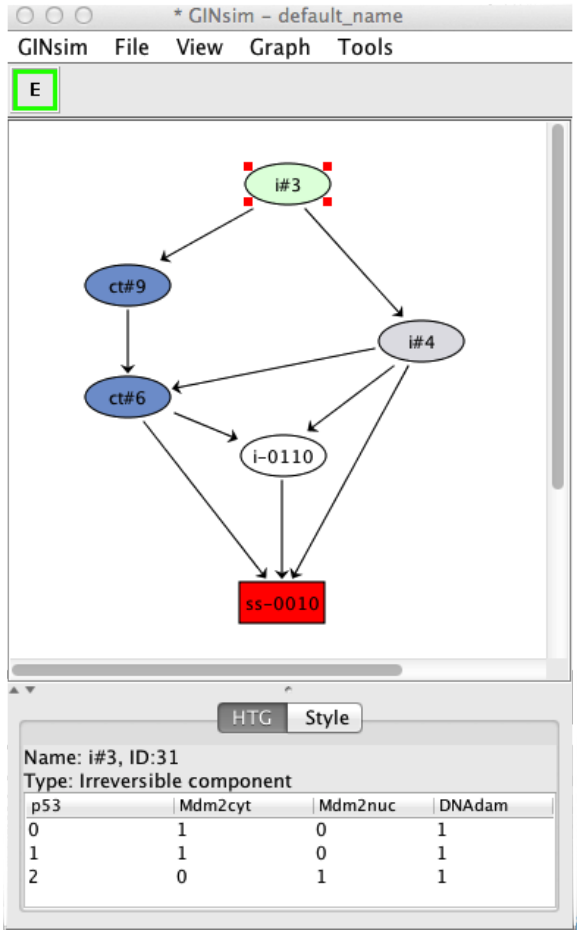

Figure 9: **Stable state determination**. This window pops up upon selection of Compute Stable States with the Tools scrolling menu. After hitting the Run button, GINsim returns all stable states using an efficient algorithm. In the wild type case, we obtain a unique stable state (0010) as shown (yellow and gray cells denote levels 0 and 1, respectively).

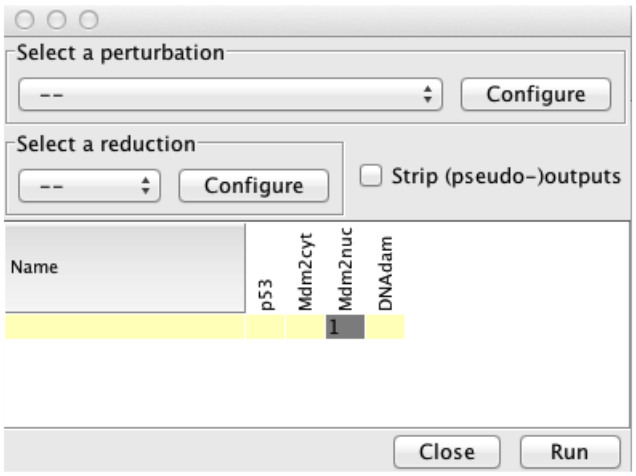

Figure 10: **Perturbation specification**. This window can be activated from the simulation launching window (Figure 4) and various other windows, including the Compute Stable States window. It enables the specification of model perturbations or mutants. The figure illustrates the specification of a simple blockade of the level of DNAdam to 1.

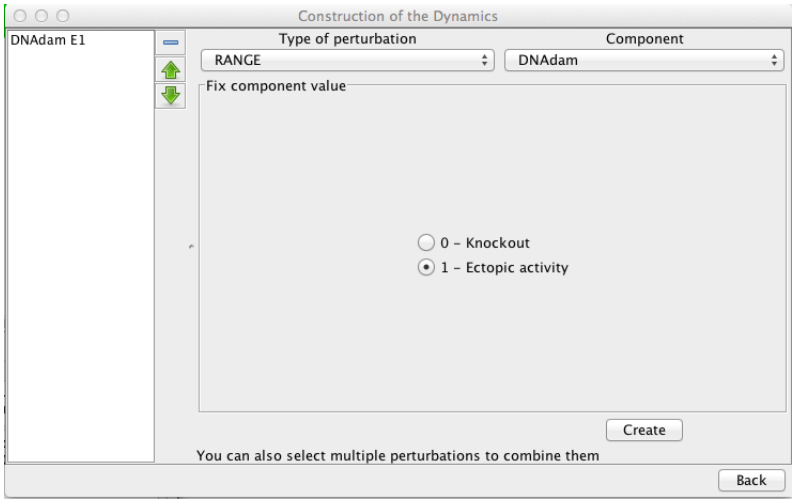

Figure 11: **Circuit analysis for the p53-Mdm2 logical model.** Among the four circuits found in the regulatory graph, three are functional: one is negative, while the other two are positive. The selected circuit (involving p53 and Mdm2nuc) is functional and positive when both Mdm2cyt and DNAdam are absent.

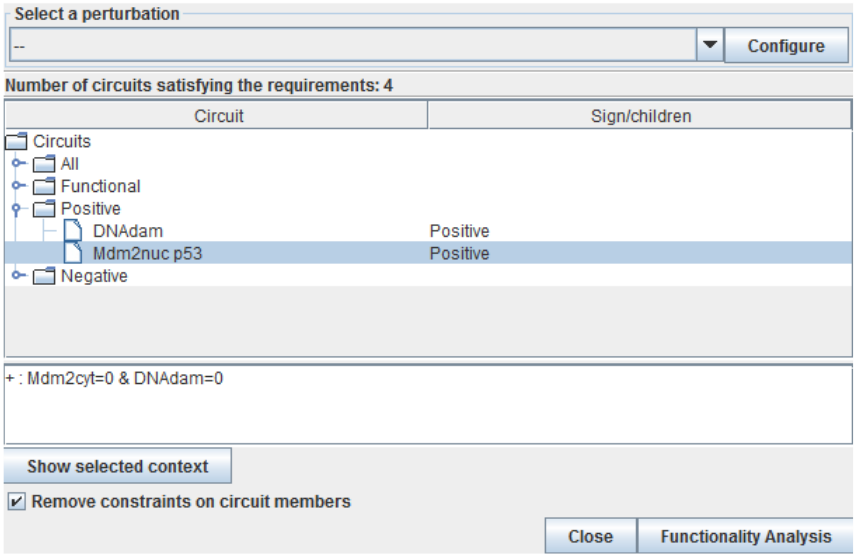

Figure 12: **Model reduction.** This window pops up following the selection of Reduce model from the Tools scrolling menu in the main GINsim window. Here, only Mdm2cyt has been selected for reduction. By hitting the Run Button, a reduced model is generated, provided that no self-regulated node is affected. Alternatively, one can close the window after the definition of one or several reduction(s) (using the + button on the left) and select a predefined reduction directly when performing simulations or other kinds of analyses.