Efficient Handling of Large Signalling-Regulatory Networks by Focusing on Their Core Control

Aurélien Naldi[1], Pedro T. Monteiro[2], and Claudine Chaouiya[2]

[1] Center for Integrative Genomics, Fac. of Biology and Medicine, Univ. of Lausanne, Switzerland
[2] IGC, Instituto Gulbenkian de Ciência, Oeiras, Portugal

Abstract. Considering the logical (Boolean or multi-valued) asynchronous framework, we delineate a reduction strategy for large signalling and regulatory networks. Consequently, focusing on the core network that drives the whole dynamics, we can check which attractors are reachable from given initial conditions, under fixed or varying environmental conditions.

More specifically, the dynamics of logical models are represented by (asynchronous) state transition graphs that grow exponentially with the number of model components. We introduce adequate reduction methods (preserving reachability of the attractors) and proceed with model-checking approaches.

Input nodes (that generally represent receptors) and output nodes (that constitute readouts of network behaviours) are each specifically processed to reduce the state space. The proposed approach is made available within GINsim, our software dedicated to the definition and analysis of logical models. The new GINsim functionalities consist in a proper reduction of output components, as well as the corresponding symbolic encoding of logical models for the NuSMV model checker. This encoding also includes a reduction over input components (transferring their values from states to transitions labels). Finally, we demonstrate the interest of the proposed methods through their application to a published large scale model of the signalling pathway involved in T cell activation.

Keywords: Qualitative modelling, Logical modelling, Model checking, Regulatory networks, Signalling networks.

1 Introduction

As ever larger signalling and regulatory maps are being identified, there is a growing need for efficient computational means to analyse the behaviours induced by these networks. Among the numerous existing modelling approaches (see *e.g.*, reviews [4,18]), the logical framework provides a convenient way to convey current qualitative knowledge and proved useful to study a significant number of published models. Here, we rely on the formalism initially proposed by R. Thomas and co-workers [20], where discrete (Boolean or multi-valued) dynamics are represented as (asynchronous) state transition graphs (STG). For such models, the number of states grows exponentially with the number of regulatory components. We propose to tackle this combinatorial explosion, by applying adequate reduction methods and by proceeding with model-checking approaches.

This manuscript focuses on signalling-regulatory networks that encompass large numbers of input components (denoting external stimuli) and output components (used as readouts of network behaviours). In concrete biological networks, input components vary, often being externally controlled (*e.g.*, light availability, presence of nutrients, heat shock, etc.). It is thus relevant to consider that input components freely vary (*i.e.*, are under no specific control) and to slightly extend the current definition of logical regulatory graphs. Then, the STG encompasses transitions between all the states that share the same values for internal components, denoting the sole changes of the input components.

For these signalling-regulatory networks, our rationale is to reduce the complexity of the corresponding models, while ensuring the full preservation of the properties of interest. These relate to asymptotical behaviours embodied in terminal strongly components of the state transition graphs (referred to as attractors) as well as the reachability of those attractors from given initial conditions. Moreover, we analyse the possible switches between attractors, upon variations of the input components.

The reduction method presented in [11] possibly leads to the loss of some trajectories. Here, we show that when applied to output cascades, this reduction has no impact on reachability properties. However, it cannot be applied to input cascades if attractor reachability is to be preserved. Hence, we propose another strategy to lessen the size of the state space, by transferring the values of input nodes to transition labels, thus reducing the state space by at least 2^n (for n input Boolean nodes). Furthermore, we discuss the nature of stable states in these labelled state transition graphs, in the case of varying input components. Finally, we resort to model-checking to analyse these complex dynamics.

We demonstrate the potential of this approach on the large scale Boolean model that accounts for T cell activation as defined by Saez-Rodriguez *et al.* [16].

In [1], the authors introduce a “decimation algorithm“, which amounts to removing variables that have no impact on the long-term behaviour. While similar to ours, their method is valid for deterministic Boolean networks (with synchronous updates). Indeed, for such models, the reduction method presented in [11] has clearly different impacts on the dynamics. Considering asynchronous Boolean dynamics, Saadatpour *et al.* recently proposed a reduction strategy for large signalling transduction networks, relying on the fact that input cascades stabilise under constant input conditions [15]. Here, we aim at going further, first by ensuring that all the asynchronous dynamics is preserved (reducing only output cascades), second by considering varying input conditions.

The paper is organised as follows. First, we recall the basics of the logical formalism and of the model reduction method in Section 2. Section 3 presents the method that allows us to focus on the core network, reducing the output cascades and projecting the dynamics over the input components. Implementation aspects are discussed in Section 4. The proposed method is then applied to further analyse a published model of the signalling pathway involved in T cell activation [16]. The paper ends with some conclusions and prospects.

2 Background

In this section, we recapitulate (and extend) essential definitions concerning the logical formalism. Then, we recall the basics of the model reduction as defined in [11].

Definition 1. A logical signalling-regulatory graph (LSRG) $\mathcal{R} = (\mathcal{G}, \mathcal{K})$ *is defined by:*

- $\mathcal{G}$ *a set of* n *components partitioned into three subsets:* $\mathcal{I} = \{g_j\}_{j=1\dots n_i}$*, the set of input components,* $\mathcal{P} = \{g_j\}_{j=(n_i+1)\dots(n_i+n_p)}$*, the set of proper components, and* $\mathcal{O} = \{g_j\}_{j=(n_i+n_p+1)\dots n}$*, the set of output components.*
- *Discrete variables denoting the* levels *of the components:* $\forall_{g_i} v_i \in \mathcal{D}_i = \{0 \dots M_i\}$*. Then* $v = (v_i)_{g_i \in \mathcal{G}}$ *is a* state *and* $\mathcal{S} = \prod_{g_i \in \mathcal{G}} \mathcal{D}_i$ *is the* state space.
- *Logical functions defining the behaviours of the components:*
  - *For* $g_i \in \mathcal{P} \cup \mathcal{O}$*,* $\mathcal{K}_i$ *is a multi-valued function that specifies the (unique) target value* $\mathcal{K}_i(v)$ *of* g_i*, given the current state* v*:* $\mathcal{K}_i : \mathcal{S} \to D_i$.
  - *Input components* $g_i \in \mathcal{I}$ *are either set to constant values in their domains (*$\mathcal{K}_i : \mathcal{S} \to D_i$*):*

$$\forall v \in \mathcal{S}, \mathcal{K}_i(v) = v_i,$$

or freely vary ($\mathcal{K}_i : \mathcal{S} \to D_i \cup D_i^2$*):*

$$\forall v \in \mathcal{S}, \begin{cases} \textit{if } v_i = 0, & \mathcal{K}_i(v) = v_i + 1, \\ \textit{if } v_i = M_i, & \mathcal{K}_i(v) = v_i - 1, \\ \textit{if } M_i > 1 \textit{ and } 0 < v_i < M_i, & \mathcal{K}_i(v) = (v_i + 1, v_i - 1). \end{cases}$$

This definition slightly extends the classical definition of Logical Regulatory Graphs as introduced in *e.g.*, [11], since it specifically distinguishes input components, which account for environmental conditions and are either strictly controlled (*i.e.*, kept constant to their current values) or not (*i.e.*, freely vary).

