Vaginay *et al. Applied Network Science* (2022) 7:73
https://doi.org/10.1007/s41109-022-00505-8

Applied Network Science

RESEARCH Open Access

# From quantitative SBML models to Boolean networks

Athénaïs Vaginay[1,2*], Taha Boukhobza[1] and Malika Smaïl-Tabbone[2]

*Correspondence:
athenais.vaginay@loria.fr

[1] CNRS, CRAN, Université de Lorraine, 54000 Nancy, France
[2] CNRS, Inria, LORIA, Université de Lorraine, 54000 Nancy, France

**Abstract**

Modelling complex biological systems is necessary for their study and understanding. Biomodels is a repository of peer-reviewed models represented in the Systems Biology Markup Language (SBML). Most of these models are quantitative, but in some cases, qualitative models—such as Boolean networks (BNs)—are better suited. This paper focuses on the automatic transformation of quantitative SBML models to Boolean networks. We propose SBML2BN, a pipeline dedicated to this task. Our approach takes advantage of several SBML elements (reactions, rules, events) as well as a numerical simulation of the concentration of the species over time to constrain both the structure and the dynamics of the Boolean networks to synthesise. Finding all the BNs complying with the given structure and dynamics was formalised as an optimisation problem solved in the answer-set programming framework. We run SBML2BN on more than 200 quantitative SBML models, and we provide evidence that one can automatically construct Boolean networks which are compatible with the structure and the dynamics of an SBML model. In case the SBML model includes rules or events, we also show how the evaluation criteria are impacted when taking these elements into account.

**Keywords:** Boolean networks, Model transformation, SBML, Systems Biology

## Introduction

Life is based on biological systems that are essentially composed of biological species (molecules such as genes, proteins, metabolites) acted upon by processes. They are highly complex because species abundances and interactions change over time in response to external stimuli, as well as to dynamical intra-system processes. Throughout this article, the biological system that serves as a running example is an enzymatic process, which consists of three reactions: an enzyme E reversibly binds to a molecule of substrate S to form the complex C (reactions $\mathcal{R}_{\text{on}}$ and $\mathcal{R}_{\text{off}}$). Then C is transformed into two molecules of a product P, while E returns to its free state (reaction $\mathcal{R}_{\text{cat}}$). The classical chemical notation of this system is:

$$\overbrace{\underbrace{\mathsf{E}+\mathsf{S} \underset{\mathrm{k_{off}}\times[\mathsf{C}]}{\overset{\mathrm{k_{on}}\times[\mathsf{E}]\times[\mathsf{S}]}{\rightleftharpoons}} \mathsf{C}}_{\mathcal{R}_{\mathrm{off}}} \xrightarrow{\mathrm{k_{cat}}\times[\mathsf{C}]} \mathsf{E}+2\cdot\mathsf{P}}^{\mathcal{R}_{\mathrm{cat}}}}^{\mathcal{R}_{\mathrm{on}}} \tag{1}$$

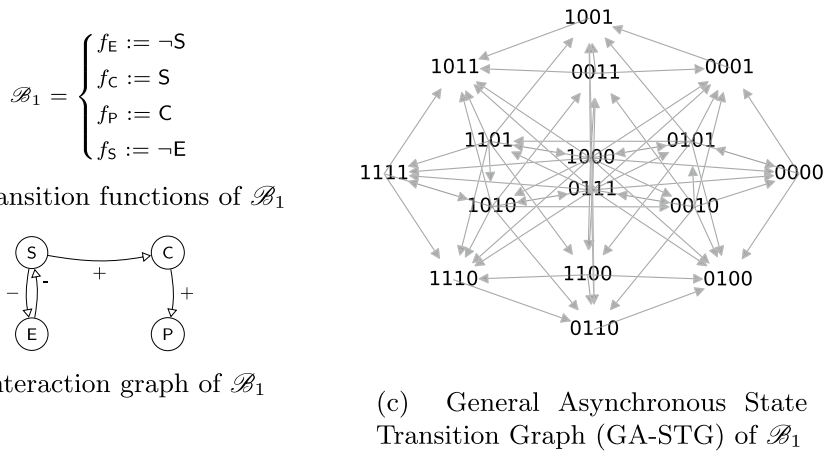
Each of the three reactions is represented by an arrow from the *reactants* (i.e., species consumed during the reaction) to the *products* (i.e., species created during the reaction). Each arrow is annotated with the *speed* of the associated reaction. In our example, the speed of each reaction $\mathcal{R}$ is proportional, with a factor $\mathrm{k}_{\mathcal{R}}$, to the product of the concentration of the reactants (denoted with square brackets).

Dynamical models are used to understand the complex behaviour of biological systems. Such models can use many different computational paradigms and formalisms, ranging from detailed ones (such as ordinary differential equation—ODE) to simple ones (such as Boolean network—BN). Basically, models come in two flavours: quantitative and qualitative. With quantitative models (such as differential equations), we traditionally study concentrations of the species (i.e., values in $\mathbb{R}^+$) over time. For example, we can study how the speed of production of P is affected by the presence of an inhibitor of the enzyme E. With qualitative models (such as Boolean networks), the exact values are abstracted in a finite number of levels (for example present or not, encoded as 1 and 0 respectively), and it is the sequence of configurations that matters.

The Biomodels repository contains a curated collection of over a thousand published models of biological systems (Malik-Sheriff et al. 2020). Models in Biomodels are encoded in the Systems Biology Markup Language (SBML), which is the most widely used standardised representation language in the field of Systems Biology. The SBML representation consists in a set of reactions, rules and events which can be then be interpreted quantitatively using several formalisms to retrieve, for example, concentration of the species over time (with deterministic or stochastic simulation) (Hoops et al. 2006).

Most of the models in Biomodels are quantitative models, and were initially published as ODE models. However, while counter-intuitive, it can be interesting to downgrade the model to a qualitative model such as Boolean networks (Davidich and Bornholdt 2008). First, a simpler model helps its analysis: interpretation of properties such as attractors, for example, is much easier to do qualitatively than quantitatively. Second, ODEs can be difficult to work with, especially when the task is related to model checking, control, or model aggregation. These tasks are well-studies using BNs (Klarner et al. 2020; Biane et al. 2018). Overall, we think that an automatic conversion of a given quantitative SBML model to a (set of) BN(s) on which we can perform qualitative analysis could help to grasp useful insights about the system under study.

