Constructing and Analysing Computational Models of Cell Signalling with BMA

Benjamin A. Hall[1*] and Jasmin Fisher[2*]

[1]MRC Cancer Unit, University of Cambridge, Cambridge, UK; [2]UCL Cancer Institute, University College London, London, UK

*Correspondence should be addressed to Ben Hall bh418@mrc-cu.cam.ac.uk or Jasmin Fisher jasmin.fisher@ucl.ac.uk

Keywords: systems biology, formal verification, cell signalling, executable models

Abstract

BioModelAnalyzer (BMA) is an open-source graphical tool for the development of executable models of protein and gene networks within cells. Based upon the *Qualitative Networks* formalism, the user can rapidly construct large networks, either manually or by connecting motifs selected from a built-in library. After the appropriate functions for each variable are defined, the user has access to three analysis engines to test their model. In addition to standard simulation tools, the tool includes an interface to the stability testing algorithm and a graphical Linear Temporal Logic (LTL) editor and analysis tool. Alongside this, we have developed a novel ChatBot to aid users constructing LTL queries, and to explain the interface and run through tutorials. Here we present worked examples of model construction and testing through the interface. As an initial example, we discuss fate decisions in *D. discoidum* and cAMP signalling. We go on to describe the workflow leading to the construction of a published model of the germline of *C. elegans*. Finally, we demonstrate how to construct simple models from the built-in network motif library.

Introduction

Computational modelling represents a major strand of research in the life sciences. Over the past two decades, advances in modelling biological processes at the level of the cellular network have enabled novel insights into a wide range of processes including fundamental biological questions of organ development (Bonzanni et al., 2009; Hall, Piterman, Hajnal, & Fisher, 2015; Nusser-Stein et al., 2012), blood cell fate determination (Bonzanni et al., 2013; Chuang et al., 2015; Moignard et al., 2015; Silverbush et al., 2017), the dynamics of metabolic systems (Riedel et al., 2018) and ion channel networks (Shorthouse et al., 2018), amongst others. This progress has led directly to contributions in the biomedical sciences too, allowing for new discoveries with clinical relevance in diseases such as cancer.

Within this larger body of work, the field of executable models continues to expand (Fisher & Henzinger, 2007). In this field highly abstract models of genes, proteins, and cells are used to capture the key functional behaviour of biomolecules, rather than the specific chemical concentrations or physical parameters, following approaches used in the analysis of software and hardware systems. This class of models, which include Boolean and qualitative networks, finite state automata, and others, has several distinct advantages over traditional approaches. When considering complex networks, particularly in human cells, there may be multiple genes that have not been identified, and even for those genes that have been identified the physical parameters describing the protein function may not be available and require fitting. The fitting of multiple parameters raises specific issues of testability, but more generally incompleteness in models may undermine predictions on, for example, specific parameters if the predicted parameters result from the interactions of missing components. The simplified representation of gene activity within an executable model is less sensitive to this type of problem, as the coarse functional outcome should be captured more accurately than fine-grained details.

A further advantage of executable models is their amenability to analysis by methods from the field of formal verification (Claessen, Fisher, Ishtiaq, Piterman, & Wang, 2013; Cook, Fisher, Krepska, & Piterman, 2011). This field has been developed to analyse the behaviour of computer software and hardware systems, and uses model checking approaches to prove properties of models. In contrast to simulations, these analyses offer guarantees over all of state space. As such they can allow researchers to state that a specific behaviour never occurs, or always occurs. This is particularly valuable for systems that are highly robust (where proofs can show failures never occur) or suffer from rare failures (where the proofs should find the failure).

Finally, the level of abstraction used in executable models naturally maps to a large number of biological experiments. In many systems, cell decision-making processes lead to discrete outcomes, such as cell fate determination, or phenotype specification. Genetic screens, whether performed by random mutagenesis or more recently CRISPR screens, frequently generate this kind of experimental outcome with excellent sampling. In this context executable models, with discrete levels of activity and the ability to address all possible outcomes through formal verification, naturally maps to the experimental outcomes.

Constructing such models however remains difficult and frequently requires a dual-expertise reflecting both an understanding of the underlying biology and computing proficiency. Several tools have been developed to present an appropriate user interface to reduce the barriers to update, including Bio Model Analyzer (BMA) (Benque et al., 2012), GINsim (Naldi et al., 2009), CellCollective (Helikar et al., 2012), and BioLayOut3D (Wright, Angus, Enright, & Freeman, 2014). In many cases, these ease users into the process of model building through graphical user interfaces (GUI) but for some analyses require use of programming or command line interfaces (CLI), for example through the use of related tools such as NuSMV (Cimatti et al., 2002). The mixture of both GUI and CLI usage within a workflow is widespread and reflects the breadth of use cases and the need for some users to access advanced tools. It also reflects the reliance on formal languages for describing properties (i.e. formal logic or temporal logic) which do not necessarily map well into GUIs. In the case of the BMA tool, this has led us to investigating ChatBots as a more user-friendly alternative to either a rigid GUI, or a technically demanding CLI (Ahmed et al., 2017), available through Skype (https://bit.ly/3aAvrKZ).

In the following set of tutorials, we demonstrate how to use the BMA tool to construct and analyse a series of different biological systems. The BMA is a web-based tool www.biomodelanalyzer.org that presents a blank canvas for users (Figure 1) to construct and analyse models whilst avoiding use of the CLI as much as possible. The tutorials introduce key, distinguishing features of the approach and tool; developing models and specifications of varying complexity, using the graphical linear temporal logic (LTL) editor (Ahmed et al., 2017; Claessen et al., 2013), and the motif library (a collection of pre-built mini-networks, encoding key biological functions) (Paterson et al., 2018). BMA is open source (MIT license) and is run as a publicly available web service for all users. No model data is stored on the server, but for users who want to run calculations locally, for privacy or security reasons, an offline installation package is also made available.