The partition of $\mathcal{G}$, the set of regulatory components, will be useful in what follows. Note however that it could be partially derived from the logical functions $\mathcal{K}$. Indeed, given a proper or output component $g_i \in \mathcal{P} \cup \mathcal{O}$, one can define the set of nodes that influence g_i, denoted $Reg(g_i)$: $\forall g_k \in \mathcal{G}$, $g_k \in Reg(g_i)$ iff $\exists v \in \mathcal{S}, \mathcal{K}_i(v) \neq \mathcal{K}_i(v')$, where $v_k = v'_k \pm 1$ and $v_j = v'_j$, $\forall j \neq k$. If $g_k \in Reg(g_i)$, then there is an interaction from g_k to g_i and its sign can also be deduced from the logical function $\mathcal{K}_i$ (see *e.g.*, [11] for further detail). Moreover, we have

$$\begin{aligned} & g \in \mathcal{O} \Leftrightarrow \nexists g' \in \mathcal{G}, g \in Reg(g'), \\ & g \in \mathcal{P} \Leftrightarrow Reg(g) \neq \emptyset \text{ and } \exists g' \in \mathcal{G}, g \in Reg(g'), \\ & g \in \mathcal{I} \Leftrightarrow g \notin \mathcal{P} \cup \mathcal{O}. \end{aligned}$$

2.1 State Transition Graphs Representing LSRGs Dynamics

The behaviours of LSRGs are represented as State Transition Graphs, defined below. For proper and output components, we denote $\Delta_i(v)$ the "direction" of the update of g_i in state v:

$$\Delta_i(x) = \begin{cases} 0 & \text{if } K_i(v) = v_i, \\ \frac{|K_i(v)-v_i|}{K_i(v)-v_i} & \text{otherwise.} \end{cases}$$

Definition 2. *Given a LSRG $\mathcal{R} = (\mathcal{G}, \mathcal{K})$, its full, asynchronous State Transition Graph (*STG*) is a graph $\mathcal{E}(\mathcal{R}) = (\mathcal{S}, \mathcal{T})$ where:*

– *the nodes are the states in $\mathcal{S}$,*
– *the arcs denote transitions between states,*
  - *transitions over proper and output components are such that:*

$$(v, w) \in \mathcal{T} \subset \mathcal{S}^2 \iff \\ \exists g_i \in \mathcal{P} \cup \mathcal{O} \text{ s.t. } \mathcal{K}_i(v) \neq v_i,\ w_i = v_i + \Delta_i(v) \text{ and } w_j = v_j\ \forall j \neq i,$$

  - *transitions over a varying input component g_i are as follows (depending on v_i and M_i, there is a transition increasing v_i, a transition decreasing v_i or both):*

$$\begin{cases} (v, w) \in \mathcal{T}, \text{ with } w_i = v_i + 1,\ w_j = v_j\ \forall j \neq i, \iff v_i < M_i, \\ (v, w) \in \mathcal{T}, \text{ with } w_i = v_i - 1,\ w_j = v_j\ \forall j \neq i \iff v_i > 0. \end{cases}$$

We denote $\mathcal{E}_{cste}(\mathcal{R})$ (or simply $\mathcal{E}_{cste}$), the STG where input components are kept constant, and $\mathcal{E}_{var}$ the STG where input components freely vary. The STG $\mathcal{E}_{cste}$ is made of at least as many disconnected sub-graphs as the number of fixed input combinations (see Section 3.3 and Figure 1). In $\mathcal{E}_{var}$ these sub-graphs are connected through transitions over varying inputs, connecting all states that differ only in their values of input components.

Note that one can consider a sub-graph of the full STG, by defining initial state(s). This graph can be constructed by visiting all successors of the initial state(s) and proceeding with the exploration until no new state is encountered.

The main relevant properties to be analysed relate to LSRGs asymptotical behaviours that are called attractors. In STGs, they correspond to terminal Strongly Connected Components (SCCs). When input components are maintained constant, we have:

– *Stable states* (*i.e.*, terminal SCCs reduced to a unique state);
– *Complex attractors* (*i.e.*, terminal SCCs encompassing at least two states). Within these complex attractors, we can further distinguish *elementary terminal cycles*, in which all states have a unique outgoing transition.

In $\mathcal{E}_{var}$, there is no stable state (as defined above) and we need to revisit the definition of these attractors (see Figure 1 and Section 3.3).

Beside the identification of attractors, it is often important to check reachability properties, *e.g.*, which attractors are reachable from an initial condition, what are the properties of all trajectories leading to those reachable attractors, etc.

As already mentioned, we face a combinatorial explosion of the number of states that hampers efficient analysis of STGs. The consideration of priority classes, based on well-founded assumptions, amounts to choose between concurrent transitions and thus constitutes a convenient way of reducing the size of a STG (see [5]). Another approach, yet related, consists in reducing the size of the model (the number of its components). The method is briefly described below, a full description being available in [11].

2.2 Model Reduction

The reduction method presented in [11] consists in iteratively taking components off the model, which is adequately modified. The intuitive idea is to transfer the role of the reduced component to its regulators. Importantly, reduction of auto-regulated components is not allowed, to ensure that essential dynamical properties are preserved. Indeed, regulatory circuits are known to be at the origin of multi-stability (for positive circuits) and of stable oscillations (for negative circuits) [19]. In the same way, with our extended definition of logical models, where input components may freely vary, we will not allow reduction of these input nodes.

More precisely, taking a (non-autoregulated) component g_r off a LSRG $\mathcal{R} = (\mathcal{G}, \mathcal{K})$ leads to a new LSRG $\mathcal{R}^r = (\mathcal{G}^r, \mathcal{K}^r)$ with a reduced state space denoted $\mathcal{S}^r = \prod_{g_i \in \mathcal{G}^r = \mathcal{G} \setminus \{g_r\}} \mathcal{D}_i$. It is useful to define:

- The projection $\pi^r : \mathcal{S} \to \mathcal{S}^r$ such that $\forall v \in \mathcal{S}$, $\forall g_i \neq g_r$, $\pi^r(v)_i = v_i$;
- The "retrieval" function $s^r : \mathcal{S}^r \to \mathcal{S}$ such that $\forall x \in \mathcal{S}^r$, $\forall g_i \neq g_r$, $s^r(x)_i = x_i$ and $s^r(x)_r = \mathcal{K}_r(v)$, for $v \in \mathcal{S}$ such that $\pi^r(v) = x$. We say that $s^r(x)$ is the *representative state* of the equivalence class $[s^r(x)]_{\sim r}$ containing the states that are projected on $x \in \mathcal{S}^r$ (all these states differ solely in their values for the component g_r).

