Finding instability in biological models

Byron Cook[1,2], Jasmin Fisher[1,3], Benjamin A Hall[1], Samin Ishtiaq[1], Garvit Juniwal[4] and Nir Piterman[5]

[1] Microsoft Research
[2] University College London
[3] University of Cambridge
[4] University of California, Berkeley
[5] University of Leicester

Abstract. The stability of biological models is an important test for establishing their soundness and accuracy. Stability in biological systems represents the ability of a robust system to always return to homeostasis. In recent work, modular approaches for proving stability have been found to be swift and scalable. If stability is however not proved, the currently available techniques apply an exhaustive search through the unstable state space to find loops. This search is frequently prohibitively computationally expensive, limiting its usefulness. Here we present a new modular approach eliminating the need for an exhaustive search for loops. Using models of biological systems we show that the technique finds loops significantly faster than brute force approaches. Furthermore, for a subset of stable systems which are resistant to modular proofs, we observe a speed up of up to 3 orders of magnitude as the exhaustive searches for loops which cause instability are avoided. With our new procedure we are able to prove instability and stability in a number of realistic biological models, including adaptation in bacterial chemotaxis, the lambda phage lysogeny/lysis switch, voltage gated channel opening and cAMP oscillations in the slime mold *Dictyostelium discoideum*. This new approach will support the development of new clinically relevant tools for industrial biomedicine.

Keywords: stability, instability, verification, biology

1 Introduction

Traditional computer science approaches are playing an increasingly important role in the modeling and analysis of biological systems. Formal verification approaches for biological signaling systems have been successfully applied in wide range of different organisms and phenomena [1–3]. In different systems, proofs of both reachability [4, 5], and stability [6] can give powerful insights into the mechanisms of cell differentiation and homeostasis. Stability specifically offers a valuable tool when considering systems which can be reliably considered as being at equilibrium or homeostatic. We consider stability here in terms of a guarantee that the system always eventually moves towards a single self-loop state, regardless of the initial state of the model. In the context of biological

systems, instability can therefore indicate a developmental switch (e.g. bifurcation) or oscillation (e.g. cycles with lengths greater than 1). In contrast stability demonstrates that the system is at a robust equilibrium, as any temporary perturbation will eventually converge to the equilibrium state.

The development of formal models of biological systems further offers a new platform for discoveries in the life sciences and medical research. By translating the diagrammatic models typically generated in experimental disciplines into forms which can be explored using verification techniques we can highlight inadequacies in the model and propose new testable hypotheses. In contrast to models based on precise reproduction of physical or chemical properties of a given biological phenomenum, *executable* models avoid a reliance on highly accurate quantitative data from experimental studies. For questions such as "could a drug targeted to this protein ever kill the cell?" or "does the model accurately represent a robust equilibrium?", the relative independence of formal models from detailed physical constants is a strength of the technique over traditional physicochemical simulation. Furthermore, this degree of abstraction more closely mimics the qualitative data generated by genetic screens. Boolean and qualitative networks specifically, have been successfully used in the study of diverse systems. Initially applied to the study of gene regulation [7] these formalisms have been applied to blood cell differentiation, skin homeostasis and cancer development [8–11]. The ability to analyze models with these approaches has great relevance to clinicians and the biomedical industries. Furthermore, Boolean networks have been used successfully to systematically model drug interactions in tumorigenesis [12], in order to rationally identify new drug targets. This ability to validate drug targets *in silico* offers the potential to avoid costly failures at late stage clinical trials, and as such new model checking techniques has great relevance to clinicans and the biomedical industries.

Existing tools suitable for the analysis of biological qualatitive networks, such as GinSim [13] and NuSMV [14], explore stability through the use of efficient representations of the state transitions as binary decision diagrams and multi-valued decision diagrams [15], coupled with simulation. The reliance on exhaustive simulation however limits the size of the models which can practically be analysed, forcing users with large models to reduce the size of the model by a semi-automated process of model reduction (e.g. [10]). Additionally, encoding complex transition functions in these tools is laborious, making expression of realistic biological models difficult.

Proofs of stabilization in biological systems are complicated by the complex temporal relationships between interacting elements which are necessary for stability, and prevent the use of scalable techniques which abstract these details away. These temporal interactions are necessary to describe systems which show adaptation [16], or timed switches [17]. Previous work presented an algorithm for proving stability [6] which is sound and complete, and reverts to an exhaustive search for cycles of increasing length (up to the diameter of the system) if stability cannot be proved rapidly. Failure to find multiple self-loop states or cycles proves the stability of the system. Thus, if a stable model is resistant to quick proofs of stability, the search for cyclic counterexamples can be prohibitive. As a result of the rapid growth in the number and diversity of biological qualitative

networks, we have recently identified several realistic, stable models which resist the approach of [6].

In this paper we tackle the problem of proving instability with a new modular approach. This greatly increases the speed of proving stability in several systems by rapidly identifying loops arising from cyclic instabilities. Previous techniques proved stability by taking lemmas of the form: $[FG(p1) \wedge \cdots \wedge FG(pk)] \Rightarrow FG(q)$, where $p1 \ldots pk$ are formulae over the inputs of a given component, and q is a formula about the component's output, where F and G denote "eventually" and "always" in linear temporal logic [18]. If this fails to find a single self-loop state, exhaustive searches for multiple self-loop states and loops are performed.

To avoid this costly calculation, our new approach searches for counterexamples using a divide/conquer technique, based around a modular approach for proving stability. If a single or multiple self-loops cannot be found when analyzing the system, the state space is divided into two, and each individually searched for local self-loops. Finally, through an analysis of the cut and a small number of steps of simulation either counter examples of cycles are found or stability is proved.

Our new approach increases the speed of the proof of instability and stability (in the case that stability cannot be proved easily) by over 2 orders of magnitude compared with previous approaches [6] in addition to being sound and complete.