In a previous paper published in proceedings of the Complex Networks and Applications (CNA) 2021 conference, we introduced a methodology to synthesise automatically a set of Boolean networks starting from a SBML model (Vaginay et al. 2022). Our approach takes advantage of both structural and dynamical constraints extracted from the SBML model. The structure is retrieved from the reactions set of the model, and the dynamics is obtained with a deterministic simulation of the differential–algebraic system of equations reconstructed from the SBML model. Then, a declarative program constructs, for each species, all the Boolean functions respecting these constraints. The synthesised Boolean networks result from the assembly of these Boolean functions.




(a) Transition functions of $\mathscr{B}_1$

(b) Interaction graph of $\mathscr{B}_1$

(c) General Asynchronous State Transition Graph (GA-STG) of $\mathscr{B}_1$

**Fig. 1** Example of a Boolean network to model Eq. (1)

In this paper, the contributions are twofold: (1) additionally to reactions, we also consider other SBML elements (rules and events) when retrieving the structural constraints from the input SBML model; (2) we investigate the well-formedness of the input model and show how it affects the synthesis of the Boolean networks. Furthermore, we extend the explanation and give the necessary intuitions about the ASP (answer-set programming) encoding used to solve the BN synthesis problem.

The paper is organised as follows. In "Boolean networks and their synthesis" section, we introduce the key notions about Boolean networks and the principles of their synthesis, starting from the given structure and dynamics of the biological system under study. In "Description of the SBML2BN pipeline" section, we present the SBML2BN pipeline and describe each of its steps. Then, we report the evaluation of SBML2BN by running it on more than 200 curated SBML models from the Biomodels database ("Evaluation of the SBML2BN pipeline" section). We also give details on the pipeline implementation, which reuses and extends several published methods and software packages. Finally, we close the paper with conclusions and a few perspectives.

## Boolean networks and their synthesis

### Definitions

Boolean networks (BNs) were introduced by Kauffman (1969) and Thomas (1973) to model genetic regulatory networks. Concepts used in BNs are described in a recent review (Schwab et al. 2020). An example of BN is given in Fig. 1 and used to illustrate the concepts introduced in the following.

The *components* of a BN are the species of the considered biological system. For example, the BN $\mathscr{B}_1$ (Fig. 1) has four components: S, P, E and C. A *configuration* of a BN is a vector that associates a Boolean value ($\mathbb{B} = \{0/\text{inactive}; 1/\text{active}\}$) to each of the $n$ components of the BN (in alphabetical order). For example, in the configuration 0000, no components is active, while only C is active in the configuration 1000. A BN with $n$ components has $2^n$ possible configurations.

Each component X has an associated *transition function* $f_X : \mathbb{B}^n \to \mathbb{B}$ that maps the configurations of the BN to the next value of the component. In this paper, the transition

functions are written as Boolean expressions in Disjunctive Normal Form (DNF), i.e., disjunctions of conjunctions. The conjunctions are *satisfiable*, i.e., they do not contain both a literal and its contrary. Each of the conjunctions composing a DNF is a *1-implicant* of this DNF as it implies that the DNF is True when it evaluates to True. The symbols $\neg$, $\wedge$, $\vee$ represent respectively the negation, conjunction and disjunction. For example, the transition function $f_{\mathsf{C}} := (\mathsf{E} \wedge \neg\mathsf{S}) \vee (\neg\mathsf{E} \wedge \mathsf{S})$ states that the value of C will be 1 if either the value of E or of S was 1 in the previous configuration. Figure 1a shows examples of transition functions with only one term. An implicant is prime whenever removing any literal results in the negation of the original implication. A DNF containing only prime implicants is called a *subset-minimal* DNF. A DNF is *cardinal-minimal* if it is the smallest (with respect to the cardinal of the set of literals appearing in the DNF) subset-minimal DNFs compatible with a partial truth table.

The structure of a BN is defined in terms of parent–child relationships between the components. A component P that appears in the transition function of a component X is called a *parent* of X. If the parent P is negated in the DNF associated with X, we say that the *polarity* of the influence of P on X is negative (noted $-$). Conversely, if the parent is not negated, its influence is positive (noted $+$). In case P has both a positive and a negative influence on X, the influence is non-monotonous (noted $\pm$). The *Interaction Graph* (IG) summarises these relationships as a directed graph. The directed edge $\mathsf{P} \overset{\sigma}{\to} \mathsf{X}$ is labelled with $\sigma \in \{+, -, \pm\}$ depending on the polarity of the influence P has on X. The interaction graph of $\mathscr{B}_1$ (Fig. 1b) contains the edges $\mathsf{S} \overset{-}{\to} \mathsf{E}$ and $\mathsf{S} \overset{+}{\to} \mathsf{C}$ because S appears negatively in the transition function of E and positively in the one of C. We will see in "Synthesis of BNs compatible with a structure and a dynamics" section how the IG is used to define the compatibility of a BN with respect to a given structure.

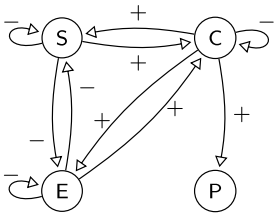
The BN *dynamics* is obtained by applying iteratively the transition functions starting from all possible configurations. The order of application of the transition functions is defined by the *update scheme*. The most common are the *synchronous*, *asynchronous* and *general asynchronous*. In the synchronous update scheme, the transition functions are applied all at once, while in the asynchronous update scheme, they are applied one by one (non-deterministically). In the general asynchronous update scheme, any number of species can be updated at each step. Thus, it includes the updates possibilities of both the synchronous and asynchronous update schemes. The *state transition graph* (STG) is a directed graph whose nodes are the $2^n$ possible configurations of the BN. It contains a directed edge from $c$ to $c'$ if $c'$ is the result of applying on $c$ the transition function(s) according to the chosen update scheme. Figure 1c shows the General-Asynchronous STG (GA-STG) of $\mathscr{B}_1$ (Fig. 1a). We will see in "Synthesis of BNs compatible with a structure and a dynamics" section how the presence of specific edges in the GA-STG of a BN is used to measure the compatibility of this BN with respect to a given dynamics.