Furthermore, the reduction of g_r consists in modifying the logical functions (more precisely, the functions of those components g_i such that $g_r \in Reg(g_i)$):

$$\forall x \in \mathcal{S}^r,\ \mathcal{K}_i^r(x) = \mathcal{K}_i(s^r(x)).$$

Note that excluding auto-regulated and input components as candidates for reduction ensures the existence and uniqueness of the representative state. We recapitulate a number of properties concerning the dynamical behaviour of a reduced LSRG (details and proofs can be found in [11]). Let consider $\mathcal{R} = (\mathcal{G}, \mathcal{K})$ a LSRG and $\mathcal{R}^r = (\mathcal{G}^r, \mathcal{K}^r)$ its reduced version (taking off g^r). Let denote $\mathcal{E} = (\mathcal{S}, \mathcal{T})$ and $\mathcal{E}^r = (\mathcal{S}^r, \mathcal{T}^r)$ their STGs. We have,

1. $\forall u, v \in \mathcal{S}$, if u is a representative state ($u_r = \mathcal{K}_r(u)$), then: $(u, v) \in \mathcal{T} \Rightarrow (\pi^r(u), \pi^r(v)) \in \mathcal{T}^r$;

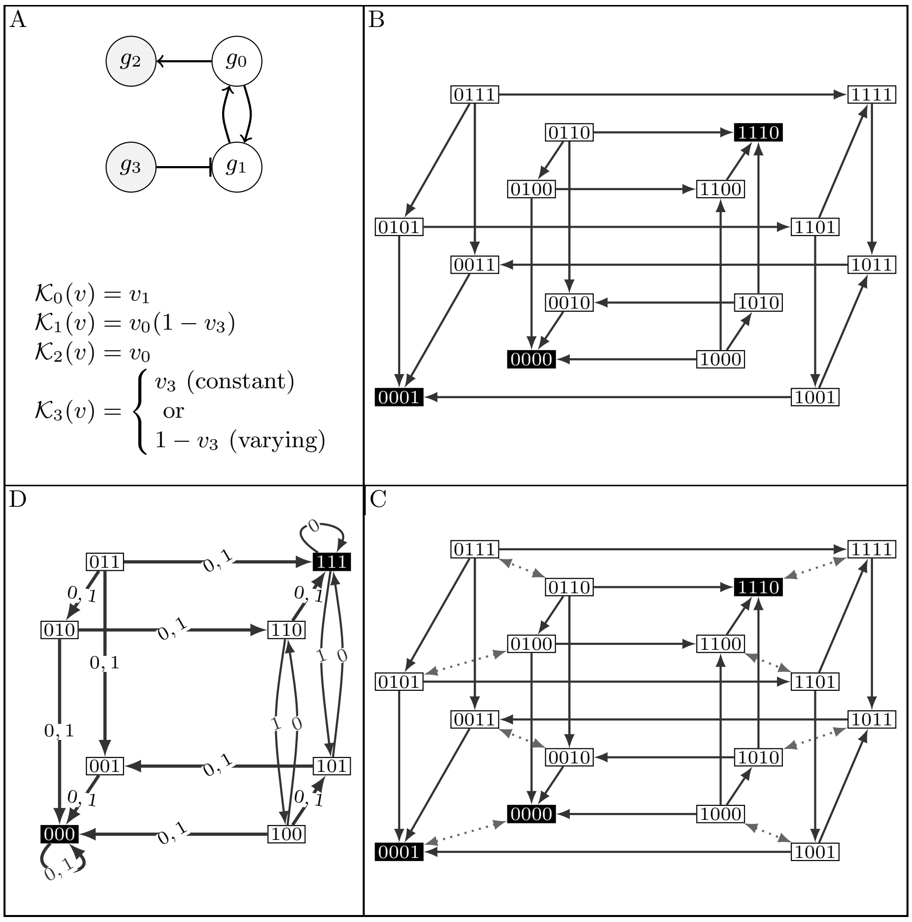

Fig. 1. (A) Simple example of a (Boolean) LSRG, with two proper components, one input and one output, and the associated logical functions. **(B)** The corresponding (full) STG $\mathcal{E}_{cste}$, when the input component g_3 is kept constant (states denote the values of g_0, g_1, g_2 and g_3 in this order). There are 3 stable states. **(C)** The (full) STG $\mathcal{E}_{var}$, considering a free variation of g_3. **(D)** The labelled STG $\mathcal{E}^{|\{g_3\}}$, resulting from the projection over g_3 and the labelling of the transitions with its values (see Section 3.3).

2. a transition $(u, v) \in \mathcal{T}$ is not preserved (*i.e.*, $(\pi^r(u), \pi^r(v)) \notin \mathcal{T}^r$ and $\pi^r(u) \neq \pi^r(v)$) iff the following conditions are fulfilled:
   - u is not a representative state ($u_r \neq \mathcal{K}_r(u)$),
   - $\exists i \neq r$, $v_i \neq u_i$, *i.e.*, u and v differ on their values for a component g_i, which is not g_r, hence $(u, v) \in \mathcal{T}$ is a transition over g_i,
   - and $\Delta_i(u) \neq \Delta_i(s^r(\pi^r(u)))$ (the updating call on g_i in state u is not preserved in the representative state $s^r(\pi^r(u))$);
3. Stable states in $\mathcal{E}$ are conserved in $\mathcal{E}^r$: u stable in $\mathcal{E}$ implies that u is a representative state and $\pi^r(u)$ is stable. Moreover, if z is stable in $\mathcal{E}^r$, then $s^r(z)$ is stable in $\mathcal{E}$;

4. If $(u_1 \dots u_p)$ is a elementary terminal cycle in $\mathcal{E}$ then $(\pi^r(u_1) \dots \pi^r(u_p))$ is a elementary terminal cycle in $\mathcal{E}^r$;
5. If $C \in \mathcal{S}$ is a complex attractor in $\mathcal{E}$, $\pi^r(C)$ contains at least one complex attractor in $\mathcal{E}^r$.

Summarising, stable states and elementary attractive cycles are preserved by the reduction (they only occur for constant input components). Complex attractors may appear as the result of an SCC disconnection provoked by the loss of transitions. Note that, all transitions over varying input components are preserved in $\mathcal{T}^r$ (since, for any state $v \in \mathcal{S}$, they equally exist in $\mathcal{T}$ for all states in the equivalence class $[v]_{\sim r}$).

3 Focusing on Core Networks in LSRGs

In this section, based on topological considerations, we define three set of components, each playing a distinct role in the dynamics of a LSRG. We then describe the reduction of output cascades and projection over input components as relevant means to reduce the size of the dynamics, yet keeping all its properties.

3.1 Splitting the Set of Components into Three Relevant Subsets

Given an LSRG $\mathcal{R} = (\mathcal{G}, \mathcal{K})$, its set of components is defined as the union of the set of inputs $\mathcal{I}$, the set of output $\mathcal{O}$ and the set of proper components $\mathcal{P}$. Here, still on topological considerations, we define another partition of $\mathcal{G}$ in three sets that play different role in the emergence of the dynamical properties.

The *set of input and pseudo-input components*, denoted $\widetilde{\mathcal{I}}$ is recursively defined as follows:

– $\mathcal{I} \subset \widetilde{\mathcal{I}}$ (all input components are in $\widetilde{\mathcal{I}}$);
– $\forall g_i \in \mathcal{G}$, if $Reg(g_i) \subset \widetilde{\mathcal{I}}$ then $g_i \in \widetilde{\mathcal{I}}$ (if all regulators of g_i are inputs or pseudo-inputs, then g_i is a pseudo-input).

Similarly, the *set of output and pseudo-output components*, denoted $\widetilde{\mathcal{O}}$, is defined as follows:

– $\mathcal{O} \subset \widetilde{\mathcal{O}}$ (all output components are in $\widetilde{\mathcal{O}}$);
– $\forall g_i \in \mathcal{G}$, if $\forall g_k \in \mathcal{G}$ s.t. $g_i \in Reg(g_k)$ we have $g_k \in \widetilde{\mathcal{O}}$, then $g_i \in \widetilde{\mathcal{O}}$ (if all targets of g_i are outputs or pseudo-outputs, then g_i is a pseudo-output).

Finally, *Core*, the *set of core components* of a LSRG is defined as the set of components that are neither in $\widetilde{\mathcal{I}}$ nor in $\widetilde{\mathcal{O}}$:

$$Core = \mathcal{G} \setminus (\widetilde{\mathcal{I}} \cup \widetilde{\mathcal{O}})$$

When input components (elements of $\mathcal{I}$) are maintained constant, for any attractor made up of a set of states A, we have: $\forall g_i \in \widetilde{\mathcal{I}}$, $\forall v, v' \in A$, $v_i = v'_i$ (pseudo-input components are stable).