Basic Protocol 1. Modelling the signalling network of *Dictyostelium discoidum*

The first example is intended as a gentle introduction to modelling a simple system. In this tutorial you should work through the steps and construct the models, using the suggested tests described. At the end you should understand the basic process of modelling, from developing a specification, constructing the model, to testing and refining the model. Recommended background references are papers by Kim and Laub (Kim, Heslop-Harrison, Postlethwaite, & Bates, 2007; Laub & Loomis, 1998). To complete this tutorial you will need a computer with a modern web browser such as Chrome, copies of the background papers to reference.

Dictyostelium discoidum is a slime mold with a complex life cycle, involving switching from a single cellular organism to a multicellular one and back. Here we look to model the process by which a single cell initiates the process of aggregation, in response to starvation. The model should have two developmental end states (attractors).

a. For the non-starvation state, the model should have a single fixpoint with low cAMP activity. ACA activity is low (as ACA activity is reported to trigger the starvation response).
b. For the starvation state, the model should have a single oscillation where cAMP rises and falls. ACA is active at some point of the cycle.

In this protocol we will build this model from the literature and show that it reproduces these behaviours. All work can be completed in the browser.

Steps and annotations

1. Open the web browser and navigate to http://biomodelanalyzer.org/, and click on “launch tool” to open the canvas (Figure 1). At the top of the screen are controls for editing and building the model. In the top left are the pan/select tool, and zoom tools. In the centre are variables to add to the canvas, and in the top right are undo/redo, settings, and information buttons. On the left hand side of the screen are icons for opening a new model, saving, and importing and exporting from the web browser. On the right hand side are different analysis tools; from top to bottom, stability testing, simulation, and LTL. At the bottom of the screen is the motif library.
2. Transcribe the Dictyostelium discoidum network from Figure 2A, based on the network from Kim et al. (Kim et al., 2007), making minimal edits to specify target functions.
   a. Drag and drop Variables onto the canvas, and add activations and inhibitions by selecting the appropriate tool in the top bar and dragging between variables (or a single variable for a self-interaction).
   b. Right click the activating variable for RegA, click edit, and write “1” for its target function (large, lower text box). This specifies that it has a basal level of activity.

   Note: non-overlapping edges are drawn as straight lines. This can be changed, along with different rendering options, in the settings menu found by clicking the gear icon in the top bar.

3. Test the model behaviour using the Linear Temporal Logic (LTL) interface (Figure 2B). Click the 3rd button from the top on the left hand side of the screen to open the analysis tab. To create formulas, click the lower button opening a new window. At the top of this window, user created operands are on the left, built-in operands (true, fixpoint, and self loop) are in the middle, and operators on the right.
   a. In this tool you create logical formula that describe specific simulations. When the formula is tested, the model is searched for simulations that match or fail to match the behaviour

described, returning one of three results; the formula is true for all simulations, the formula is never true, or the formula is sometimes true and sometimes false. For each result, examples can be viewed of each result to confirm that the formula has worked as expected. Only simulations that contain an attractor are returned (cycle or fixpoint), and the results are correct for an inputted length of simulation (defaulting to 10 steps). The formula itself is constructed from a mixture of built-in operators (e.g. and, or, next, upto) and operands (left to right: oscillation, fixpoint, and true), and user created operands.

b. To create a formula, drag operators and operands from the top of the screen into the middle.

c. To test the formula, click the test button that appears on the right side of the formula once it is complete. This will change the colour of the formula to indicate which result (cyan for true for all, pink for false for all, and striped pink/cyan for true sometimes, false sometimes).

d. To examine examples of simulations where the formula is either true or false, click the "open" button which replaces the test button after the analysis is complete, and click the cyan "example" button for a simulation where the formula is true, and the pink "example" button for a simulation where the formula is false.

Note: these elements can be created through a drag/drop interface, though advanced users have the option to right click on formulae to edit as text.

Note: You can create multiple formulas in the same window, and if you close and reopen the LTL tab, the formulas will be saved and retested if the model changes.

4. Create the following formulas
   a. "Eventually fixpoint" returns true for some simulations, and false for others, and the example of a matching simulation ends with low cAMP. This result shows that there exist cycles and fixpoints but does not show us how many there are. We need to use the example simulations to test this- open each example simulation, and right click on the last timepoint in the table to create an LTL state, noting down the names. Here we will assume that "A" is the fixpoint, and "B" is a state in the cycle.
   b. To show the fixpoint is unique, use the formula "(always (not A)) and (eventually fixpoint)". If the model is correct this should return always false as the only fixpoint is A.
   c. Similarly, to show the cycle is unique, use the formula "(always (not B)) and (eventually oscillation)". If the model is correct this should return always false as the only oscillation contains B.

   An alternative approach to testing these properties is to use the ChatBot interface. This takes requests in natural language (e.g. "find me a simulation that eventually reaches a fixpoint"). This will return the query rephrased into LTL, and if a model has been provided, the query will be tested and a link to view the model and query in the BMA provided.

5. If these tests are passed, the model of a single cell is correct. Name the model by clicking on the text in the left hand tab, and save the result by clicking the save icon.
6. Test that the model works when multiple cells share a single pool of cAMP (Figure 2C). Rename the model and save to prevent losing the single cell model. In the pan/select mode, shift click the cell to select it. Copy with ctrl-v, and paste the cell into other spaces with ctrl-v. Connect the external cAMP to all cells. Repeat the LTL testing to show that the system still oscillates. Save the model.