2 Verifying Stability in Qualitative Networks

Qualitative Networks (QNs) [8] have been extensively used to model biological phenomena. A QN $Q(V, T, N)$, of granularity $N + 1$ consists of n variables: $V = (v_1, v_2, \cdots, v_n)$. The state of the system is a finite map $s : V \to \{0, 1, \cdots N\}$. The set of initial states is the set of all states. Each variable $v_i \in V$ has a *target function* $T_i \in T$ associated with it: $T_i : \{0, 1, \cdots, N\}^n \to \{0, 1, \cdots N\}$. Qualitative networks update the variables using synchronous parallelism.

Target functions in qualitative networks direct the execution of the network from state $s = (d_1, d_2, \cdots, d_n)$. The *next state* $s' = (d'_1, d'_2, \cdots, d'_n)$ is computed by:

$$d'_v = \begin{cases} d_v + 1 & d_v < T_v(s) \text{ and } d_v < N, \\ d_v - 1 & d_v > T_v(s) \text{ and } d_v > 0, \\ d_v & \text{otherwise.} \end{cases} \quad (1)$$

A target function of a variable v is typically a simple algebraic function, such as sum, over several other variables $w_1, w_2, \cdots w_m$. Variables $w_1, w_2, \cdots, w_m$ are called *inputs* of v and v is an *output* of each one of $w_1, w_2, \cdots, w_m$. The input function induces a *dependency* graph of the network with the variables as nodes, where an edge (u, v) exists iff u is an *input* of v.

A QN $Q(V, T, N)$ defines a state space $\Sigma = \{s : V \to \{0, 1, \cdots N\}\}$ and a transition function $\delta : \Sigma \to \Sigma$, where $\delta(s) = s'$ such that for every $v \in V$, $s'(v)$ depends on $T_v(s)$ as in Eq. 1. For a state $s \in \Sigma$ we denote $s(v)$ also by s_v. Likewise, $\delta(s)_v = \delta(s)(v)$ is the value of v in $\delta(s)$. We say that a state s is

recurring if it is possible to get back to s after a finite number of applications of δ. That is, if for some $i > 0$, we have $\delta^i(s) = s$. As the state space of a qualitative network is finite, the set of recurring states is never empty. We say that a network is *stabilizing* if there exists a unique recurring state s. That is, there is a unique state s such that $\delta(s) = s$, and for every other state s' and every $i > 0$ we have $\delta^i(s') \neq s'$. Intuitively, this means that starting from an arbitrary state, we always end up in a self-loop state and always the same one. For an *unstable* network, we have two possibilities: (a)*multiple self-loop states*; (b) at most one self-loop state but *non-trivial cycles*.

In [6], the problem of determining whether a network stabilizes or not is solved by proving local lemmas about the range of values a variable can eventually take depending upon already proven lemmas about its inputs. Each newly proven lemma is then used to strengthen the lemmas about its outputs until nothing changes. The order in which variables are picked for strengthening is arbitrary. The proven lemmas can sometimes be enough to determine that a network stabilizes. If not, an explicit search for counter-examples is carried out. First, the existence of multiple self-loops is checked by encoding it as a boolean satisfiability problem. If this check fails, bounded model-checking (BMC) is used to find non-trivial cycles of increasing length. For stabilizing networks where modular lemmas are not strong enough to show the same (see Fig. 1b), BMC unrolling to a length more than or equal to the system's diameter is required to show non-existence of cycles, which is infeasible even for moderately sized networks.

Here, we revisit the modular proof-based approach from [6], using a technique similar in spirit to abstract interpretation [19, 20]. We present a novel, scalable instability detection algorithm in Sec. 3 that reuses the old algorithm as one of its sub-procedures.

2.1 Over-approximating Recurring States

All states of a QN are considered initial states. Let $\Sigma_i, i \geq 0$ be the set of states of the QN that are reachable in i or more steps starting from some initial state. Note that if a state s of a QN is not reachable in i or more steps, then it is not reachable in i' or more steps for every $i' > i$. Hence, $\Sigma = \Sigma_0 \supseteq \Sigma_1 \supseteq \Sigma_2 \supseteq \cdots$ is a decreasing sequence. Since the state space is finite, there will exist $l \leq \text{diameter}(\Sigma)$, such that $\Sigma_{l'} = \Sigma_l$, for every $l' \geq l$. The set Σ_l is the set of all recurring states of the network, which is a singleton for a *stabilizing* network.

Computing the exact reachability sets is not feasible in practice. Instead we try to over-approximate the set of recurring states by using a layer of abstraction to represent sets of states. Analogous to interval domain from abstract interpretation [19, 20], for each variable v, we just keep track of the range of its possible values. Let $[i, j]$, $i, j \in \mathbb{Z}$, $i \leq j$, denote the interval containing all integers from i to j inclusive. Interval $[i_1, j_1]$ *contains* another interval $[i_2, j_2]$ iff $i_1 \leq i_2$ and $j_1 \geq j_2$. Let $\mathcal{L}_N$ be the the set of all intervals contained in $[0, N]$. An element $(I_1, \cdots, I_n)$ of the set $\mathcal{S} = \mathcal{L}_N^n$ represents a sub-space of Σ where variable v can take all values in the interval I_v. We refer to elements of $\mathcal{S}$ as *regions*. A region $\rho_I = (I_1, \cdots, I_n)$ is said to *contain* another region $\rho_J = (J_1, \cdots J_n)$ (written as

$\rho_I \sqsupseteq \rho_J$) iff I_k *contains* J_k for every $1 \leq k \leq n$. $(\mathcal{S}, \sqsupseteq)$ is a finite (hence complete) partial order with $[0, N]^n$ as the $\top$ element. A region $\rho = ([l_1, h_1], \cdots, [l_n, h_n])$ can be equivalently represented by functions V^ρ_{lo} and V^ρ_{hi} s.t. $\forall v \in \{0, \cdots, n\}$, $V^\rho_{lo}(v) = l_v$ and $V^\rho_{hi}(v) = h_v$. The set of states s in ρ s.t. $\delta(s)$ is outside of ρ is denoted by $\rho\bullet = \{s \in \rho \wedge \delta(s) \notin \rho\}$. If $\rho\bullet$ is not empty, we say the region ρ is *open* wrt δ, otherwise it is *closed*.