Moreover, while input and pseudo-input components transmit external stimuli to the core components, these drive the dynamics of output and pseudo-output components. We also refer to the sets $\widetilde{\mathcal{O}}$ as output cascades, and $\widetilde{\mathcal{I}}$ as input cascades.

3.2 Reduction of Output Cascades

Since an output has no effect on other components, output components can be removed from a LSRG, with no impact on the behaviour. This is formalised by the following property.

Property 1. Let $\mathcal{R} = (\mathcal{G}, \mathcal{K})$ a LSRG, $g_r \in \mathcal{O}$ an output component of $\mathcal{R}$ and $\mathcal{E} = (\mathcal{S}, \mathcal{T})$ the associated STG. Then $\mathcal{E}^r = (\mathcal{S}^r, \mathcal{T}^r)$ the STG of the LSRG $\mathcal{R}^r$ resulting from the reduction of g_r, verifies:

$$\forall u, v \in \mathcal{S},\ (u, v) \in \mathcal{T} \Longrightarrow (\pi^r(u), \pi^r(v)) \in \mathcal{T}^r \text{ or } \pi^r(u) = \pi^r(v), \tag{1}$$

$$\forall x, y \in \mathcal{S}^r,\ (x, y) \in \mathcal{T}^r \Longrightarrow (s^r(x), s^r(y)) \in \mathcal{T}. \tag{2}$$

Proof. We only prove the first point that corresponds to the preservation of all the transitions, the proof of Eq. 2 can be found in [11], lemma 1. Let us consider $(u, v) \in \mathcal{T}$, then,

– if transition (u, v) involves g_r (the reduced component), u and v are in the same equivalence class $[u]_{\sim r} = [v]_{\sim r}$ (they only differ in their values of g_r), therefore their projection is equal: $\pi^r(u) = \pi^r(v)$;
– if transition (u, v) involves $g_i \in \mathcal{I}$, an input component, which freely varies, then, this transition is obviously preserved: $(\pi^r(u), \pi^r(v)) \in \mathcal{T}$;
– if transition (u, v) involves $g_i \in \mathcal{P} \cup \mathcal{O}$, a proper or an output component (different from g_r), then $v_i = \mathcal{K}_i(u) + \Delta_i(u)$ and, because g_r is an output component, we have also $\mathcal{K}_i^r(\pi^r(u)) = \mathcal{K}_i(u)$, hence this transition is preserved: $(\pi^r(u), \pi^r(v)) \in \mathcal{T}$.

As a consequence, attractors are fully preserved in $\mathcal{E}^r$, including complex ones, and more than that, reachability of these attractors is conserved. This follows from the fact that a path in $\mathcal{E}$ is preserved if the reduction preserves all its transitions (see [11]). We say that the reduction of an output component is *lossless*.

As mentioned before, output components often serve as readouts of a model. Hence it is important to retrieve their values (typically in a stable state or in a complex attractor). This is easily done because the behaviour of an output component is fully described by the sole representative states in the original STG. Hence retrieving the behaviours of output components only requires to store their logical functions (see Section 4).

Since the reduction of an output component is lossless, it is obviously also the case for the reduction of several output components.

Following the reduction of an output component, some of its former regulators become output components. These are the pseudo-outputs (that only regulate outputs or other pseudo-outputs). As such pseudo-outputs become outputs after reduction of their targets, they can be reduced as well, still preserving the dynamical behaviour. Therefore, the reduction of the whole set $\widetilde{\mathcal{O}}$ of output and pseudo-output components does not affect the dynamics.

To be able to recover the values of the components in $\widetilde{\mathcal{O}}$, we need to keep trace of the reduction of pseudo-outputs, redefining the logical functions of the targets of previously reduced components (see Section 4).

3.3 Input Components

Regulatory functions of proper and output components define their behaviours, depending on the current state of their regulators. An input component has no regulator and its function is thus assumed to be either constant or to freely vary as specified in Definition 1. For a LSRG with constant input values, its STG is composed by a set of disconnected graphs, one for each combination of input values (see Figure 1, panel B). Given an initial value of all the input variables, the behaviour is thus restricted to a sub-graph, easing the analysis of large systems. However, when input components freely vary (*i.e.*, are under no specific control), the STG encompasses transitions between all the states that have the same values of the internal components, denoting the sole changes of the input components. Considering that states are characterised by proper and output components values, the whole behaviour can be represented by a STG, with transitions labelled by the values of the input variables, yielding a compacted, labelled STG (see Figure 1). Below, we define such a projection over the input variables.

Definition 3. *Given $\mathcal{R} = (\mathcal{G} = \mathcal{I} \cup \mathcal{P} \cup \mathcal{O}, \mathcal{K})$, a LSRG and $\mathcal{E} = (\mathcal{S}, \mathcal{T})$ its STG. The corresponding* labelled STG *$\mathcal{E}^{|\mathcal{I}} = (\mathcal{S}^{|\mathcal{I}}, \mathcal{T}^{|\mathcal{I}})$ is defined as follows:*

- $\mathcal{S}^{|\mathcal{I}} = \prod_{g_i \in \mathcal{P} \cup \mathcal{O}} D_i$,
- $\forall v^{|\mathcal{I}}, w^{|\mathcal{I}} \in \mathcal{S}^{|\mathcal{I}}$, $(v^{|\mathcal{I}}, L, w^{|\mathcal{I}}) \in \mathcal{T}^{|\mathcal{I}}$ *iff* $\exists v, w \in \mathcal{S}$ *with* $(v, w) \in \mathcal{T}$ *such that* $\forall g_i \in \mathcal{P} \cup \mathcal{O}, v_i^{|\mathcal{I}} = v_i$ *and* $w_i^{|\mathcal{I}} = w_i$*, with L the label of this transition defined as the set of all the values of the input components for which this transition is observed in E:*

$$L = \{u \in \Pi_{g_i \in \mathcal{I}} D_i \text{ s.t. } \forall g_i \in \mathcal{I},\, v_i = u_i (= w_i)\}\,.$$

When a LSRG has a significant number of input components, this representation may presents a true gain in the number of states ($\Pi_{g_i \in \mathcal{G} \setminus \mathcal{I}} |D_i|$ instead of $\Pi_{g_i \in \mathcal{G}} |D_i|$), still keeping all the information regarding the input components. Such a graph structure combining labels on both states and transitions is already used by the formal verification community, and is called a Kripke Transition System (KTS) [7].

By definition, stable states in a STG have no output transitions (see Figure 1, panel B). However, when using model checking techniques, states of the system to be verified must give rise to at least one transition. Notably, a self-loop must be added to each stable state when translating the system into a KTS (*e.g.*, the implemented export to NuSMV). This is particularly useful to represent labelled STGs (Definiton 3).

Definition 4. *Given a labelled STG $E^{|\mathcal{I}} = (S^{|\mathcal{I}}, T^{|\mathcal{I}})$, a state $v \in S^{|\mathcal{I}}$ is:*

a strong stable state *iff* $\forall w \in S^{|\mathcal{I}}, \forall L \in \Pi_{g_i \in \mathcal{I}} D_i, w \neq v \Rightarrow (v, L, w) \notin T^{|\mathcal{I}}$,
a weak stable state *iff* $\forall w \in S^{|\mathcal{I}}, \exists L \in \Pi_{g_i \in \mathcal{I}} D_i, w \neq v \Rightarrow (v, L, w) \notin T^{|\mathcal{I}}$.