#### Synthesis of BNs compatible with a structure and a dynamics

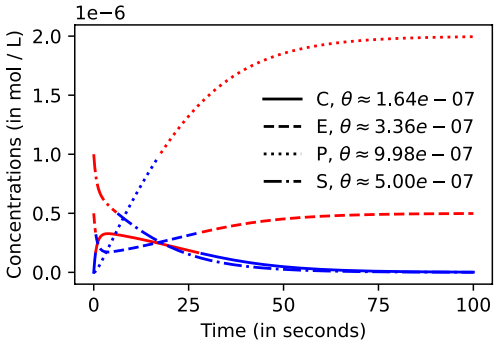
In general, a Boolean network that models a biological system has to satisfy two categories of *constraints*. On one hand, its structure has to comply with what is known on the system's structure. This knowledge concerns the list of species involved (genes, proteins...) and how they may influence each others. A *Prior Knowledge Network* (PKN) encodes such knowledge. It is defined similarly to the interaction graph of a

$$
\begin{aligned}
\frac{\mathrm{d}[\mathsf{C}]}{\mathrm{d}t} &= \mathrm{k_{on}}[\mathsf{E}][\mathsf{S}] - \mathrm{k_{off}}[\mathsf{C}] - \mathrm{k_{cat}}[\mathsf{C}] \\
\frac{\mathrm{d}[\mathsf{E}]}{\mathrm{d}t} &= -\mathrm{k_{on}}[\mathsf{E}][\mathsf{S}] + \mathrm{k_{off}}[\mathsf{C}] + \mathrm{k_{cat}}[\mathsf{C}] \\
\frac{\mathrm{d}[\mathsf{P}]}{\mathrm{d}t} &= 2\,\mathrm{k_{cat}}[\mathsf{C}] \\
\frac{\mathrm{d}[\mathsf{S}]}{\mathrm{d}t} &= -\mathrm{k_{on}}[\mathsf{E}][\mathsf{S}] + \mathrm{k_{off}}[\mathsf{C}]
\end{aligned}
\qquad
\begin{aligned}
\mathrm{k_{on}} &= 1 \times 10^{6}\,\mathrm{L\,mol^{-1}\,s^{-1}} \\
\mathrm{k_{off}} &= 0.2\,\mathrm{s^{-1}} \\
\mathrm{k_{cat}} &= 0.1\,\mathrm{s^{-1}} \\
[\mathsf{C}]_0 &= 0\,\mathrm{mol\,L^{-1}} \\
[\mathsf{E}]_0 &= 5 \times 10^{-7}\,\mathrm{mol\,L^{-1}} \\
[\mathsf{P}]_0 &= 0\,\mathrm{mol\,L^{-1}} \\
[\mathsf{S}]_0 &= 1 \times 10^{-6}\,\mathrm{mol\,L^{-1}}
\end{aligned}
$$

(a)




(b)




(c)

**Fig. 2** For the running example depicted in Eq. (1): **a** ODE system and its parametrisation; **b** prior knowledge network; **c** multivariate time series obtained by simulation of the ODE system, midrange-based binarisation thresholds, and resulting binarisation (blue if 0 and red if 1)

Boolean networks: it is a directed graph whose nodes are the species of the studied system, and an edge $\mathsf{Y} \overset{\sigma}{\multimap} \mathsf{X}$ exists whenever we presume $\mathsf{Y}$ might have a role to play (with a polarity $\sigma \in \{+, -, \pm\}$ in the dynamics of $\mathsf{X}$. The *potential parents* of a component $\mathsf{X}$ are the species $\mathsf{Y} \in \mathcal{C}$ such that $\mathsf{Y} \multimap \mathsf{X} \in \mathscr{P}$. Figure 2b shows an example of PKN of the enzymatic reaction Eq. (1). In this PKN, $\mathsf{S}$, $\mathsf{C}$ and $\mathsf{E}$ are potential parents of $\mathsf{E}$ with polarities $-$, $+$ and $-$ respectively. The PKN is used to constrain the structure of the synthesised BNs: a BN is *compatible* with a given PKN if its interaction graph is a spanning subgraph of the PKN. In other words, the interaction graph of a BN compatible with a given PKN is formed of the nodes and a subset of the edges of the PKN. This results in constraining which species can appear as variables in each transition function and the polarity of those variables. Hence, a component $\mathsf{P}$ is allowed in the transition function of a component $\mathsf{X}$ with a polarity $\sigma$ if the PKN contains an edge $\mathsf{P} \overset{\sigma}{\multimap} \mathsf{X}$. For example, $\mathscr{B}_1$ (Fig. 1a) is compatible with the PKN given in Fig. 2b. On the contrary, a Boolean network having the transition function $f_{\mathsf{E}} := \neg\mathsf{S} \vee \neg\mathsf{C}$ is not compatible. Indeed, despite $\mathsf{C}$ being a possible parent of $\mathsf{E}$, the negative polarity is not allowed since $\mathsf{C} \overset{-}{\multimap} \mathsf{E}$ is not in the PKN.

On the other hand, the dynamics of the synthesised BN has to comply with what is known on the system's dynamics. Starting from a binarised multivariate Time Series (TS) of the concentrations of the species over time, we can extract a sequence

of configuration transitions. For example, the sequence extracted a midrange-based binarisation of the multivariate TS given in Fig. 2c is $1001 \rightarrow 0001 \rightarrow 0101 \rightarrow 0100 \rightarrow 0110 \rightarrow 1010$. For a given synthesised BN, we define its *coverage ratio* as the number of transition present in its General Asynchronous STG (GA-STG), divided by the number of distinct transitions in the sequence. Ideally, we would like the sequence to be a *walk* in the GA-STG i.e., that the GA-STG contains all the transitions appearing in the sequence. In such a case, the coverage ratio of the GA-STG in regard to the configuration's sequence is of 1, and the Boolean network is said to be *fully compatible* with the multivariate TS. However, it is not always possible to retrieve the complete walk in the GA-STG (Paulevé et al. 2020). In this case, the goal is to have the best coverage ratio.