Basic Protocol 2. **Modelling the germline progression of *C. elegans***

The *C. elegans* germline (Hall et al., 2015) represents a model system for understanding the role of Notch (LIN-12 in *C. elegans*) and Ras (LET-60 in *C. elegans*) in cancers. In the niche, cells divide in a stem cell pool in response to Notch activation by delta, produced by the distal tip cell. The process of division drives cells away from the distal tip cell, causing Notch to become inactive and meiosis is initiated (pachytene stage). At a later time, apoptosis and further development (diplotene) is triggered by LET-60 activation, before subsequent LET-60 inactivation as the germ cells separate from the syncytium, enter the bended region of the organ, and enter the process of diakinesis. Finally, cells move down the proximal arm, and if exposed to sperm (triggering LET-60 activation), become fertilised and enter the uterus. As there are several developmental stages and different triggers acting on the system, the final model needs to reflect all of this complexity. Based on the fact that fate progression is invariant – wild-type cells always proceed through the stages in a single order – Basic Protocol 2 will demonstrate how to build a large model to match all of the desired behaviours by starting with a smaller model and building upon it.

To complete this tutorial you will need a computer with a modern web browser such as Chrome, copies of the background papers to reference, and the models supplied in supplementary information.

The initial specification is highly simplified. We assume that there are two genes (LIN-12 and LET-60, three developmental outcomes (mitosis, pachytene, and diplotene) each associated with a different "environment" (set of external stimuli). The environments are determined by variables representing delta present, a LET-60 activating ligand (here named sGF for "some growth factor"). We assume that the model is stable in each environment to ensure reliable fate progression. The initial specification is

- If the target function of delta is 1 and sGF is 0, the model is stable. In the fixpoint mitosis is 1, meiosis is 0, and differentiation is 0.
- If the target functions of delta and sGF are 0, the model is stable. In the fixpoint mitosis is 0, meiosis is 1, and differentiation is 0.
- If the target function of delta is 0 and sGF is 1, the model is stable. In the fixpoint, mitosis is 0, meiosis is 1, and differentiation is 1.

Steps and annotations

1. Construct the network from Figure 3A by adding variables and interactions as before ("Germline v1.json"). Note that only one environment is not part of the specification- where both LET-60 and LIN-12 are active.
2. Test the different environments by editing individual target functions, and performing stability testing. To do this, right click on each of the environent ligands (Delta or sGF), click edit in the context menu, and write a number (1 or 0) into the target function dialog box (bottom). Test the model by clicking the stability testing button (right hand side of screen, top button). You should find that the model matches the specification as described above. The model is stable under all conditions, and the value of mitosis, meiosis and differentiation are as noted.

   Note: testing the model under the unspecified condition (delta and sGF are both equal to one) reveals that the model is unstable

3. Expand on this model based on known interactions from the literature (Figure 3B, "Germline v2.xml"). Each pathway is expanded to include more genes, and with that their more complex relationships. The original behaviours however remain the same, with each environment

leading to a stable phenotype. The increased detail in the model allows us to extend our original specification to include other genes.

4. At this stage the model had been used in a hybrid simulation under different conditions. This is discussed in detail in Hall et al, but briefly, this model was used to describe cell state whilst a physical simulation was used to describe cell movements in the organ. Physical interactions arising from cell collision and division move cells along the organ and through different signalling environments, altering the state of the BMA model for each cell. One observation was that if LET-60 and LIN-12 were active at the same time, the fate order did not necessarily change if the cell had time to reach a fixpoint before the overlapping region was encountered. This is despite the model under those conditions being unstable, suggesting that only one fixpoint is reachable from the earlier.
5. At this point we extend the core specification, taking account of this observation and including diakinesis explicitly. The specification states that when LIN-12 and LET-60 are active the model is stable. When both are inactive however the model is unstable, and there are fixpoints representing diakinesis and pachytene states. Show this by performing stability testing, and click further testing to show that the fixpoints exist (they appear as a bifurcation).
6. Extend the model to follow this specification (Figure 3C, "Germline v3.xml"). Fate determination is modelled as the interactions between downstream gene targets of the LIN-12 and LET-60 pathways.

   Note: Through the console tool, fate progression can be tested explicitly using the "Pathfinder" algorithm. Run the command

   ```
   BioCheckConsole.exe -engine PATH –model model1.json -model2 model2.json
   -state initial_state.csv -state2 target_state.csv
   ```

   model1.json and model2.json are exported BMA models representing two different conditions, whilst initial_state.csv and target_state.csv are csv files describing the final fixpoint for each state.

7. Similar to the above, extend the model to include the "fertilisation" state activated by LET-60 mechanisms to the tension between diakinesis and pachytene states. Again, the finding that fate progression invariance is achieved through "islands of stability"- that is, fixpoints that can only reach one fixpoint after a change in conditions.
8. Finally, based on observations in the hybrid model, add a cell cycle "clock" variable (triggered by time measured in the hybrid simulation) to trigger diplotene (Figure 3D, "Germline v5.2.json").

Basic protocol 3. Constructing a model of the cell cycle using motifs

"Motifs" are small networks that perform a specific function in the cell. Tyson, Chen, and Novak (Tyson, Chen, & Novak, 2003) characterised these models for continuous modelling in detail. Subsequently, we established that executable models were similarly expressive, and integrated a set of motifs into the tool itself (Paterson et al., 2018). Such motifs aid the construction of complex networks by providing a set of fundamental cellular patterns that can be directly integrated into the network. Moreover, the tool allows users to create their own motifs based on their systems requirements.