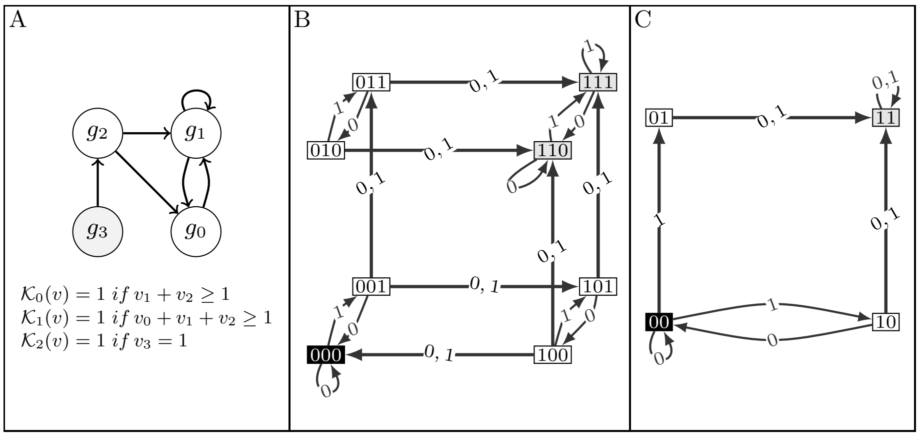
A
B
C
g_2
g_1
g_3
g_0
$\mathcal{K}_0(v) = 1\ if\ v_1 + v_2 \geq 1$
$\mathcal{K}_1(v) = 1\ if\ v_0 + v_1 + v_2 \geq 1$
$\mathcal{K}_2(v) = 1\ if\ v_3 = 1$
011
111
010
110
001
101
000
100
01
11
00
10

Fig. 2. **(A)** Simple example of a (Boolean) LSRG, with two proper components, an input and a pseudo-input, and the associated logical functions. Here $\widetilde{\mathcal{I}} = \{g_3, g_2\}$. **(B)** The labelled STG $\mathcal{E}^{|\{g_3\}}$, resulting from the projection over the input component g_3 and the labelling of the transitions with its values. The states 000, 110 and 111 are weak stable states. The last two states constitute a strong stable core ensemble (see text). **(C)** The labelled STG of the model where g_2 has been reduced. Here, the previous core ensemble {110, 111} can be recovered from the strong stable state 11.

These definitions are sufficient for fixed input components and also cover all the cases observed in our toy example (Figure 1, panel D), and in the T cell activation model described in Section 5. They are however not sufficient in other situations, where signalling cascades do vary upon input variations and the "core" network remains stable. This is illustrated in Figure 2.

It is thus necessary to better classify stable patterns, and identify what we would call stable core patterns (see states 110 and 111 in Figure 2). To assess pattern stability over input component variations, we rely on the behaviour of the components in *Core*, the set of components that belong to the core network (see Section 3.1).

Let $Stable \subset \mathcal{S}$ be the set of stable states for constant input components: $Stable = \{v \in \mathcal{S}, \mathcal{K}_i(v) = v_i, \forall i \in \mathcal{G}\}$. It is worth noting that GINsim provides a very efficient algorithm to determine this set [9,12]. Then, for all $v \in Stable$, we define $Core(v) = \{v' \in Stable, \forall g_i \in Core, v'_i = v_i\}$, the set of stable states that have the same values for the core components. Then, for varying inputs, we classify the stable states as follows,

- if $|Core(v)| = 1$, then v is a *weak stable state* (there is a unique input configuration for which this state is stable);
- if $\forall v' \in Core(v), \forall g_i \in \mathcal{P}, v_i = v'_i$, then v is a *strong stable state* (since all these stable states only differ in their input component values);
- otherwise ($|Core(v)| > 1$ and $\exists v' \in Core(v), \exists g_i \in \mathcal{P} \setminus Core, v_i \neq v'_i$), v defines what we could call a *stable core ensemble.* Then, similarly to the stable states, we could define *strong stable core ensembles* (such that $|Core(v)|$

equals the number of configurations of the input values) and *weak stable core ensembles*.

Note that if $v' \in Core(v) \subset Stable$, states v and v' necessarily share the same values on their output (and pseudo-output) components.

Another method to assess the stability of patterns upon input variations, would consist in reducing the input cascades (iteratively all the pseudo-input components in $\widetilde{\mathcal{I}} \setminus \mathcal{I}$). Then, the strong stable states and core ensembles of the original model are recovered from the (strong) stable states of this reduced model (see Figure 2, panel C). Similarly, weak stable states and core ensembles are recovered from the weak stable states of the reduced model. However, it is important to recall here that this reduction, although it conserves the number of stable states, could modify their reachability. This method is rather similar to that described by Saadatpour *et al.* who, for constant input values, consider the components that will reach stable values and propose to reduce them [15].

Although such considerations on stability upon input variations could also apply to complex attractors, we leave this extension for future work.

4 Implementation

The software GINsim is dedicated to the definition and analysis of logical models [9]. It provides, among a variety of functionalities, the reduction method presented in Section 2 [11]. Here, we briefly describe implementation aspects of the methodology presented in this paper. First, the reduction of the output cascades is implicitly made in GINsim. The model is then exported to be verified using the model-checker NuSMV. A new stable release of GINsim is expected in the near future. Meanwhile, a beta version of the tool with these new functionalities is available, along with supplementary material, at `http://compbio.igc.gulbenkian.pt/nmd/node/46`.

4.1 Output Nodes Manipulation in GINsim

GINsim [9] has been extended to automatically annotate output nodes based on the structure of the LSRG. We have added an internal method that identifies the pseudo-outputs and turns them into output components. For this, we apply the reduction method to remove references to pseudo-outputs from the logical functions of their targets. This trick ensures that all outputs can be defined as depending only on core components, their values can thus be computed independently. This is supported by the argument below.

Considering g_i an output and g_j a pseudo-output regulating g_i ($g_j \in Reg(g_i)$). $\mathcal{K}_j$, the logical function of g_j, is not modified, whereas $\mathcal{K}_i$, the logical function of g_i, is replaced by $\mathcal{K}_i^j$ its new function obtained by the reduction of g_j.

Note that, for the time being, the LSRG obtained through this manipulation is used only for the NuSMV export, where outputs are defined as macros, not characterising a state.

4.2 NuSMV Model Encoding and Verification

In [8], we have implemented an export functionality in the context of GINsim, making possible the use of the NuSMV symbolic model-checker [2]. This NuSMV model description enabled the consideration of the following updating policies: synchronous, asynchronous or priority classes. This export also permitted to distinguish between input and non-input components, enabling the reduction of the state space over input components (described in section 3.3).

Here, this GINsim export functionality is extended with the capability to distinguish between proper and output components (Section 3.2), containing only the logical rules governing the proper components. The input valuations are still projected over the transitions where they are valid, and the output valuations are computed as macros depending on the state (defined only by the proper components), permitting a reduction of the state space proportional to the number of (pseudo-)output components. Pseudo-outputs are tackled through the previously discussed method: from the NuSMV export point of view, they are treated just like outputs.

As mentioned earlier, to be able to represent information both on states (proper components) and on transitions (input components), we consider a graph structure known in the formal verification community as Kripke Transition System (KTS) [7]. The main version of NuSMV supports the verification of temporal logic properties over KTSs, but only when these properties do not make a reference to input components, *i.e.*, do not impose restrictions on the transitions of the KTS. This version only permit us to verify properties considering varying inputs, with no query on their values.