All in one, a Boolean network is *compatible* with a Prior Knowledge Network (PKN) if its interaction graph is a spanning subgraph of the PKN, and the compatibility between a Boolean network and a multivariate Time-Series (TS) is quantified using the *coverage ratio*. An ideal Boolean network synthesis method would *only* construct Boolean networks compatible with the given PKN and with the *maximal* coverage ratio (ideally of 1) achievable in regard of the given multivariate TS.

## Description of the SBML2BN pipeline

We propose SBML2BN, a pipeline for the automatic synthesis of Boolean networks starting from an existing quantitative SBML description of a biological system. All the necessary concepts about SBML are described in "SBML in a nutshell" section. The structure (PKN) and the dynamics (TS) of the biological system under study are extracted from the SBML model ("Extraction of the PKN from the SBML model" and "Extraction of the time-series from the SBML model" sections). In the BN synthesis step ("Boolean networks synthesis" section), the former hard constrains the structure of the resulting BNs, while the latter acts as soft constraints. The pipeline finishes with the evaluation of the *set* of the BNs it produces ("Evaluation of synthesised Boolean networks" section).

### SBML in a nutshell

The Systems Biology Markup Language (SBML) (Keating et al. 2020) is an XML markup language. The SBML file representing the biological system from Eq. (1) is given in the Additional file 1. The SBML standard[1] specifies how the different elements are named and structured. This paper focuses on a subset of SBML models: those containing all the necessary information for the SBML2BN pipeline to interpret the model as a simulable differential–algebraic system of equations with discontinuous events. We refer to these SBML models as *complete quantitative SBML models*. We describe the content of such models as follows.

[1] http://sbml.org/Documents/Specifications.

**Table 1** The three kind SBML rules

| Rule type | Description | General form |
|---|---|---|
| Algebraic | Left-hand side is zero | $0 = f(\boldsymbol{W})$ |
| Assignment | Left-hand side is a scalar | $x = f(\boldsymbol{V})$ |
| Rate | Left-hand side is a rate-of-change | $\mathrm{d}x/\mathrm{d}t = f(\boldsymbol{W})$ |

Let $x$ be a variable, $f$ a numerical function, $\boldsymbol{V}$ a vector of variables that does not include $x$, and $\boldsymbol{W}$ a vector of variables that may include $x$

#### *Species*

A species corresponds to a pool of entities (such as ions, proteins and other molecules) that makes sense in the context of a given model. Its concentration can change over time, according to the processes described in the SBML model (reactions, rules and events).

#### *Reactions*

A reaction $\mathcal{R}$ describes a process that can change the amount of one or more species. It is defined as a list of *reactants*, a list of *products*, and a *kinetic law* $e_{\mathcal{R}}$ (i.e., a mathematical expression which gives the speed of $\mathcal{R}$). For each species $\mathsf{X}$ involved in $\mathcal{R}$, the *net stoichiometry* $\nu_{\mathcal{R}}^{\mathsf{X}}$ of $\mathsf{X}$ in $\mathcal{R}$ is the amount of $\mathsf{X}$ as a product minus its amount as a reactant. If $\nu_{\mathcal{R}}^{\mathsf{X}} > 0$ (resp. $< 0$), $\mathsf{X}$ is *effectively* produced (resp. consumed) by $\mathcal{R}$. If $\nu_{\mathcal{R}}^{\mathsf{X}} = 0$, then $\mathsf{X}$ is somehow involved in $\mathcal{R}$ (i.e., it influences the speed of the reaction), without having its amount modified. Such species is called a *modifier*. A modifier which increases (resp. decreases) the speed of the reaction is an *activator* (resp. *inhibitor*). In some SBML models, specific annotations [using the Systems Biology Ontology (Courtot et al. 2011)] indicate the exact role of the modifiers.

A Chemical Reaction Network (CRN) consists of a set of reactions taking place using a given set of species. In this sense, an SBML model consisting only of a set of species and a set of reactions is a CRN. CRNs are well studied and numerous theoretical and practical tools are available to analyse them (Calzone et al. 2006; Hoops et al. 2006).

#### *Events*

An event corresponds to a discontinuous change in the dynamics, as it performs some given assignments as soon as some given condition become true. For example, in the model n°111[2] describing the cell cycle of fission yeast (Novak et al. 2001), two events are used to reset the cell mass M (divide it by two) when MPF decreases through 0.1:

| | Condition | Assignment |
|---|---|---|
| Event 1 | $(\mathrm{MPF} \leq 0.1) \wedge (\mathrm{flag}_{\mathrm{MPF}} = 1)$ | $\mathrm{M} = \mathrm{M}/2$<br>$\mathrm{flag}_{\mathrm{MPF}} = 0$ |
| Event 2 | $\mathrm{MPF} > 0.1$ | $\mathrm{flag}_{\mathrm{MPF}} = 1$ |

[2] https://www.ebi.ac.uk/biomodels/BIOMD0000000111.

#### *Rules*

A rule constrains the model for the entire duration of a simulation, as it defines relationships among variables (species concentrations or parameters values) which hold at all times. The SBML standard defines three types of rules: algebraic, assignment and rate. They are briefly defined in Table 1. In the model n°111, for example, an assignment rule is used to set the value of a parameter $\sigma$ from two species and a parameter along all the simulation: $\sigma = \mathrm{cdc}_{13}\mathrm{T} + \mathrm{rum}_1\mathrm{T} + \mathrm{Kdiss}$.

#### *Completeness and well-formedness*

A quantitative SBML model is *complete* if it specifies all initial values of concentrations and kinetic parameters used in the model. When a model is not complete, it cannot be simulated quantitatively as we do it in "Extraction of the time-series from the SBML model" section.

We consider an SBML model to be *well-formed* when it respects the SBML specifications, and when each of its reactions $\mathcal{R}$ respects the following criteria introduced in Fages et al. (2012):

1. Its kinetic expression $e$ is well-defined and partially differentiable. It is positive if $\mathcal{R}$ is irreversible;
2. A species $\mathsf{Y}$ belongs to the set of reactants or activators of $\mathcal{R}$ if and only if $\frac{\partial e}{\partial \mathsf{Y}} > 0$ for some positive values of concentration;
3. A species $\mathsf{Y}$ belongs to the set of inhibitors of $\mathcal{R}$ if and only if $\frac{\partial e}{\partial \mathsf{Y}} < 0$ for some positive values of concentration.