In this tutorial we will revisit the cell cycle model from Paterson et al., demonstrating how a simplified version can be constructed out of available motifs. Here you should follow the instructions to build the model using the steps described. We will go on to show how one motif is not behaving as expected in the model and show how users can create their own motifs and reuse them. The basic cell-cycle model from Tyson (Tyson et al., 2003) can be considered as being constructed around three motifs; two mutual inhibition motifs and a negative feedback leading to an oscillation. These are coordinated through interaction with cyclin dependent kinases, whose activity rises in response to activation of the mutual inhibition motifs until finally causing the cell to commit to division.

To complete this tutorial you will need a computer with a modern web browser such as Chrome, copies of the background papers to reference, and the models supplied in supplementary information.

Steps and annotations

1. Open the motif editor by clicking on the pull-out at the bottom of the screen. Individual motifs can then be dragged and dropped onto the canvas. Drag the mutual inhibition motif to the canvas twice, and the oscillation once as shown (Figure 4A).
2. Add an "extracellular" protein to the centre of the model and rename this CDK (right click, select edit, edit name). Move the receptors on the cells to neighbour this (drag and drop after clicking to "pan" icon, top left, to start "pan/select" mode), rename the cells and variables, and connect with activations and inhibitions by dragging interactions as shown (Figure 4B). In the model, commitment to S and M are both activated and activate CDK, whilst commitment to division initiates a process that reduces CDK activity and its downstream effectors. To represent this commitment a self-interaction is included for the division variable. Additionally, the division process is reset once CDK activity drops to zero.
3. Edit the target functions and ranges for the variables. A key feature of the model is that rising CDK activity drives the movement of the cell through different stages of the cell cycle. To reflect this, we increase the range of CDK activity to 0-14, with a mind that a level greater than one initiates progression through G1, 5 or more triggers G2, and a level of 10 initiates commitment to division. Set the range of Division to 0-1, to reflect the switch and commitment to the division process. We also want the process of division to inhibit the G1 and G2 modules, whilst division is considered "complete" once CDK levels reach 1. Finally, CDK has a constant basal activity due to growth signals. To achieve this, the following functions should be used for variables:
    a. G1.I = var(CDK)*4-var(Division)
    b. G2.I = var(CDK)*2-var(Division)
    c. I_a = var(CDK)/4
    d. Division = ceil(min(var(CDK),(max(var(A_a),var(Division)))))
    e. CDK = 4+var(M)/3+var(S)/3-var(Division)

4. Finally, test the model by simulation. Run a simulation of 50 steps and the model should give a cycle (Figure 4C). This reproduces the key features of the cycle- S precedes M, which precedes division, whilst the level of CDK activation rises to a plateau, rises again in G2, and then falls following commitment to division.
5. There is however one feature that is incorrect. The first cycle is correct, but the second cycle (which represents the attractor) does not have a gap between S and M. Examination of the model reveals that the mutual inhibition loop does not work as expected and that this breaks the cycle. To demonstrate how to use your own motifs, open the model MI5.
6. Select the whole motif by either left clicking the cell, or right-clicking and dragging to select all the variables. Then, right click the selected variables and choose "Selection -> Create Motif".
7. Open the model and drag the new motif from the pull-out as before. It is automatically named "Clipboard model", as is a generic description, both of which can be edited by the user by clicking and editing the text in place. Replace the motifs in the network and reuse the target functions as above and re-simulate (Figure 4D).

Guidelines for Understanding Results

The protocols here stand as an introduction to the construction and analysis of executable network models of biological systems. We have focused here on the construction of gene regulatory networks, but this approach can be applied to a wide range of biological systems. These have included such diverse systems as metabolic networks (Riedel et al., 2018) and ion flow across membranes (Shorthouse et al., 2018). This work and others like it illustrate that the methods are not tied to a specific class of models, and can be broadly applied to different problems.

Commentary

Background

In each protocol we have constructed a model based on the knowledge of interacting gene networks and the characterised behaviours of the cell. We start from simple motifs that reproduce the basic specification, and build up an increasingly detailed picture of the genes and cell behaviours. The resulting models demonstrate that the proposed mechanism is sound, and can be used to describe the observation. It further predicts new behaviours- the activation and inhibition of genes not included in the original network, and conditions that are untested and not part of the original specification. These predictions can be validated experimentally, and once confirmed added back to the model for future work.

A central theme of each protocol is the construction of models from smaller, well understood components. Directly transcribing complex networks from the literature is possible, but can quickly lead to the development of models that cannot be debugged and refined. The development of core components that interact enables reliable and understandable model building and furthermore can help downstream users of the models.

Critical Parameters

The correctness of the model is most clearly dependent on the network of genes involved. The orders of dependencies will determine how mutations combine with one another to change cell properties. More subtly, the target functions must describe thresholds for activity, and the precise relationship between genes. A pair of activating interactions may combine redundantly to induce activity, or the downstream gene may require both active inputs to trigger. Identifying the appropriate functions to link genes will play a major part of the process of model building.

More subtly, large networks built directly from the literature may obscure more simple underlying mechanisms of cell control. A model of ~50 genes (for example the model of leukaemia from (Chuang et al., 2015), https://bit.ly/30UJBSo) could be constructed directly from the literature, and may be partially correct with refinement. However, if a complex model cannot be refined, the size of the network may make it impossible to identify which features are causing problems. Starting from a simple mechanism based on key genes, and building in the detail after establishing that the small network is correct can avoid this.

Trouble Shooting

Broadly speaking, two classes of analysis error are more common than others. Models may be too large and lead to timeout of the analysis used. If this is the case it may be necessary to simplify the model by removing genes, or reconsider whether the specification can be tested with simulation or stability analysis, both of which are faster than LTL.