Let v be a variable and $(w_1, \cdots, w_m)$ be its inputs. We define a function $F : \mathcal{S} \to \mathcal{S}$, which updates the bounds of eventually possible values of v using the bounds on the values of its inputs as restricted to ρ. Let $(w_1, \cdots, w_m) = \text{inputs}(v)$. We compute the set of values of the target function T_v applied to all possible input combinations in ρ and use that to update the interval of v. Formally, $F(\rho) = (f_1(\rho), \cdots f_n(\rho))$, with

$$f_v(\rho) = [\min(\theta_v(\rho)), \max(\theta_v(\rho))], \text{ where} \tag{2}$$

$$\theta_v(\rho) = T_v([V^\rho_{lo}(w_1), V^\rho_{hi}(w_1)] \times \cdots \times [V^\rho_{lo}(w_m), V^\rho_{hi}(w_m)]) \tag{3}$$

by suitably lifting the definition of T_v to sets of states.

Note that F is monotonic because $\rho_1 \sqsupseteq \rho_2$ implies $T_v(\rho_1) \supseteq T_v(\rho_2)$ and thereby $f_v(\rho_1)$ *contains* $f_v(\rho_2)$. By Kleene's fixed point theorem, F will have a greatest fixed point νF, which can be computed by finite number of repeated applications of F on $\top$. Since F is monotonic, repeated applications of F on $\top$ would give rise to a decreasing sequence of regions $\top \sqsupseteq F(\top) \sqsupseteq F^2(\top) \sqsupseteq \cdots$. Every element of this sequence is a closed region, because an outgoing transition from some state s in $F^i(\top)$ would mean that in s, target function of some variable v takes a value not contained in $[V^{F^i(\top)}_{lo}(v), V^{F^i(\top)}_{hi}(v)]$, implying $F^{i+1}(\top) \not\sqsubseteq F^i(\top)$. Using these observations, we claim the following.

Lemma 1. *The greatest fixed point νF is an over-approximation of the set of recurring states in Σ.*

We can prove this by induction on the number of times F is applied. $\top = [0, N]^n = \Sigma$ contains all recurring states, which proves the base case. Assume the inductive hypothesis that for some $i \geq 0$, $F^i(\top)$ contains all recurring states. We show that one more application of F cannot remove at least one recurring states from $F^i(\top)$. For some state $s \in F^i(\top) \setminus F^{i+1}(\top)$, there exists a variable v s.t. $f_v(F^i(\top))$ does not contain s_v. For the sake of contradiction, assume s is recurring. Then, there exists $m > 0$ s.t. $\forall i \in \{1, \cdots, m-1\} \cdot \delta^i(s) \neq s$ and $\delta^m(s) = s$. Since all variables can change by at most 1 in each transition according to Eq. 1, there must exist $i, j \in \{0, \cdots, m-1\}$ s.t. $T_v(\delta^i(s)) \geq s_v$ and $T_v(\delta^j(s)) \leq s_v$. Since $F^i(\top)$ is closed, $\delta^i(s), \delta^j(s) \in F^i(\top)$. This means $f_v(F^i(\top))$ must contain s_v which leads to a contradiction. Thus, $F^{i+1}(\top)$ also contains all recurring states, proving our claim.

2.2 Computing the Greatest Fixed Point νF

We refer to one application of some f_v as an *update*. Then, νF is a fixed point of a system of equations, one equation corresponding to each f_v. The simplest

algorithm to compute νF is to repeatedly apply F until a fixed point is reached. One application of F corresponds to a parallel application of each f_v. This algorithm is far from optimal since it does not exploit the dependencies in the network. In the worst case, when each application of F changes either lower or upper bounds of exactly one of the variables by 1, it can perform $\mathcal{O}(Nn^2)$ updates. Biological models expressed as QNs are typically expected to have a small granularity, reflecting the high level of abstraction from the underlying physico-chemical nature. Since every f_i is monotonic, any sequential algorithm will find the greatest fixed point as long each f_i is applied a sufficient number of times. Each update can be compared to a lemma generation step (Algorithm 4 in [6]).

Computing νF is often scalable because most variables in a typical QN have a small number of inputs. Each update to a variable means going over all possible combinations of its inputs $w_1, \cdots, w_m$ within the current region ρ, i.e. $([V^\rho_{lo}(w_1), V^\rho_{hi}(w_1)] \times \cdots \times [V^\rho_{lo}(w_m), V^\rho_{hi}(w_m)])$, which is feasible when the number of inputs is small. In many cases the target functions are also monotonic in some inputs, which allows for checking only the boundaries of the region (instead of the complete Cartesian product) to get the min/max possible target function value.

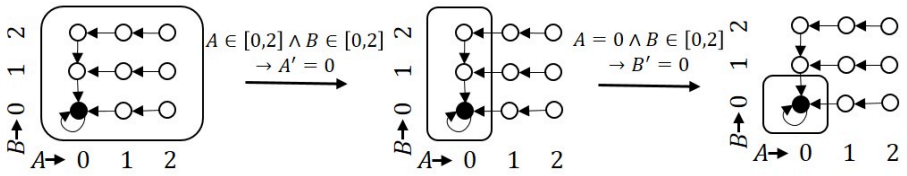
In cases where νF is a singleton, we safely conclude that the network stabilizes. We denote such networks as being *trivially stable*. When it is not a singleton, it may still be possible that the network is stable and the over-approximation is too coarse. Such networks are termed as being *non-trivially stable*.