The well-formedness ensures the consistency of the description of the reactions with their kinetic expression, which is an important precondition for our pipeline. A model consisting of the reaction "$\mathsf{X}$ disappears at the given constant speed k" (noted $\mathsf{X} \xrightarrow{\mathrm{k}\times[\mathsf{Y}]} \_$) is not well-formed. Indeed, despite $\mathsf{Y}$ appearing in the kinetic expression and thus having an impact on the degradation of $\mathsf{X}$ ($\frac{\partial e}{\partial \mathsf{Y}} \neq 0$), it is not listed as reactant nor modifier of the reaction. The tool Biocham (Calzone et al. 2006) is able to determine if a given SBML model is well-formed, and to improve its well-formedness when possible (Fages et al. 2012).

Altogether, SBML can represent hybrid quantitative models from which we can reconstruct a differential–algebraic equations system. Assuming this system has a solution, it can be simulated numerically to retrieve the species concentrations over time (see "Extraction of the time-series from the SBML model" section).

### Extraction of the PKN from the SBML model

This first step consists in the construction of the PKN (noted $\mathscr{P}$). Figure 2b is the PKN constructed by SBML2BN for Eq. (1). The nodes of the PKN are the SBML species of the SBML model. As for the edges, they are obtained by applying the following routines:

- on each reaction of the SBML model:
  - **if** Y is a reactant or an activator and X disappears **then** $\mathsf{Y} \xrightarrow{-} \mathsf{X} \in \mathscr{P}$
  - **if** Y is an inhibitor and X appears **then** $\mathsf{Y} \xrightarrow{-} \mathsf{X} \in \mathscr{P}$
  - **if** Y is a reactant or an activator and X appears **then** $\mathsf{Y} \xrightarrow{+} \mathsf{X} \in \mathscr{P}$
  - **if** Y is an inhibitor and X disappears **then** $\mathsf{Y} \xrightarrow{+} \mathsf{X} \in \mathscr{P}$
- on each rule:
  - **if** Y is a species which appears on the right side of a rule (assignment or rate) defining X **then** $\mathsf{Y} \xrightarrow{+} \mathsf{X} \in \mathscr{P}$ and $\mathsf{Y} \xrightarrow{-} \mathsf{X} \in \mathscr{P}$
- on each event:
  - **if** Y is a species which appears in a condition triggering a discontinuous change of X **then** $\mathsf{Y} \xrightarrow{+} \mathsf{X} \in \mathscr{P}$ and $\mathsf{Y} \xrightarrow{-} \mathsf{X} \in \mathscr{P}$

The four routines concerning the reactions correspond to the routines that are used to derive the so-called Syntactical Influence Graph (SIG) of a CRN (Fages and Soliman 2008a). In the conference version of this paper (Vaginay et al. 2022), they were the only ones used to construct the PKN from an SBML model. We will later discuss the impact of the use of the two others routines.

#### Extraction of the time-series from the SBML model

The goal of this step is to retrieve the concentrations of the species over time. The changes are determined by a differential–algebraic system of equations which is reconstructed from the SBML model and then integrated numerically. Figure 2a shows the system, parametrisation and initial conditions retrieved from the SBML model (see Additional file 1) of the running example—which does not use rules nor events. Figure 2c shows the multivariate TS obtained by simulating Fig. 2a for $t_{\max} = 100$ s (chosen arbitrarily). The reconstruction of the system and its numerical integration are done as follows.

##### *Reconstruction of the differential–algebraic system of equations*

An expression representing the overall rate of change of the amount of each species is constructed from the set of SBML reactions. This expression corresponds to the algebraic sum of the contributions of all the *relevant* reactions (i.e., reactions in which a given species is involved as a product or a reactant). For example, in the running example Eq. (1), the species C is involved as a product in reaction $\mathcal{R}_{\text{on}}$ and as a reactant in reactions $\mathcal{R}_{\text{cat}}$ and $\mathcal{R}_{\text{off}}$. Altogether, the overall rate of change of C is: $\frac{\mathrm{d}\mathsf{C}}{\mathrm{d}t} = \nu^{\mathsf{C}}_{\mathcal{R}_{\text{on}}} \cdot e_{\mathcal{R}_{\text{on}}} + \nu^{\mathsf{C}}_{\mathcal{R}_{\text{off}}} \cdot e_{\mathcal{R}_{\text{off}}} + \nu^{\mathsf{C}}_{\mathcal{R}_{\text{cat}}} \cdot e_{\mathcal{R}_{\text{cat}}}$ with $\nu^{\mathsf{C}}_{\mathcal{R}_{\text{on}}} = 1$, and both $\nu^{\mathsf{C}}_{\mathcal{R}_{\text{off}}}$ and $\nu^{\mathsf{C}}_{\mathcal{R}_{\text{cat}}} = -1$. As stated in the SBML representation (see Additional file 1) of Eq. (1), the speed of each reaction $e_{\mathcal{R}}$ is proportional (with a factor $k_{\mathcal{R}}$) to the product of the concentration of reactants of the reaction, leading to the equations in Fig. 2a.