Secondly, target functions may be too complex for the underlying tools to compute. This will give an unknown response when using analysis. This can be avoided by working to simplify the functions, reducing the number of operators used. If this is unsuccessful, any bugs/issues can be reported through the tool's github page (https://github.com/hallba/BioModelAnalyzer/issues).

Acknowledgements

BAH is supported by the Royal Society through a URF (UF130039) and through an MRC Grant in Aid to the MRC Cancer Unit. J.F. is a member of the Mark Foundation Institute for Integrated Cancer Medicine at the University of Cambridge.

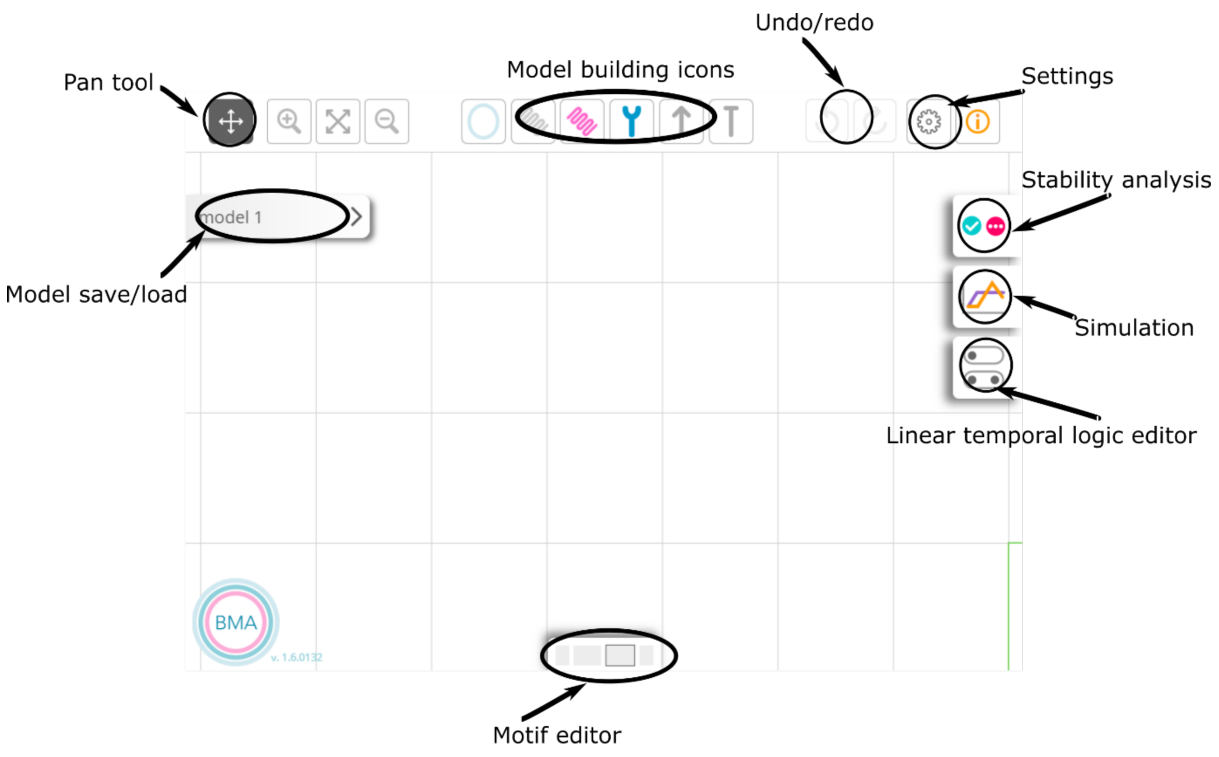

Figure 1. The BioModelAnalyzer user interface. Users interact with the model menu, motif editor, and analysis tools by clicking to "pull out" an interface that sits over the main canvas. Networks can be dragged and dropped or click-to-drop using the model building icons, whilst users can pan and zoom around the canvas through the pan tool.