2.3 Example

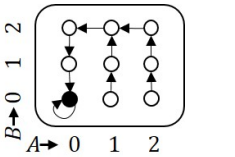
Fig. 1 shows how the fixed-point computation would proceed on the example transition systems. Each of them are of granularity 3 and have two variables A and B. The target functions corresponding to the transition system in Fig.1a are: $T_A(a, b) = 0$, $T_B(a, b) =$ if $a = 0$ then 0 else b. Target functions of the other two transitions systems are cumbersome to write and are hence skipped in the text. In case of Fig.1a, two updates are enough to determine that the only recurring state is $A = 0, B = 0$. In Fig.1b even though the network is stabilizing, it is not possible to strengthen the intervals of either A or B beyond $[0, 2] \times [0, 2]$. In Fig.1c, the network is unstable due to presence of a cycle, and one update each to intervals of A and B leads to the greatest fixed point region $[0, 1] \times [0, 1]$. We still have to explicitly check for presence of cycles within this smaller region.

3 Finding Instability

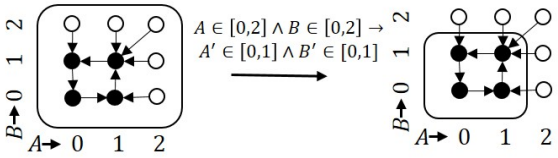
If the greatest fixed point νF is not a singleton, we can think of the following disjoint possibilities: (a) the network is unstable due to presence of (a.1) at least two self-loop states, or (a.2) at most one self-loop state but at least one non-trivial cycle; (b) the network stabilizes and νF is too coarse to conclude that, in which case it has exactly one self-loop and no non-trivial cycles. Checking (a.1) can be encoded as a formula satisfiability problem. A decision procedure



(a) Trivially Stable



(b) Non-trivially Stable



(c) Cyclic Instability

Fig. 1: Computing νF on transition systems of example QNs. Circles denote states and arrows between states denote transitions. Solid circle means that the state is recurring. Rounded rectangles are used to represent regions (interval domain).

is used to check the existence of two distinct states: u and w such that both are self-loops: $\forall i \in \{1, \cdots, n\} \cdot (\delta(u)_i = u_i) \wedge (\delta(w)_i = w_i)$. This decision procedure usually works very well. On the contrary, as mentioned earlier the check for distinguishing (a.2) and (b) is a brute force call to a decision procedure that searches for loops of increasing length (up to the diameter of the network). Especially in the case that (b) is true, this has prohibitive performance. This motivates development of a new algorithm to distinguish between cases (a.2) and (b).

First, we define some terminology. Let ρ be a region of the QN Q. A pair (ρ_1, ρ_2) of disjoint regions such that $\rho_1 \cup \rho_2 = \rho$ is called a *cut* of ρ. As both ρ_1 and ρ_2 are regions and $\rho_1 \cup \rho_2$ must be ρ, it follows that there exists some variable v and a value d such that $\rho_1 = \{s \in \rho \mid s(v) \leq d\}$ and $\rho_2 = \{s \in \rho \mid s(v) > d\}$. In this case it must also be the case that $\rho_1 \cup \rho_2 = \rho_1 \sqcup \rho_2$ (where $\sqcup$ is the join operation in the lattice of regions). Let $\rho_1 \bullet \rho_2 = \{s | s \in \rho_1 \wedge \delta(s) \in \rho_2\}$ be the set of states in ρ_1 that have a transition to some state in ρ_2 and $\delta(\rho_1 \bullet \rho_2)$ is the image of $\rho_1 \bullet \rho_2$ wrt δ. We have $\delta(\rho_1 \bullet \rho_2) \subseteq \rho_2$. A pair of sets of states $\gamma = (\gamma_{\rho_1}, \gamma_{\rho_2})$ is a *frontier* of the cut (ρ_1, ρ_2) iff $\rho_1 \bullet \rho_2 \subseteq \gamma_{\rho_1} \subseteq \rho_1$, $\delta(\rho_2 \bullet \rho_1) \subseteq \gamma_{\rho_1}$ and $\rho_2 \bullet \rho_1 \subseteq \gamma_{\rho_2} \subseteq \rho_2$, $\delta(\rho_1 \bullet \rho_2) \subseteq \gamma_{\rho_2}$. A cut (ρ_1, ρ_2) can have one of three natures:(1) *zero*-way iff there is no transition from a state in ρ_1 to a state in ρ_2 and also the other way;(2)*one*-way iff there are transitions in exactly one direction; (3) *two*-way if there are transitions in both directions. That is, (ρ_1, ρ_2) is: *zero*-way iff both $\rho_1 \bullet \rho_2$ and $\rho_1 \bullet \rho_1$ are empty; *one*-way iff exactly one of $\rho_1 \bullet \rho_2$ and $\rho_2 \bullet \rho_1$ is empty; *two*-way iff both $\rho_1 \bullet \rho_2$ and $\rho_2 \bullet \rho_1$ are non-empty. The nature of a frontier is also defined similarly, based on directionality of transitions between γ_{ρ_1} and γ_{ρ_2}. Note that all frontiers of a cut have the same nature as the cut, hence determining the nature of some frontier suffices to determine the nature of the cut.

We use (s_0, k) to denote a *simple* cycle $(s_0, \delta(s_0) \neq s_0, \cdots, \delta^{k-1}(s_0) \neq s_0, \delta^k(s_0) = s_0)$ of length $k > 0$. A cycle (s_0, k) is non-trivial iff $k > 1$. We say that (s_0, k) is within a region ρ if $\forall i \in \{0, \cdots, k\} \cdot \delta^i(s_0) \in \rho$. Algorithm 1 guarantees to find a non-trivial cycle (if one exists) within a region ρ by using two generic procedures Shrink and Cut.

Shrink: $\mathcal{S} \rightarrow \mathcal{S}$. Shrink takes a region ρ as input and returns a region $\rho' \sqsubseteq \rho$ such that ρ' still contains every cycle that exists within ρ.

Cut: $\mathcal{S} \rightarrow (\mathcal{S} \times \mathcal{S}) \times (\mathcal{P}(\Sigma) \times \mathcal{P}(\Sigma))$. Cut takes a region ρ as input and returns a cut (ρ_1, ρ_2) of ρ and a frontier $\gamma = (\gamma_{\rho_1}, \gamma_{\rho_2})$ of the cut.