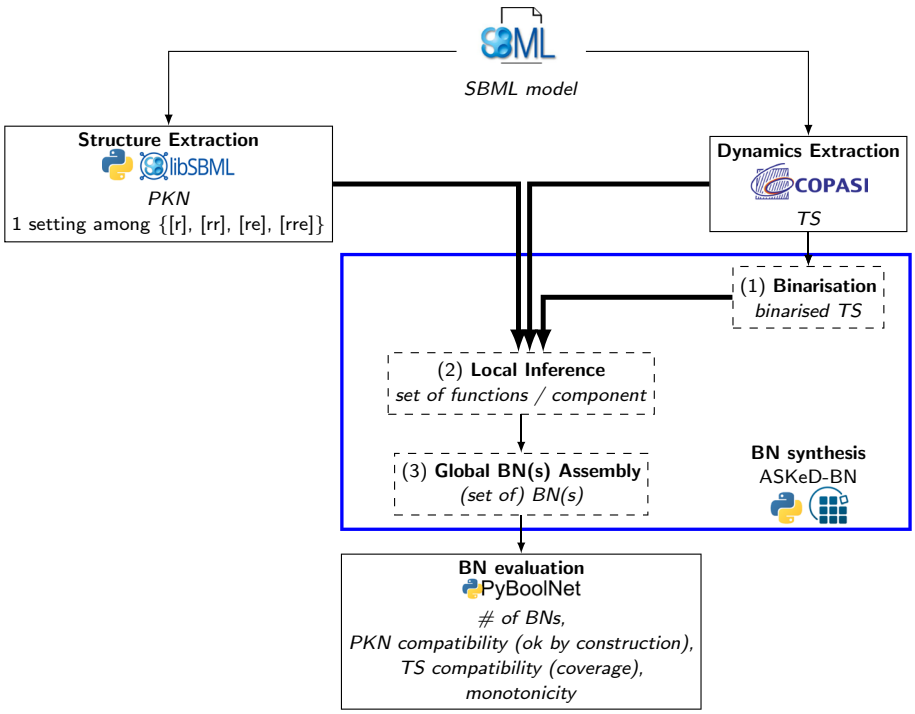



**Fig. 3** Workflow of the SBML2BN pipeline. The BN synthesis step (blue box) is composed of 3 steps among which the local inference of the transition functions which uses 3 inputs (thick arrows): a PKN, a TS, and its binarisation

As for the rules ("Rules" section) and events ("Events" section), they respectively define additional relationships among variables that hold at all time steps and trigger some discontinuous changes as soon as the specified conditions are met.

#### *Numerical integration*

We assume that the reconstructed differential–algebraic system has a solution. We run a deterministic numerical time integration from $t = 0$ to $t_{\max}$. During the simulation, the solver adjusts automatically the size of the time-step in order to reduce the approximation error and to trigger the events on appropriate time steps.

### Boolean networks synthesis

This step infers a set of Boolean networks from the extracted multivariate TS and PKN. In such a context, and despite the PKN constraining the structure of the BNs to synthesise, the synthesis problem is under-specified. The reason is that only one multivariate TS is used: for the running example, there are a priori $2^{2^4} = 65536$ possibilities or which 128 satisfy the PKN. To further constrain the number of solutions, it is thus it is very important to use dynamical constraints as well.

In Vaginay et al. (2021), we introduced ASKeD-BN and showed that it is the best synthesis method available in the case of signed PKN and complete multivariate TS (i.e., without missing time steps), when compared to the state-of-the-art methods (they will be briefly presented in "Related works" section). ASKeD-BN exhaustively synthesises

BNs compatible with a given PKN and a multivariate TS with respect to constraints closely related to the notion of *compatibility* defined in "Synthesis of BNs compatible with a structure and a dynamics" section. It is decomposed in three steps: (1) the TS binarisation, (2) the *local inference* of transition functions and (3) the *global BN assembly*. The second step generates the set of formulas that respect the structure and dynamics known for each component, while the third step generates all the BNs from the formulas found in the second step. Figure 3 summarises the steps of the algorithm.

#### *TS binarisation*

The values in the time-series $\mathscr{T}$ obtained in "Extraction of the time-series from the SBML model" section are real valued, but ASKeD-BN needs the corresponding Boolean observations. The choice of the binarisation is crucial for the outcome. For a given species $\mathsf{X} \in \mathcal{C}$, ASKeD-BN can directly use its binarisation threshold $\theta_{\mathsf{X}}$ if it is provided. Otherwise, we compute $\theta_{\mathsf{X}}$ as the midrange of the observations of $\mathsf{X}$: $(\min + \max)/2$, where min and max are the observed minimum and maximum of $\mathsf{X}$ in the time series. With $\mathsf{X}_t$ the value of the concentration of $\mathsf{X}$ at time $t$, the binarised value of $\mathsf{X}$ at time $t$ is 1 if $\mathsf{X}_t \geqslant \theta_{\mathsf{X}}$ and 0 otherwise. Other binarisation procedures are possible (such as mean and median-based), but despite its simplicity, the midrange-based binarisation procedure produces good results when applied in the context of BN synthesis from biological data (Videla et al. 2015; Ostrowski et al. 2016). In particular, midrange-based binarisation may be less impacted than the average and median-based binarisation by periods of time where the concentration of a species oscillates in a small range of values.

#### *Local inference of transition functions*

This step constructs, for each species $\mathsf{X} \in \mathcal{C}$, all the simplest transition functions that are compatible with the given PKN, and explain the TS as well as possible. Overall, it solves both a combinatorial problem (structure constraint) and an optimisation problem (dynamics and minimality constraints).

*Minimality Constraint—What are the best functions with regard to their size?* The candidate transition functions are represented in Disjunctive Normal Form (DNF), i.e., disjunction of conjunctions. We represent a satisfiable conjunctive clause over a list of species $\mathcal{I} \subseteq \mathcal{C}$ as an assignment $c_{\mathcal{I}} : \mathcal{I} \to \{-1, 1, 0\}$. The assignment encodes how each species appears in the clause (negatively/positively/does not appear). For example, if $\mathcal{I} = (\mathsf{X}, \mathsf{Y}, \mathsf{Z})$, the assignment $(1, 1, 0)$ encodes $\mathsf{X} \wedge \mathsf{Y}$ and $(1, -1, 1)$ encodes $\mathsf{X} \wedge \neg\mathsf{Y} \wedge \mathsf{Z}$. A DNF is represented by a set of such assignments. Thanks to a constraint minimising the number of literals used in the DNF, our encoding retrieves only *cardinal-minimal DNF*.

*Structural Constraint—What are the best functions with regard to the PKN $\mathscr{P}$?* For a component $\mathsf{X} \in \mathcal{C}$, we denote with $\mathcal{P}(\mathsf{X})$ the set consisting of its parents:$\mathcal{P}(\mathsf{X}) = \{\mathsf{Y} \mid \mathsf{Y} \twoheadrightarrow \mathsf{X} \in \mathscr{P}\}$. A clause $c_{\mathcal{P}(\mathsf{X})}$ is said to *respect* a given PKN if all its assignments $\sigma$ different from 0 correspond to influences in the PKN. That is, for all species $\mathsf{Y} \in \mathcal{P}(\mathsf{X})$, if $c_{\mathcal{P}(\mathsf{X})}(\mathsf{Y}) \neq 0$ then $\mathsf{Y} \xrightarrow{c_{\mathcal{P}(\mathsf{X})}(\mathsf{Y})} \mathsf{X} \in \mathscr{P}$. If $\mathcal{P}(\mathsf{X}) = \emptyset$, the only transition functions that can be synthesised are the constant functions True and False.