In order to perform verifications over KTSs with properties imposing restrictions on transitions, we consider a particular NuSMV extension, denoted NuSMV-ARCTL-TLACE [6]. This extension supports the ARCTL temporal logic [13], an extension of CTL [3], which adds action-restricted operators through an additional argument allowing the specification of restrictions on transitions over KTSs. Through the use of the ARCTL temporal logic, it is then possible to perform verifications over reduced models of signalling-regulatory networks, considering restrictions on their input components. These restrictions thus allow the verification of a property with specific (combinations of) values of the input components, or a chaining of CTL and ARCTL operators to verify reachability properties over KTS paths with unrestricted and specific input values, respectively.

A CTL temporal logic operator accepts as argument a set of restrictions over non-input components, *e.g.*, `EF(var1 & !var2)`, exploring each transition independently of the value of the input components. On the other hand, an ARCTL temporal logic operator accepts an additional argument defining a restriction over input components, *e.g.*, `EAF(inp4 & !inp7)(var1 & !var2)`, exploring only the transitions satisfying `inp4 & !inp7` (without imposing restrictions on other input components that might exist).

5 Application to the Model of T Cell Activation

T cells (or T lymphocytes) are immune cells reacting to the presence of specific antigens in the organism and playing a key role in the selection of the immune response. The T cell family is divided into several subfamilies, each with a specific role. Their antigen specificity arises from a randomly-generated membrane receptor: the T cell receptor (TCR). New T cells are continuously generated, then undergo a selection mechanism to avoid self-immunity before circulating in the organism. In the absence of their specific antigen, these cells will enter apoptosis after a few days. However, when it encounters its specific antigen, a T cell is activated and elicits the corresponding immune response. This activation is triggered by binding of the antigen on their T cell Receptor (TCR). Cells are kept alive as long as they receive TCR stimulation. The TCR activation pathway is thus a crucial part of the initiation and maintenance of a specific immune response.

The TCR recognises specific antigens (peptides) associated to the Major Histocompatibility Complex (MHC) on the surface of Antigen Presenting Cells. This recognition also involves TCR co-receptor (CD4 for T helper cells, CD8 for cytotoxic T cells) and is accompanied by CD28 co-stimulation. Saez-Rodriguez *et al.* [17] proposed a comprehensive logical model of the TCR activation pathway, taking into account the CD4 co-receptor and the action of the CD28 co-stimulatory molecule. This TCR activation model encompasses 94 components, including 35 proper components (see Figure 3).

The dynamical analysis performed in [17] focuses on short-term effects thus studying the states reached by signal propagation. This is done by impeding the feedback loops to function through the definition of slow events. Here, we are interested in the more complex behaviours that arise when feedback loops are taken into account. Firstly, we have defined a GINsim version of this model and confirmed that we obtain the same stable states in absence of the feedback loops, both in the wild-type condition and after applying model perturbations.

For the full model (with feedback loops), we observe that stable states only exist in the absence of TCR stimulation (with or without CD4 and CD28). Indeed, the TCR signal triggers oscillations, embodied in complex attractors. Note that such oscillations will be transient by nature as the TCR signal is not a stable stimulation. In what follows, we start by identifying the attractors (stable states in the absence of TCR signal and oscillations when TCR is present) using GINsim capabilities. We then resort to model checking to gain further insights into long term behaviours upon changes of the input signals; in particular, switches from one attractor to another.

5.1 Attractor Identification

To study the reachability of attractors by model-checking, we first need to identify these attractors, which is easy for stable states but more challenging for complex attractors as shown hereafter.

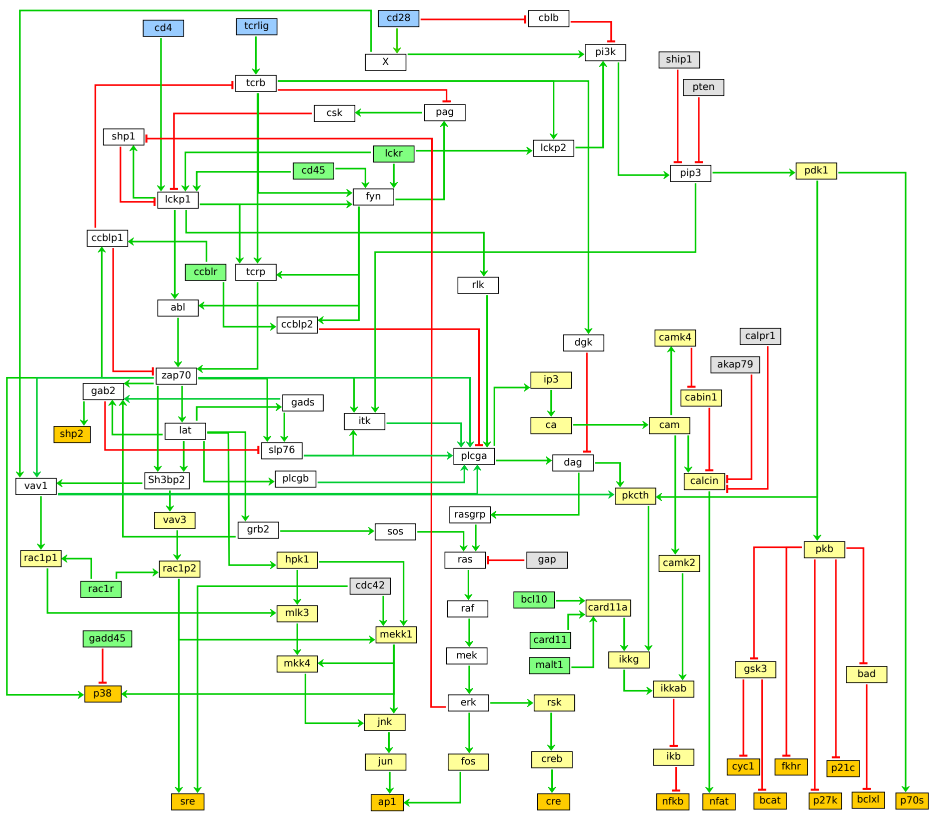

Fig. 3. TCR activation model proposed in [17]. This model encompasses 94 components: 3 inputs (blue), 14 fixed components (green for active and grey for inactive), and 14 outputs (orange), 28 pseudo-outputs (yellow), 35 proper components (white). Green arrows denote activations, while red T-arrows denote inhibitions.

Using the stable state search tool available in GINsim [12], we have identified all the stable states and, using the STG construction, we have checked their reachability from the initial state defined in [16]. This initial state comprises all components set to zero, except the three repositories `lckr`, `ccblr`, `rac1r` and five fixed components `cd45`, `gadd45`, `bcl10`, `card11` and `malt1`.

We have then identified the complex attractors by performing simulations in the presence of the TCR ligand. In this case, the computation of the STG is not tractable on the full model, even after the reduction of the output cascades. We have thus considered a further reduced model in which 10 internal components have been manually selected for reduction (aiming for a minimal impact of this reduction on the dynamical behaviour, we selected: `cblb`, `cblbp1`, `cblbp2`, `sos`, `lckp1`, `lckp2`, `mek`, `raf`, `ras`, and `X`). We recall that this reduction of core components ensures that no attractor can be lost, but may impede their reachability. We thus use this technic to identify complex attractors before using model-checking on the full model to assess their reachability. Using this simplified model, we could identify its complex attractors. Starting from states within these sets of states, we could further refine the descriptions of these attractors for the full model (without output cascades, though).