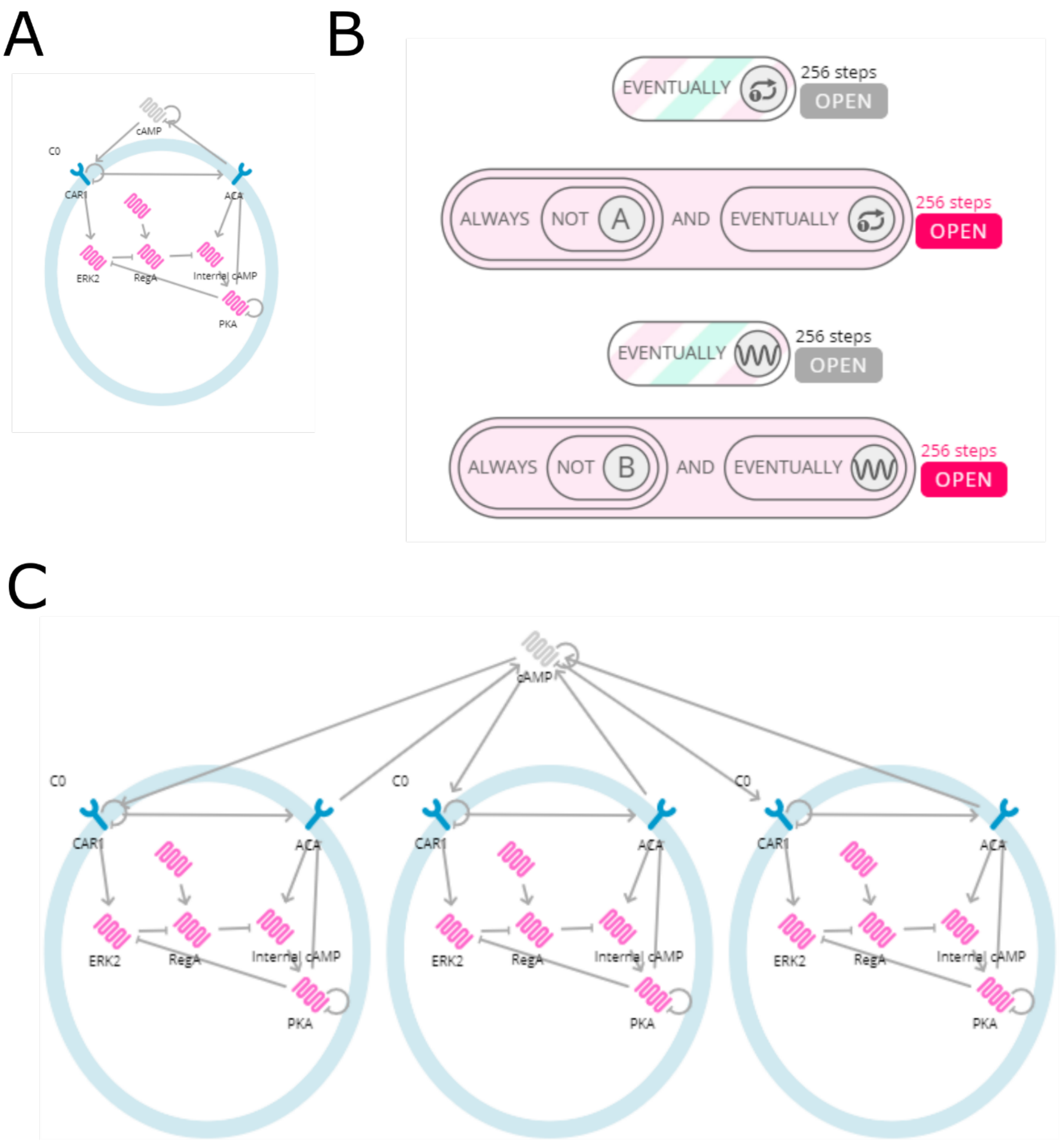

Figure 2. Modelling Dictyostelium discoidum cAMP signalling network. (A) The initial model transcribed from Laub et al. (B) LTL formula that show that there exists a single fixpoint attractor and a single cycle. “Eventually selfloop” shows that some simulations end in a cycle whilst others in a fixpoint (indicated by pink/blue stripes). Using the examples of the fixpoint (“A”) and cycle (“B”) we show that these are unique (solid pink). An alternative query “Eventually (A or B)” would true for all simulations (and appear solid blue). (C) A model with more than one cell and a shared external cAMP pool shows synchronous oscillations.

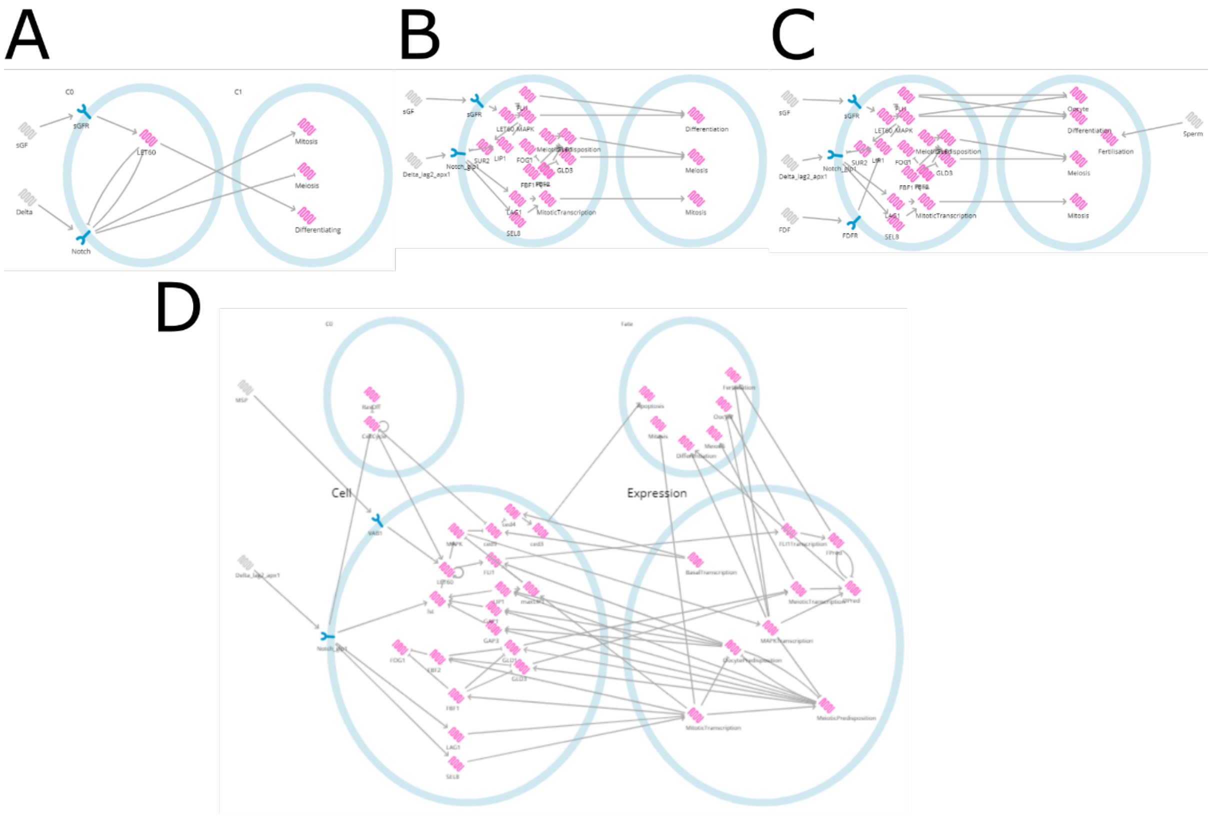

Figure 3. Evolving models of the *C. elegans* germline. A-D show sequentially developed models of increasing complexity. (A) A small model is constructed to fit a simple specification, and later expanded based on known genes from the literature (B). (C) an extended specification requires an expansion of cell phenotypes and interactions, whilst the final model (D) includes details of the cell cycle.