Later in the section, we describe concrete implementations of these procedures that were used in our experiments, but it should be noted that every implementation that follows the specifications would work as far as correctness of Algorithm 1 is concerned. Algorithm 1 is a recursive procedure that first applies Shrink to the input region ρ to find a smaller region containing all cycles that exist within ρ. If the smaller region contains a single state, we can conclude that ρ does not have non-trivial cycles. Otherwise, we use Cut to split the shrunk region in two disjoint sub-regions ρ_1 and ρ_2 to which Algorithm 1 can be applied recursively. If a cycle is found within one of the sub-regions, it can be returned as a cycle within ρ. In case no cycle is found within either of the sub-regions, we still have to look for cycles that may exist across the cut (ρ_1, ρ_2). This is done using Algorithm 2.

Algorithm 2 uses a frontier $\gamma = (\gamma_{\rho_1}, \gamma_{\rho_2})$ of the cut (ρ_1, ρ_2) found by Cut. If the cut is *one/zero*-way, there can't exist a non-trivial cycle across this cut (existence of a cycle across (ρ_1, ρ_2) would imply that there is at least one transition from a state in ρ_1 to a state in ρ_2 and also the other way and hence both $\rho_1 \bullet \rho_2$ and $\rho_2 \bullet \rho_1$ would be non-empty). We use a frontier to find the nature of the cut because $\rho_1 \bullet \rho_2$ and $\rho_2 \bullet \rho_1$ can be difficult to compute exactly. How to compute a frontier and determine its nature is described in more detail in Sec. 3.1.

In case the cut is *two*-way, we start an exhaustive search for a cycle across the cut by sequentially running simulations starting at each state in γ_{ρ_1}. Each simulation is run until either the current state in the simulation is outside of region ρ or a lasso is found. If there is cycle across the cut, this search is guaranteed to find it because there would exist a state s_{ρ_1} on the cycle such that $\delta(s_{\rho_1}) = \rho_2$ and hence $s_{\rho_1} \in \rho_1 \bullet \rho_2 \subseteq \gamma_{\rho_1}$.

Lemma 2. FindInstability(νF) *returns a simple non-trivial cycle* $(s_0, k > 1)$ *iff there exists at least one non-trivial cycle in* Q *and returns* null *otherwise.*

A rigorous proof of this statement can be sketched using structural induction on regions following the reasoning above.

3.1 Concrete Implementations of Shrink and Cut

In this section we describe the concrete implementation of Shrink and Cut that we used in our experiments.

Shrink: Given a region ρ, consider the modified target function T_v^ρ of a variable v:

$$T_v^\rho(s) = \min(V_{hi}^\rho(v), \max(V_{lo}^\rho(v), T_v(s)))$$

Algorithm 1: FindInstability

Input: A region ρ
Output: Either a simple non-trivial cycle $(s_0, k > 1)$ within ρ or null if ρ does not contain a non-trivial cycle

$\rho \leftarrow \textsc{Shrink}(\rho)$
if ρ contains a single state **then**
  return null
else
  $(\rho_1, \rho_2), \gamma \leftarrow \textsc{Cut}(\rho)$
  $res_1 \leftarrow \textsc{FindInstability}(\rho_1)$
  if $res_1 \neq$ null **then return** res_1
  $res_2 \leftarrow \textsc{FindInstability}(\rho_2)$
  if $res_2 \neq$ null **then return** res_2
  return $\textsc{FindCycleAcrossCut}((\rho_1, \rho_2), \gamma)$
end

Algorithm 2: FindCycleAcrossCut

Input: A cut (ρ_1, ρ_2) of the region $\rho_1 \cup \rho_2 = \rho$ and frontier $\gamma = (\gamma_{\rho_1}, \gamma_{\rho_2})$ of the cut. The regions ρ_1 and ρ_2 do not have any cycles within them.
Output: Either a simple non-trivial cycle $(s_0, k > 1)$ within ρ s.t. $\exists i, j \in \{0, \cdots, k\}.\delta^i(s_0) \in \rho_1 \wedge \delta^j(s_0) \in \rho_2$ or null if there is no such cycle

if γ is *one*-way or *zero*-way **then**
  return null
else
  $iter_\gamma \leftarrow \mathsf{iterator}(\gamma_{\rho_1})$
  $cyc \leftarrow$ null
  while $cyc =$ null $\wedge \neg$exhausted?$(iter_\gamma)$ **do**
    $cur \leftarrow \mathsf{getElemAndAdvance}(iter_\gamma)$
    $seen, i \leftarrow \mathsf{emptyMap}, 0$
    while $cur \notin \mathsf{keys}(seen) \wedge cur \in \rho$ **do**
      $seen[cur] \leftarrow i$
      $cur, i \leftarrow \delta(cur), i + 1$
    end
    if $cur \in \rho$ **then**
      $len \leftarrow i - seen[cur]$
      if $len > 1$ **then** $cyc \leftarrow (cur, len)$
  end
  return cyc
end

The modified transition function δ_ρ is defined using Eq. 1 by replacing T_v by T_v^ρ. The intention with T_v^ρ is to create target functions tailored to ρ so that ρ is closed wrt δ_ρ, while still preserving all transitions that are completely within ρ. This is done by forcing the target function value to be within $[V_{lo}^\rho(v), V_{hi}^\rho(v)]$, truncating to the upper/lower bound if the original value is too large/too small.