Our encoding states that it is forbidden to generate a DNF containing a clause that does not respect the given PKN. This constraint ultimately results in synthesising

Boolean networks whose interaction graphs are spanning subgraph of the PKN ("Synthesis of BNs compatible with a structure and a dynamics" section).

*Dynamical Constraint—What are the best functions with regard to the TS $\mathscr{T}$?* Let $\mathcal{I} = \mathcal{P}(\mathsf{X}) \cup \{\mathsf{X}\}$. From the binarised TS $\hat{\mathscr{T}}$, we extract the deduplicated sequence of configurations $\mathscr{S}_{\mathcal{I}}$. It is a sequence of vectors of $\mathbb{B}^{|\mathcal{I}|}$ as it only concerns the species in $\mathcal{I}$. The configurations that repeat over several successive time steps in $\hat{\mathscr{T}}$ are discarded. Hence, the $n$th configuration in $\mathscr{S}$ is different from the configuration $n-1$. We denote with $s$ the function that returns the list of time steps that repeat the $n$th configuration. For example, the deduplicated sequence of configurations obtained from the binarised observations $000, 100, 100, 110, 110, 111, 000$ is $000 \to 100 \to 110 \to 111 \to 000$. For this example, $s(1) = \{1\}$ and $s(2) = \{2, 3\}$. For each transition in this sequence, *input* refers to the configuration of the species in $\mathcal{P}(\mathsf{X})$ (which may include $\mathsf{X}$ itself), and *output* denotes the next status of $\mathsf{X}$. It is possible to get a sequence with inconsistencies: the same input leads to several outputs. However, we never get missing values thanks to the numerical integration from "Extraction of the time-series from the SBML model" section.

For a given candidate transition function $f_{\mathsf{X}}$, the *unexplained* configurations are the configurations where:

- $\mathsf{X}$ is activated in the $n$th configuration of $\mathscr{S}$, but $f_{\mathsf{X}}$ does *not* evaluate to True when using assignment of $\mathcal{P}(\mathsf{X})$ from the $n-1$th configuration;
- $\mathsf{X}$ is deactivated in the $n$th configuration of $\mathscr{S}$, but $f_{\mathsf{X}}$ evaluates to True when using the assignment of $\mathcal{P}(\mathsf{X})$ from the $n-1$th configuration.

The unexplained configurations form the set $\mathscr{U}$. For each unexplained configuration $n \in \mathscr{U}$, we compute an error $\epsilon_n$ that is the sum of "how far" the value of $\mathsf{X}$ is from the threshold for all time steps for which the configuration $n$ is repeated: $\epsilon_n = \frac{\sum_{t \in s(n)} |\mathsf{X}_t - \theta_{\mathsf{X}}|}{|s(n)|}$. The total error $\epsilon$ is $\sum_{n \in \mathscr{U}} \epsilon_n$. It is of 0 if the candidate function explains all transitions. Our encoding generates all the functions that minimises $\epsilon$. This constraint ultimately results in synthesising functions that fit to the transitions.

#### *Global Boolean networks assembly*

The final inferred Boolean networks are produced from the transition functions synthesised in the previous step, by selecting one formula per species. ASKeD-BN produces all possible assemblies. There may be numerous assemblies, since their number corresponds to the product of the number of functions synthesised for each species. However, in practice, the local inference often finds one function for each species, resulting in a unique assembly.

In case there are several assemblies, we also investigated the aggregation of the solutions by simply merging the different DNFs found for each species. More complex combinations could be investigated, such as what is done in Aghamiri and Delaplace (2021) where the unique assembly produced is the most appropriate one in regard to required global properties (such as stable states and monotonicity of the network).

```
Require: a component X, a PKN 𝒫, a multivariate TS 𝒯 and its binarisation 𝒯̂
function LOCAL_INFERENCE(X, 𝒫, 𝒯, 𝒯̂)
    min_e ← ∞                                   ▷ stores the smallest error computed so far
    min_l ← ∞                                   ▷ stores the smallest cardinality seen so far
    C ← generate_candidates(X, X.parents)       ▷ all combinaisons of all possible conjunctions on X.parents
    for c in C do
        if compatible(c, 𝒫) then
            c.error = compute_error(c, 𝒯, 𝒯̂)
            if c.error < min_e then
                min_e ← c.error
                min_l ← len(c.literals)
            else if c.error = min_e ∧ len(c.literals) < min_l then
                min_l ← len(c.literals)
    return [c for c in C if c.error = min_e ∧ len(c.literals) = min_l]
                                                ▷ candidates with smallest error and the smallest number of literals
```

**Fig. 4** Naive imperative algorithm for the local inference of ASKeD-BN

### Evaluation of synthesised Boolean networks

In this last step, we evaluate the quality of the BNs synthesised. Several criteria are considered:

- **number of BN generated**, which should be small.
- **compatibility of the synthesised BNs with the PKN and the TS extracted from the model** Since they are compatible with the PKN (by construction), our quality check focuses on the compatibility with the multivariate TS. As explained in "Synthesis of BNs compatible with a structure and a dynamics" section, the *coverage ratio* is the proportion of transitions extracted from the TS that are in the general-asynchronous STG. We compute this coverage ratio for each BN synthesised by the pipeline. Then we aggregate the individual coverage ratios by computing their median and standard deviation. Ideally, the pipeline returns *only* BNs with *maximal* coverage ratios i.e., with a median of 1 and a std of 0 ("Synthesis of BNs compatible with a structure and a dynamics" section).
- **monotonicity of the local transition functions**. Following the basic principle of parsimony, a biological species is usually assumed to have either an activation or an inhibition role towards another species (Sontag 2007). As a result, non-monotonous local transition functions (i.e., functions which contain a literal and its contrary) are supposed to be quite unlikely. Only a non-monotonous PKN can result in the synthesis of non-monotonous transition functions. That is, we count the number of parsimonious local update functions generated when the given PKN is non-monotonous. Note that, by construction, a PKN built from rules and events is non-monotonous.