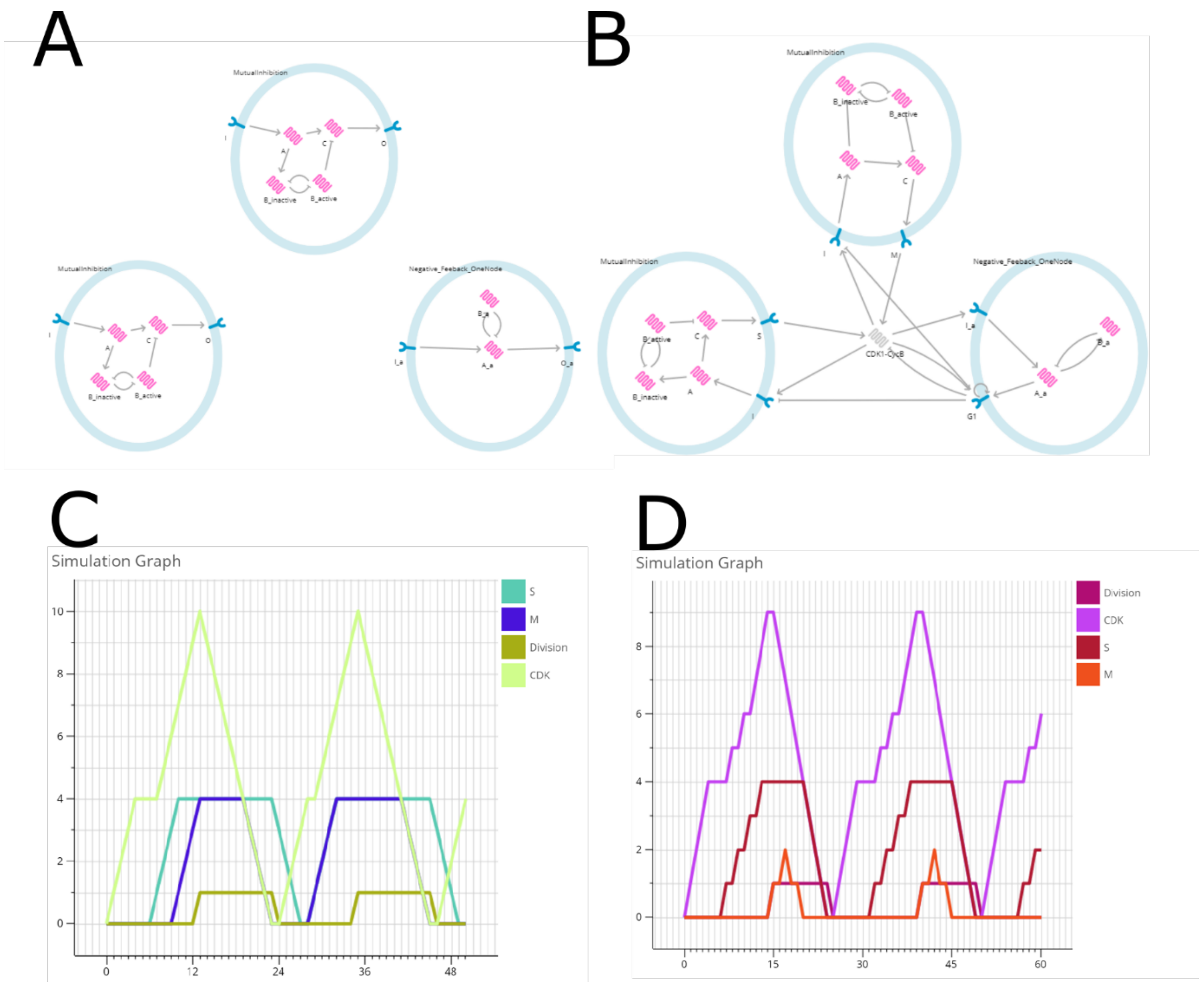

Figure 4. Constructing a cell cycle model from motifs. (A) Initially, subsystems are dragged from the motif editor. (B) Adding relevant interactions between specific genes allows the subdomains to coordinate with one another. (C) Initial simulations show that the model mostly reproduces expected cycles, whilst a refined model (D) does this more accurately.

Ahmed, Z., Benque, D., Berezin, S., Dahl, A. C. E., Fisher, J., Hall, B. A., . . . Skoblov, N. (2017, 2017//). *Bringing LTL Model Checking to Biologists.* Paper presented at the Verification, Model Checking, and Abstract Interpretation, Cham.

Benque, D., Bourton, S., Cockerton, C., Cook, B., Fisher, J., Ishtiaq, S., . . . Vardi, M. Y. (2012, 2012//). *Bma: Visual Tool for Modeling and Analyzing Biological Networks.* Paper presented at the Computer Aided Verification, Berlin, Heidelberg.