The function F_ρ is defined by replacing T_v by T_v^ρ in Eq. 3. Let $\mathcal{S}_\rho$ be the set of all regions contained within ρ and $(\mathcal{S}_\rho, \sqsupseteq)$ be the corresponding partial order. It can be shown that for every state s: (1) $s \in \rho \setminus \rho\bullet \rightarrow \delta_\rho(s) = \delta(s)$, and (2) $s \in \rho\bullet \rightarrow \delta_\rho(s) \in \rho$. This means ρ is closed wrt to δ_ρ and hence the greatest fixed point $\nu_{\mathcal{S}_\rho} F_\rho$ on $\mathcal{S}_\rho$ is well defined. Thus, $\nu_{\mathcal{S}_\rho} F_\rho$ is an over-approximation of the set of states in ρ that are recurring wrt δ_ρ. Also, if there exists a cycle (wrt δ) within ρ, it will also be a cycle wrt δ_ρ and hence be within $\nu_{\mathcal{S}_\rho} F_\rho$. $\textsc{Shrink}(\rho) = \nu_{\mathcal{S}_\rho} F_\rho$ can be computed following the approach in Sec. 2.

Cut: The straight-forward way to cut a region ρ is to split the interval of one of the variables into two. Let $\rho|_{[v]=I}$ denote the region obtained from ρ by replacing the interval corresponding to v by I and keeping other intervals unchanged. Upon splitting v at α with $V_{lo}^\rho(v) \leq \alpha < V_{hi}^\rho(v)$, we get the cut (ρ_1, ρ_2) with $\rho_1 = \rho|_{[v]=[V_{lo}^\rho(v), \alpha]}$ and $\rho_2 = \rho|_{[v]=[\alpha+1, V_{hi}^\rho(v)]}$. Since we know that for every state s and variable v, $\delta(s)_v$ differs from s_v by at most 1, we can safely choose $(\gamma_{\rho_1}, \gamma_{\rho_2})$ as a frontier of this cut, where $\gamma_{\rho_1} = \{s | s \in \rho|_{[v]=[\alpha,\alpha]}\}$ and $\gamma_{\rho_2} = \{s | s \in \rho|_{[v]=[\alpha+1,\alpha+1]}\}$. The nature of this cut can be determined by encoding the problem of checking the nature of the frontier as a boolean satisfiability problem. Checking existence of a transition from γ_{ρ_1} to γ_{ρ_2} is equivalent to checking existence of a state w s.t. $w \in \rho_1 \wedge w_v = \alpha \wedge \delta(w) \in \rho_2 \wedge \delta(w)_v = \alpha + 1$, where $w \in \rho_1$ would be a conjunction of simple predicates restricting the range of each variable in w to be within ρ_1 and likewise for $\delta(w) \in \rho_2$.

We have n choices for variable v and then $V_{hi}^\rho(v) - V_{lo}^\rho(v)$ choices for the splitting point α for each v. $\textsc{Cut}(\rho)$ enumerates through all these possibilities and returns the first one that has a *zero/one*-way nature. A *zero/one*-way cut saves the effort of finding cycles across it later. If all possibilities are *two*-way, then it returns a balanced cut, one for which ρ_1 and ρ_2 are similar in size.

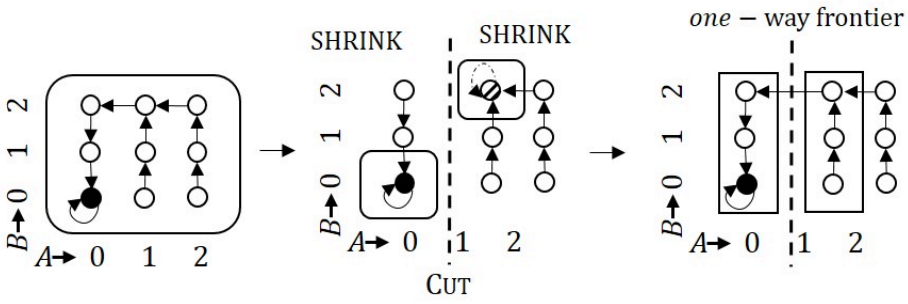
3.2 Efficiency of FindInstability

In the worst case, Algorithm 1 can be equivalent to a brute-force enumeration over all states of the QN that tries to build cycles by running simulations, but the effectiveness of Shrink to eliminate parts of the state space that do not contain cycles and the existence of *zero/one*-way cuts makes it scale to considerably large QNs. Experiments show that this can be better than a bounded model-checking based approach by a few orders of magnitude. However, a stable system which resisted an initial Shrink and for which only a *two*-way cut could be found would be expected to require such a brute-force search. One possible approach to reduce the search space would be to search specifically for *two*-way cuts whose frontiers can be bisected into a *two*-way cut and a *zero*-way cut. In the continued development of new models, we hope to discover examples of this behaviour, and will be look to address this issue in future. In the absence of such models at

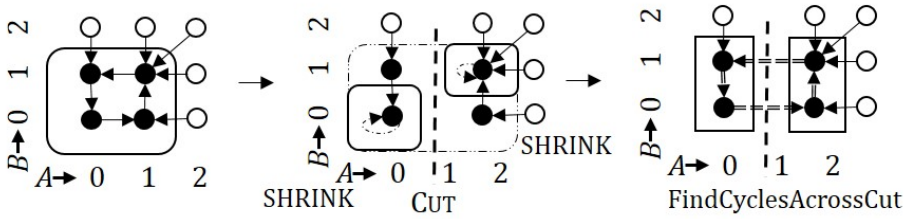
present however we have here restricted our testing to implementations of the Algorithm 1.

3.3 Example

Fig.2 illustrates how Algorithm 1 makes progress on examples from Fig.1b and Fig.1c. In Fig.2a, for the non-trivially stable system, CUT splits the interval of A at 0 to produce two sub-regions. One SHRINK operation on each of the sub-regions reduces them to singletons, hence confirming non-existence of cycles within them. Owing to the cut being *one*-way, non-existence of cycles across it is also easily checked. In Fig.2b, Algorithm 1 is applied to the fixed point obtained previously ($[0,1] \times [0,1]$). Similar to the previous example, interval of A is split at 0, and the sub-regions get reduced to singletons by SHRINK. However, the cut is *two*-way and Algorithm 2 successfully finds a cycle by running a simulation from a point in the frontier.