Table 1 illustrates the identified attractors. We named these attractors `SSxxx` or `SCCxxx` to represent weak stable states or strongly connected components, respectively. Additionally, we specified the name of each attractor according to the (set of) input combination(s) for which each attractor is stable, following the order `cd28`, `cd4` and `tcrlig`.

The set of states composing each complex attractor is covered by a schema, or pattern (see Table 1). For each complex attractor, we specify an ARCTL property and use the NuSMV-ARCTL-TLACE model-checker to verify that, for each of these patterns, no state covered by the pattern enables a transition leaving that set of states. The following property exemplifies the case of the pattern `SCC001` corresponding to complex attractor that arises in the absence of both `cd28` and `cd4` and the presence of `tcrlig`: `INIT SCC001; SPEC EAF(!cd28 & !cd4 & tcrlig)(!SCC001)`. All the equivalent properties defined for each complex attractor (see Supplementary files) returned `false`, ensuring that each pattern indeed captures the terminal SCC, containing at most some states belonging to its (strict) basin of attraction. The patterns describing the attractors given in Table 1, show that the weak stable state `SS0*0`, where `cd28` is absent, is part of the complex attractor `SCC001`, which in turn is part of the complex attractor `SCC011`. The analogous occurs when `cd28` is present in the case of the remaining attractors.

In the attractor summary of Table 1, considering the wild-type condition, we can observe that the `tcrlig` input variable is responsible for the existence (resp. absence) of oscillations, whenever it is present (resp. absent). This is confirmed by a circuit analysis, which provides the conditions under which the existing negative circuits are functional (see [12,14,19]). Here, the main functional negative circuit, `zap70 ccbblp1`, depends on the presence of `tcrp`, which in turn depends on the presence of `tcrb`, which is directly controlled by the value of the `tcrlig` input variable. The other negative circuit, `shp1 lckp1`, depends on the absence of `csk`, therefore on the presence of `tcrb`, but also directly depends on the `cd4` input variable.

While the other input components alone are not capable to trigger oscillations, we can note that in absence of `tcrlig`, `cd4` has no effect on the stable state identity, while `cd28` changes the activity of some proper and output components. In particular, `cd28` activates `pkb`, which triggers anti-apoptotic signals. This role of `cd28` is maintained in the presence of `tcrlig`, while `cd4` increases the number of oscillating components, dramatically increasing the number of states in the attractor.

If we consider more closely the activation pattern of some crucial output components, we can observe that `ap1`, `nfat` and `nfkb` share a pattern: their activation requires both `tcrlig` and one of the other inputs (`cd4` or `cd28`). This result is consistent with the fact that the TCR activation is crucial for the activity (through `nfat`) and survival (through `nfkb` and `pkb` in presence of `cd28`) of T cells.

Table 1. List of patterns for the wild-type, for the single mutant *Δ*`fyn` and double mutant *Δ*`fyn-lckr`. Two weak stable states (`SS0*0` and `SS1*0`) are shared by all conditions: wild-type, single *Δ*`fyn` mutant and double *Δ*`fyn-lckr` mutant. Light and dark grey cells, highlight the values for the proper and output variables, which depend on `cd28` and `cd4` input variables, respectively. The `tcrlig` input variable discriminates between the weak stable states and the complex attractors. Some proper and output variables are omitted for sake of space, their values being easily deduced from those listed in the table.

Conditions		Input			Internal																											Output				
		cd28	cd4	tcrlig	abl	ccblb	ccblp1	ccblp2	csk	dag	dgk	erk	fyn	gab2	itk	lat	lckp1	lckp2	mek	pag	pi3k	pip3	plcga	ras	shp1	slp76	tcrb	tcrp	vav1	X	zap70	ap1	jnk	nfkb	nfat	pkb
All	SS0*0	0	*	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	SS1*0	1	*	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	0	0	1	0	0	1
Wild-type	SCC001	0	0	1	*	1	*	*	*	0	*	0	*	*	0	*	0	*	0	*	0	0	0	0	0	*	*	*	*	0	*	0	*	0	0	0
	SCC011	0	1	1	*	1	*	*	*	*	*	*	*	*	0	*	*	*	*	*	0	0	*	*	*	*	*	*	*	0	*	*	*	*	*	0
	SCC101	1	0	1	*	0	*	*	*	*	*	*	*	*	*	*	0	*	*	*	1	1	*	*	0	*	*	*	1	1	*	*	*	*	*	1
	SCC111	1	1	1	*	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1	1	*	*	*	*	*	*	1	1	*	*	*	*	*	1
ΔFyn	SS001	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	SS101	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0	1
	SCC011	0	1	1	*	1	*	0	*	*	*	*	0	*	0	*	0	*	*	*	0	0	*	*	*	*	*	*	*	0	*	*	*	0	*	0
	SCC111	1	1	1	*	0	*	0	*	*	*	*	0	*	*	*	*	*	*	*	1	1	*	*	*	*	*	*	1	1	*	*	1	*	*	1
ΔFyn *ΔLck*	SS0*1	0	*	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	SS1*1	1	*	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0	1

We further performed the attractor search for a given set of perturbations and we observed that some of the complex attractors are replaced by stable states in the *Δ*`fyn` single mutant, as well as in the *Δ*`fyn-lckr` double mutant (see Table 1). In the *Δ*`fyn` single mutant, we observe that most of the proper components that underwent oscillations in the wild-type condition are now fixed at zero in the stable states `SS001` and `SS101`, while a small subset of them becomes tightly dependent on the presence of `tcrlig` (`dgk`, `lckp2` and `tcrb`). However, the remaining complex attractors `SCC011` and `SCC111` differ from the wild-type condition by having `fyn` and `ccblp2` not expressed, and `lckp1`, `jnk` and `nfat` dependent on the presence of `cd28`. The *Δ*`fyn-lckr` double mutant, additionally prevents the expression of `lckp2`, making the attractors `SS001` and `SS101` no longer dependent on the absence of `cd4`. It is worth remembering that Saez-Rodriguez *et al.* [17] performed the mutant simulations focusing only on "slow" events, thus breaking the feedback loops, which allows the activation of `pkb`. In this paper, considering the full model, we observe that `pkb` becomes tightly dependent on the presence of `cd28` due to the influence of `cblb` upon `pi3k`.

5.2 Reachability Analysis

After the identification and characterisation of the attractors, we have analysed their reachability, assessing which input conditions permit to reach (or leave) each attractor.

This is done by first encoding the patterns given in Table 1 in the NuSMV model description. The characterisation of the complex attractors considers only a restriction on the fixed variables (described by the corresponding pattern). Then, for each of the attractors, we specify a set of ARCTL temporal logic reachability properties, testing the existence of a path from each stable/complex attractor to every other attractor, for all the combinations of (varying or fixed) input components. These combinations of input components are obtained by fixing some of them using ARCTL temporal operators, possibly leaving the others to freely vary (see Supplementary files).