Bonzanni, N., Garg, A., Feenstra, K. A., Schutte, J., Kinston, S., Miranda-Saavedra, D., . . . Gottgens, B. (2013). Hard-wired heterogeneity in blood stem cells revealed using a dynamic regulatory network model. *Bioinformatics, 29*(13), i80-88. doi:10.1093/bioinformatics/btt243

Bonzanni, N., Krepska, E., Feenstra, K. A., Fokkink, W., Kielmann, T., Bal, H., & Heringa, J. (2009). Executing multicellular differentiation: quantitative predictive modelling of C.elegans vulval development. *Bioinformatics, 25*(16), 2049-2056. doi:10.1093/bioinformatics/btp355

Chuang, R., Hall, B. A., Benque, D., Cook, B., Ishtiaq, S., Piterman, N., . . . Fisher, J. (2015). Drug Target Optimization in Chronic Myeloid Leukemia Using Innovative Computational Platform. *Scientific Reports, 5*, 8190. doi:10.1038/srep08190

https://www.nature.com/articles/srep08190#supplementary-information

Cimatti, A., Clarke, E. M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., . . . Tacchella, A. (2002). *NuSMV 2: An OpenSource Tool for Symbolic Model Checking*. Paper presented at the Proceedings of the 14th International Conference on Computer Aided Verification.

Claessen, K., Fisher, J., Ishtiaq, S., Piterman, N., & Wang, Q. (2013, 2013//). *Model-Checking Signal Transduction Networks through Decreasing Reachability Sets.* Paper presented at the Computer Aided Verification, Berlin, Heidelberg.

Cook, B., Fisher, J., Krepska, E., & Piterman, N. (2011, 2011//). *Proving Stabilization of Biological Systems.* Paper presented at the Verification, Model Checking, and Abstract Interpretation, Berlin, Heidelberg.

Fisher, J., & Henzinger, T. A. (2007). Executable cell biology. *Nat Biotechnol, 25*(11), 1239-1249. doi:10.1038/nbt1356

Hall, B. A., Piterman, N., Hajnal, A., & Fisher, J. (2015). Emergent stem cell homeostasis in the C. elegans germline is revealed by hybrid modeling. *Biophys J, 109*(2), 428-438. doi:10.1016/j.bpj.2015.06.007

Helikar, T., Kowal, B., McClenathan, S., Bruckner, M., Rowley, T., Madrahimov, A., . . . Rogers, J. A. (2012). The Cell Collective: Toward an open and collaborative approach to systems biology. *BMC Systems Biology, 6*(1), 96. doi:10.1186/1752-0509-6-96

Kim, J., Heslop-Harrison, P., Postlethwaite, I., & Bates, D. G. (2007). Stochastic noise and synchronisation during dictyostelium aggregation make cAMP oscillations robust. *PLoS Comput Biol, 3*(11), e218. doi:10.1371/journal.pcbi.0030218

Laub, M. T., & Loomis, W. F. (1998). A molecular network that produces spontaneous oscillations in excitable cells of Dictyostelium. *Mol Biol Cell, 9*(12), 3521-3532. doi:10.1091/mbc.9.12.3521

Moignard, V., Woodhouse, S., Haghverdi, L., Lilly, A. J., Tanaka, Y., Wilkinson, A. C., . . . Gottgens, B. (2015). Decoding the regulatory network of early blood development from single-cell gene expression measurements. *Nat Biotechnol, 33*(3), 269-276. doi:10.1038/nbt.3154

Naldi, A., Berenguier, D., Fauré, A., Lopez, F., Thieffry, D., & Chaouiya, C. (2009). Logical modelling of regulatory networks with GINsim 2.3. *Biosystems, 97*(2), 134-139. doi:https://doi.org/10.1016/j.biosystems.2009.04.008

Nusser-Stein, S., Beyer, A., Rimann, I., Adamczyk, M., Piterman, N., Hajnal, A., & Fisher, J. (2012). Cell-cycle regulation of NOTCH signaling during C. elegans vulval development. *Mol Syst Biol, 8*, 618. doi:10.1038/msb.2012.51

Paterson, Y. Z., Shorthouse, D., Pleijzier, M. W., Piterman, N., Bendtsen, C., Hall, B. A., & Fisher, J. (2018). A toolbox for discrete modelling of cell signalling dynamics. *Integrative Biology, 10*(6), 370-382. doi:10.1039/c8ib00026c

Riedel, A., Swietlik, J., Shorthouse, D., Haas, L., Young, T., Costa, A. S. H., . . . Shields, J. (2018). Tumor pre-conditioning of draining lymph node stroma by lactic acid. *bioRxiv*, 442137. doi:10.1101/442137

Shorthouse, D., Riedel, A., Kerr, E., Pedro, L., Bihary, D., Samarajiwa, S., . . . Hall, B. A. (2018). Exploring the role of stromal osmoregulation in cancer and disease using executable modelling. *Nat Commun, 9*(1), 3011. doi:10.1038/s41467-018-05414-y

Silverbush, D., Grosskurth, S., Wang, D., Powell, F., Gottgens, B., Dry, J., & Fisher, J. (2017). Cell-Specific Computational Modeling of the PIM Pathway in Acute Myeloid Leukemia. *Cancer Research, 77*(4), 827-838. doi:10.1158/0008-5472.can-16-1578

Tyson, J. J., Chen, K. C., & Novak, B. (2003). Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Curr Opin Cell Biol, 15*(2), 221-231. doi:10.1016/s0955-0674(03)00017-6

Wright, D. W., Angus, T., Enright, A. J., & Freeman, T. C. (2014). Visualisation of BioPAX Networks using BioLayout Express (3D). *F1000Research, 3*, 246-246. doi:10.12688/f1000research.5499.1