(a) Non-trivially Stable



(b) Cyclic Instability

Fig. 2: Progress of Algorithm 1 on transition systems in Fig.1b,1c. Dotted lines represent the splitting point in CUT. Dotted arrows represent the modifications made to transition functions of sub-regions in order to apply SHRINK. Sharp-edged rectangles denote frontiers. Other notation is the same as in Fig.1.

4 Benchmarks and Evaluation

We implemented the approach in a tool called BioModelAnalyzer+ (BMA+) and compared directly against the approach from [6] implemented within BioMod-

elAnalyzer (BMA, [18, 21, 22]). For each benchmark, BMA+ first computes the greatest fixed point νF. If stability cannot be proved, it tries to find a counter-example of type multiple self-loops by encoding it as a satisfiability problem. If such a counter-example cannot be found, Algorithm 1 is used to check for existence of a cycle. All calculations were performed single-threaded on a Windows 8 PC with an Intel i7 processor @ 2.1 GHz. There was an upper bound of 2GB on memory usage. Time out was set at 15 mins. We used Z3 [23] version 3 as the decision procedure. All benchmarks are available at `http://www.cs.le.ac.uk/people/npiterman/publications/2014/instability/`.

We study the results for models of different nature separately.

4.1 Trivially Stable Systems and Systems with Multiple Self-Loops

We benchmarked our new approach using a range of well characterized stable and bifurcating (multiple self-loops) systems from previous studies [6, 8, 18, 21, 24]. Both tools run extremely fast and show similar performance for these benchmarks, as would be expected. See Table 1. However, we still see a slowdown for BMA+ arising from changes in the procedure which are not relevant here.

Model	**Nature**	**N+1**	**|E|**	**|V|**	**BMA**	**BMA+**	$\frac{\text{BMA}}{\text{BMA+}}$
Leukaemia	TS	3	81	51	71	94	0.8
Diabetes	TS	3	125	87	66	93	0.7
Budding yeast	TS	5	26	16	55	80	0.7
VPC lin15KO	TS	3	140	85	56	122	0.5
Dicty single cell	TS	2	12	8	47	111	0.4
Skin 1D unstable	BF	5	89	75	2206	1973	1.1
Skin 1D	BF	5	94	75	239	238	1
Skin 1D unstable 2	BF	5	89	75	383	357	1.1
Skin 2D 5x2 TF	BF	5	239	198	337	315	1.1
MCP Array	BF	2	104	45	140	241	0.6

Table 1: Results for trivially stable (TS) and bifurcating (BF) models. N+1 is the granularity, |E| is the number of edges in dependency graph, |V| is the number of variables. BMA and BMA+ denote the running time of the respective tools in milliseconds.

4.2 Systems with cyclic instability

Oscillations occur in a wide range of different biological systems. In nerves under constant stimulation the patterns of opening and closing of ion channels in an action potential are expected to generate cycles. Oscillations are also widely found in different biological systems as a mechanism for synchronizing populations of cells in organs and whole animals. In *Dictyostelium discoideum* coordinated oscillations in groups of cells signal the transition from unicellular growth to multicellular development [25]. BMA+ consistently performs almost an order of magnitude better than BMA for these benchmarks. See Table 2.

Model	**N+1**	**|E|**	**|V|**	**K**	**BMA**	**BMA+**	$\frac{\text{BMA}}{\text{BMA+}}$
Dicty population	2	71	35	5	60066	2541	23.6
Firing Neuron	2	21	21	6	218	458	0.5
LModel	4	105	25	5	43934	9865	4.5
Leukaemia unstable	3	92	57	5	4497	446	10.1
SSkin 1D	5	46	30	11	TO	132350	>6.8
SSkin 2D 3 cells 2 layers	5	64	40	18	TO	2706	>322.6

Table 2: Results for models with cyclic instabilities. K is the length of the cycle found by BMA+. TO denotes a time out. In comparing systems where BMA times out, the speed up is calculated relative to the time limit (15 minutes) and noted with ">". Other notation is same as in Table 1

4.3 Non-trivially stable systems

The non-trivially stable systems highlight important examples of stable biological systems which cannot be proved to be stable with Shrink alone. Chemotaxis in *E. coli* is a paradigm for bacterial signaling. Attractants and repellents bind to a receptors at the cell pole, altering the activity of the kinase CheA. This in turn both alters the switching behavior of the flagellar motor and changes the sensitivity of the receptor array to allow for adaptation. The alteration of receptor sensitivity is slower than motor activation, ensuring that the flagellar switching behavior reverts to an equilibrium state in an unchanging environment. Similarly, an action potential passes along a neuron by the opening of ion channels (triggered by changes in the local membrane potential), followed by a delayed closure of the pore. The time delay aspects in both of these models (speed of the adaptation machinery in chemotaxis, and the slow closure of the ion channels) lead to them both being stable systems. However, proving stability is non-trivial because of this property. Table 3 shows the benefit of BMA+ over BMA. We observe significant speed up in all cases.

Particularly noteworthy is the improvement in the calculation of the stable state in the *E. coli* signaling system, as despite being a significantly smaller model than many others presented here (in terms of number of variables and edges), proofs of stability using the previous approach were prohibitively costly. Through a single application of the cut, the stability of the system was proved comparably quickly to a simple example of the same size.

Model	**N+1**	**|E|**	**|V|**	**BMA**	**BMA+**	$\frac{\text{BMA}}{\text{BMA+}}$
Ion channel	2	7	10	499	173	2.9
Lambda phage	2	13	8	3113	197	15.8
Resting neuron	2	28	21	TO	244949	>3.7
E. coli chemotaxis	5	10	9	TO	250	>3600

Table 3: Results for non-trivially stable models. TO denotes time out. Other notation is same as in Tables 1

and 2.