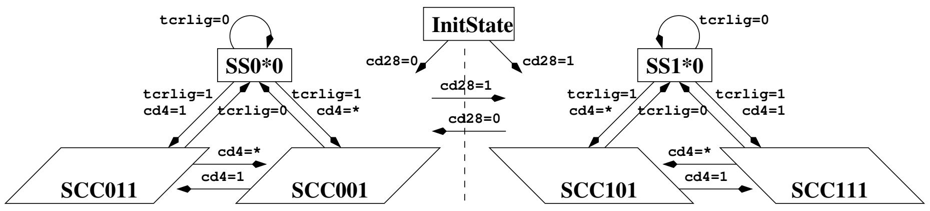

Fig. 4. State space characterization, of the necessary input conditions to switch between the stable and complex attractors specified in Table 1, with respect to the input variables valuation. It is worth noting that the `SS0*0` stable state is included in the `SCC001` complex attractor, and this is itself included in the `SCC011` complex attractor. The transitions between these attractors are then dependent on value restrictions of the `cd4` and `tcrlig` input variables. The analogy is valid for the `SS1*0`.

Figure 4 presents the verification results, indicating the necessary input conditions to switch between attractors. First, we confirm that the input component `cd28` divides the state space in half, setting the dynamics to focus on one group of attractors or the other. These are mirroring each other, where each is composed by a stable state and two complex attractors. Like previously mentioned, the presence of `tcrlig` controls the exit from a stable state towards its corresponding complex attractor and vice-versa. Finally, within each group of complex attractors, the presence (resp. absence) of the `cd4` input variable allows (resp. restricts) the dynamics to evolve to a larger (resp. smaller) set of states, `SCC011` or `SCC111` (resp `SCC001` or `SCC101`).

6 Conclusions and Prospects

The analysis of qualitative models of large signalling-regulatory networks is hampered by a combinatorial explosion of their state spaces. This is particularly true when properties of interest relate to reachability that often requires extensive search of the state transition graphs. Here, we propose to lessen this problem by a specific handling of input and output components. For the input components, their values are taken into account by proper labels on the transitions. This leads to a significant reduction when the model encompasses a large number of input components (as it is the case, for instance, in the model accounting for T cell differentiation defined in [10], which includes 13 inputs for a total number of 65 components). Furthermore, we used the reduction method as defined in [11] to get rid of output components and of what we called pseudo-output components. We prove that this reduction is lossless, in the sense that it preserves all the attractors and their reachability.

We aim at using the methodology presented here to revisit the T cell differentiation model [10]. In particular, we can now systematically analyse the impacts of input variations on attractor switches (accounting for a possible plasticity of the T cells), as well as mutant conditions.

In the near future, we will make the reduction of output cascades fully functional in GINsim. More precisely, upon user request, output cascades will be reduced and made implicit, STGs will be computed disregarding the corresponding variables, which values will be possibly recovered for given set of states (*e.g.*, in attractors).

In Section 3.3, we have discussed the determination of stable patterns, including in the case of varying input components. When inputs freely vary, pseudo-inputs also vary, but these variations may not affect the stability of the core network. We thus introduced the notions of (strong or weak) stable states and stable core ensembles. While GINsim implements an efficient algorithm to identify all stable states (for constant input components), we still need to delineate a method to determine the complex attractors. Indeed, for large models (as it is the case for the TCR model revisited in Section 5), the full characterisation of all the complex attractors is often difficult or even intractable. We then could extend the concepts of strong and weak stability to these complex attractors.

Ackowledgements. We acknowledge the support from the Portuguese FCT (PM post-doctoral fellowship SFRH/BPD/75124/2010 and project grant PTDC/EIACCO/099229/2008). We are grateful to D. Thieffry and E. Remy for inspiring discussions. The first version of the TCR model in GINsim was defined by D. Thieffry. Finally, we thank E. Mohr for her help to interpret the results concerning the TCR model.

References

1. Bilke, S., Sjunnesson, F.: Stability of the Kauffman model. Phys. Rev. E 65(016129) (2001)
2. Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 359–364. Springer, Heidelberg (2002)
3. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. 8, 244–263 (1986)
4. de Jong, H.: Modeling and simulation of genetic regulatory systems: a literature review. J. Comput Biol. 1(9), 67–103 (2002)

5. Fauré, A., Naldi, A., Chaouiya, C., Thieffry, D.: Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. Bioinformatics 22(14), 124–131 (2006)
6. Lomuscio, A., Pecheur, C., Raimondi, F.: Automatic verification of knowledge and time with nusmv. In: Veloso, M.M. (ed.) Proc. 20th Intl. Joint Conf. on Artificial Intelligence (IJCAI 2007), pp. 1384–1389. Morgan Kaufmann Publishers Inc. (2007)
7. Müller-Olm, M., Schmidt, D., Steffen, B.: Model-Checking: A Tutorial Introduction. In: Cortesi, A., Filé, G. (eds.) SAS 1999. LNCS, vol. 1694, pp. 330–354. Springer, Heidelberg (1999)
8. Monteiro, P.T., Chaouiya, C.: Efficient verification for logical models of regulatory networks. In: Rocha, M.P., Luscombe, N., Fdez-Riverola, F., Rodríguez, J.M.C. (eds.) 6th International Conference on PACBB. AISC, vol. 154, pp. 259–268. Springer, Heidelberg (2012)
9. Naldi, A., Berenguier, D., Fauré, A., Lopez, F., Thieffry, D., Chaouiya, C.: Logical modelling of regulatory networks with GINsim 2.3. Biosystems 97(2), 134–139 (2009)
10. Naldi, A., Carneiro, J., Chaouiya, C., Thieffry, D.: Diversity and plasticity of th cell types predicted from regulatory network modelling. PLoS Comput. Biol. 6(9) (2010)
11. Naldi, A., Remy, E., Thieffry, D., Chaouiya, C.: Dynamically consistent reduction of logical regulatory graphs. Theor. Comput. Sci. 412, 2207–2218 (2011)
12. Naldi, A., Thieffry, D., Chaouiya, C.: Decision Diagrams for the Representation and Analysis of Logical Models of Genetic Networks. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 233–247. Springer, Heidelberg (2007)
13. Pecheur, C., Raimondi, F.: Symbolic Model Checking of Logics with Actions. In: Edelkamp, S., Lomuscio, A. (eds.) MoChArt IV. LNCS (LNAI), vol. 4428, pp. 113–128. Springer, Heidelberg (2007)
14. Remy, E., Ruet, P.: From minimal signed circuits to the dynamics of boolean regulatory networks. Bioinformatics 24(16), i220–i226 (2008)
15. Saadatpour, A., Albert, I., Albert, R.: Attractor analysis of asynchronous boolean models of signal transduction networks. J. Theor. Biol. 266(4), 641–656 (2010)
16. Saez-Rodriguez, J., Simeoni, L., Lindquist, J., Hemenway, R., Bommhardt, U., Arndt, B., Haus, U.-U., Weismantel, R., Gilles, E., Klamt, S., Schraven, B.: A logical model provides insights into T cell receptor signaling. PLoS Comput. Biol. 3(8), e163 (2007)
17. Saez-Rodriguez, J., Simeoni, L., Lindquist, J.A., Hemenway, R., Bommhardt, U., Arndt, B., Haus, U.-U., Weismantel, R., Gilles, E.D., Klamt, S., Schraven, B.: A logical model provides insights into T cell receptor signaling. PLoS Comput. Biol. 3(8), e163 (2007)
18. Schlitt, T., Brazma, A.: Current approaches to gene regulatory network modelling. BMC Bioinformatics 8(suppl. 6), S9 (2007)
19. Thieffry, D.: Dynamical roles of biological regulatory circuits. Brief. Bioinform. 8(4), 220–225 (2007)
20. Thomas, R.: Regulatory networks seen as asynchronous automata: A logical description. J. Theor. Biol. 153, 1–23 (1991)