5 Conclusions

This paper describes a new algorithm for the formal analysis of biological models, which offers a rapid approach for proving instability arising from loops. This technique builds on previous approaches by rapidly searching for cycles in cases where stability cannot be proved trivially. We find a large speed up for proving instability of cyclic systems, but additionally we show that it offers an impressive speed up when considering the behavior of stable systems with timed switches, such as bacterial chemotaxis. By proving stability in these new models our findings further reinforce the importance of inherent biological robustness [6] in signaling systems. In order to build accurate models of chemotaxis signaling in *E. coli* and action potentials in a neuron we need to include timing effects to reproduce realistic biological behavior. Without these timing effects, the models unnaturally show bifurcation and cycling behavior respectively. Our new approach allows us to better identify loops and pseudo-loops in biological signaling systems observed in this biomedically important class of systems.

Acknowledgments. We thank Dr C Pears and Dr A Watson for insightful comments and advice.

References

1. Fisher, J., Henzinger, T.: Executable cell biology. Nature Biotechnology **25** (2007) 1239–49
2. Bonzanni, N., Feenstra, K., Fokkink, W., Krepska, E.: What can formal methods bring to systems biology? In: Formal Methods. Volume 5850 of Lecture Notes in Computer Science., Springer (2009) 16–22
3. Chabrier-Rivier, N., M., C., Danos, V., Fages, F., Schächter, V.: Modeling and querying biomolecular interaction networks. Theor. Comput. Sci. **325** (2004) 25–44
4. Bonzanni, N., Krepska, E., Feenstra, K.A., Fokkink, W., Kielmann, T., Bal, H., Heringa, J.: Executing multicellular differentiation: Quantitative predictive modelling of *C.elegans* vulval development. Bioinformatics **25** (2009) 2049–2056
5. Fisher, J., Piterman, N., Hajnal, A., Henzinger, T.: Predictive modeling of signaling crosstalk during *c. elegans* vulval development. PLoS Computational Biology **3** (2007) e92
6. Cook, B., Fisher, J., Krepska, E., Piterman, N.: Proving stabilization of biological systems. In: 12th International Conference on Verification, Model Checking, and Abstract Interpretation. Volume 6538 of Lecture Notes in Computer Science., Springer-Verlag (2011) 134–149
7. Kauffman, S.: Metabolic stability and epigenesis in randomly constructed genetic nets. Journal of Theoretical Biology **22** (1969) 437 – 467
8. Schaub, M., Henzinger, T., Fisher, J.: Qualitative networks: A symbolic approach to analyze biological signaling networks. BMC Systems Biology **1** (2007)
9. Krumsiek, J., Marr, C., Schroeder, T., Theis, F.J.: Hierarchical differentiation of myeloid progenitors is encoded in the transcription factor network. PLoS ONE **6** (2011) e22649
10. 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 Comput Biol **9** (2013) e1003286

11. Bonzanni, N., Garg, A., Feenstra, K.A., Schütte, J., Kinston, S., Miranda-Saavedra, D., Heringa, J., Xenarios, I., Göttgens, B.: Hard-wired heterogeneity in blood stem cells revealed using a dynamic regulatory network model. Bioinformatics **29** (2013) i80–i88
12. Huang, S.: Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery. Journal of Molecular Medicine **77** (1999) 469–480
13. Naldi, A., Berenguier, D., Fauré, A., Lopez, F., Thieffry, D., Chaouiya, C.: Logical modelling of regulatory networks with GINsim 2.3. Biosystems **97** (2009) 134 – 139
14. Cimatti, A., Clarke, E.M., Giunchiglia, F., Roveri, M.: Nusmv: A new symbolic model verifier. In: Proceedings of the 11th International Conference on Computer Aided Verification. CAV '99, London, UK, UK, Springer-Verlag (1999) 495–499
15. Naldi, A., Thieffry, D., Chaouiya, C.: Decision diagrams for the representation and analysis of logical models of genetic networks. In: Computational Methods in Systems Biology. Volume 4695 of Lecture Notes in Computer Science., Springer (2007) 233–247
16. Wadhams, G.H., Armitage, J.P.: Making sense of it all: bacterial chemotaxis. Nat. Rev. Mol. Cell Biol. **5** (2004) 1024–1037
17. Hille, B.: Ion Channels of Excitable Membranes (3rd Edition). 3rd edition edn. Sinauer Associates Inc 2001-07 (2001)
18. Claessen, K., Fisher, J., Ishtiaq, S., Piterman, N., Wang, Q.: Model-checking signal transduction networks through decreasing reachability sets. In Sharygina, N., Veith, H., eds.: CAV. Volume 8044 of Lecture Notes in Computer Science., Springer (2013) 85–100
19. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Principles of Programming Languages. (1977) 238–252
20. Cousot, P., Cousot, R.: Static determination of dynamic properties of programs. In: Proceedings of the Second International Symposium on Programming, Dunod, Paris, France (1976) 106–130
21. Benque, D., Bourton, S., Cockerton, C., Cook, B., Fisher, J., Ishtiaq, S., Piterman, N., Taylor, A., Vardi, M.: BMA: Visual tool for modeling and analyzing biological networks. In: 24th International Conference on Computer Aided Verification. Volume 7358 of Lecture Notes in Computer Science., Springer (2012) 686–692
22. Taylor, A.S., Piterman, N., Ishtiaq, S., Fisher, J., Cook, B., Cockerton, C., Bourton, S., Benque, D.: At the interface of biology and computation. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM (2013) 493–502
23. de Moura, L., Bjørner, N.: Z3: An Efficient SMT Solver Tools and Algorithms for the Construction and Analysis of Systems. In Ramakrishnan, C., Rehof, J., eds.: Tools and Algorithms for the Construction and Analysis of Systems. Volume 4963/2008 of Lecture Notes in Computer Science., Berlin, Heidelberg, Springer Berlin (2008) 337–340
24. Beyer, A., Thomason, P., Li, X., Scott, J., Fisher, J.: Mechanistic insights into metabolic disturbance during type-2 diabetes and obesity using qualitative networks. T. Comp. Sys. Biology **12** (2010) 146–162
25. Söderbom, Fredrik and Loomis, William F: Cell-cell signaling during Dictyostelium development. Trends in microbiology **6** (1998) 402–406