### Implementation

We have made a point of supporting reproducibility and facilitating the installation of the pipeline.[3] All the tools developed or reused are open-source, well documented and freely available. The pipeline is managed using Snakemake (Mölder et al. 2021) (which ensures each step is ran properly and in the correct order) and installed using Conda

[3] https://gitlab.inria.fr/avaginay/CNA2021_extension.

(Conda 2021) (which simplifies the management of library dependencies and avoid version conflicts).

We implemented a tool to extract the PKN from an SBML model using the library libSBML (Bornstein et al. 2008). Because of many special cases, the interpretation and simulation of SBML models is difficult. Hence, we used the dedicated program COPASI (Hoops et al. 2006) to retrieve the multivariate TS (with the solver LSODA) and Biocham (Calzone et al. 2006) to determine the well-formedness of the models. PyBoolNet (Klarner et al. 2016) is used to compute the GA-STG of the BNs. As for the BN synthesis step, most of ASKeD-BN (Vaginay et al. 2021) is implemented in python, except for the local inference step which is implemented declaratively in answer-set programming (ASP) (Gebser et al. 2012). ASP relies on constraints and logic to define the solutions of a problem. It is powerful and quite adapted to the local synthesis problem, as it is both a combinatorial and optimisation problem. A naive procedural algorithm of this step is given in Fig. 4. The procedural algorithm evaluates all the candidates functions, but thanks to heuristics (inspired from SAT solvers), ASP performs clever exhaustive searches of all the solutions. Each solution it returns is a logical formula in minimal DNF which minimises the error in regard to the given TS, and respects the given PKN.

## Evaluation of the SBML2BN pipeline

### Evaluation on the running example Eq. (1)

We apply SBML2BN with the default midrange-based binarisation on the SBML file (see Additional file 1) that models Eq. (1). The Boolean network $\mathscr{B}_1$ (Fig. 1a) is the only solution we obtain. Its interaction graph (Fig. 1b) is a spanning subgraph of the PKN (Fig. 2b) by construction. It thus respects the known structure of the original SBML model. As for the GA-STG of this BN (Fig. 1c), it covers 4 transitions out of the 5 extracted from the binarised TS (Fig. 2c). Its coverage ratio is thus 0.8, and the coverage median and standard deviation of this singleton of solutions are obviously 0.8 and 0 respectively, making SBML2BN successful on this example. Using the synchronous update scheme, two fixed-point attractors are found for this BN: 0111 and 1000. They are consistent with what we would except biologically. In particular as "nothing happens" if the dynamics starts with only E present while there is no S.

We tried the BN synthesis with three other binarisation procedures: median, mean and "above 0". In these cases, the pipeline also returns a unique BN with coverage of respectively 0.6, 1 and 0.25. The BN obtained with the median-based binarisation is the same as the one obtained when using the midrange-based binarisation, but it has a reduce coverage because the sequence of configurations is slightly different. The BN synthesised with the mean-based coverage has the best coverage achievable. Despite this, we stick with the midrange-based coverage for the rest of the experiments because it is the simplest binarisation and not influenced by periods of time where a species oscillates in a small range of value.

#### Evaluation on SBML models from BioModels

BioModels (Malik-Sheriff et al. 2020) is a repository of models of biological and biomedical systems, including metabolic networks, signalling networks, gene regulatory networks and infectious diseases. All models stored in the `curated` branch of BioModels are encoded in SBML and have passed a manual curation process consisting in extensive annotation of the models elements and asserting the results from the paper in which the model was originally published are reproducible by the SBML model. In particular, we retrieve the duration of simulations from these curation reports, when applicable.

The latest available release of Biomodels[4] contains 640 curated SBML models, including 369 complete quantitative SBML (i.e., models for which SBML2BN is able to extract a PKN and a multivariate TS, see "SBML in a nutshell" section). However, the complexity of the BN synthesis problem increases exponentially with the number of parents for each component. Indeed, the number of possible transition functions for a component with $p$ parents is $2^{2^p}$. Assuming the problem is not tractable if a component has more than 10 incoming edges, we are considering the same 209 SBML models than in Vaginay et al. (2022). The number of species in these models ranges from 1 to 61 (median $= 8$, std $\sim 11$), but bigger models would not have been a problem *per se* since ASKeD-BN is not directly impacted by the number of species. Among these 209 models, 38 models use rules and/or events (3 have both). Only 30% (64) of these models are well-formed according to the tool Biocham.

In order to study the impact of the integration of the rules and events in the construction of the PKN, the pipeline is ran on these models in four different settings:

| Setting | PKN built from | # models concerned |
|---|---|---|
| [r] | Reactions only | 209 |
| [re] | Reactions + events | 3 |
| [rr] | Reactions + rules | 38 |
| [rre] | Reactions + rules + events | 3 |
| | | Total # xp = 353 |

In each setting, the pipeline is globally assessed according to four criteria:

- the runtime ("Runtime" section): attests that the pipeline scales to real SBML models.
- the average number of BNs returned for each SBML model ("Number of BNs synthesised" section): attests that the problem is sufficiently constrained such that the pipeline does not return an overwhelming number of alternative solutions, among which it would be difficult to choose.
- the distribution of median and standard deviation summarising the coverage ratios of the BNs synthesised for each SBML model ("Compatibility of the BNs with the TS (coverage analysis)" section): attest the compatibility of the dynamics of the BNs with the TS.

[4] release 31 ftp://ftp.ebi.ac.uk/pub/databases/biomodels/releases/2017-06-26/.

**Table 2** Number of models processed and CPU time of the BN synthesis step

| Setting | # to process | # done < 30 h (%) | CPU time median (min) ± std |
|---|---|---|---|
| [r] (reactions only) | 209 | 155 (~ 75) | ~ 33 ± 5.6 h |
| [re] (reactions + events) | 3 | 2 (~ 66) | ~ 28 ± 14 min |
| [rr] (reactions + rules) | 38 | 29 (~ 75) | ~ 7 ± 6 h |
| [rre] (reactions + rules + events) | 3 | 1 (~ 33) | ~ 51 |
| Total | 253 | 187 (~ 75) | ~ 31 ± 5.6 h |

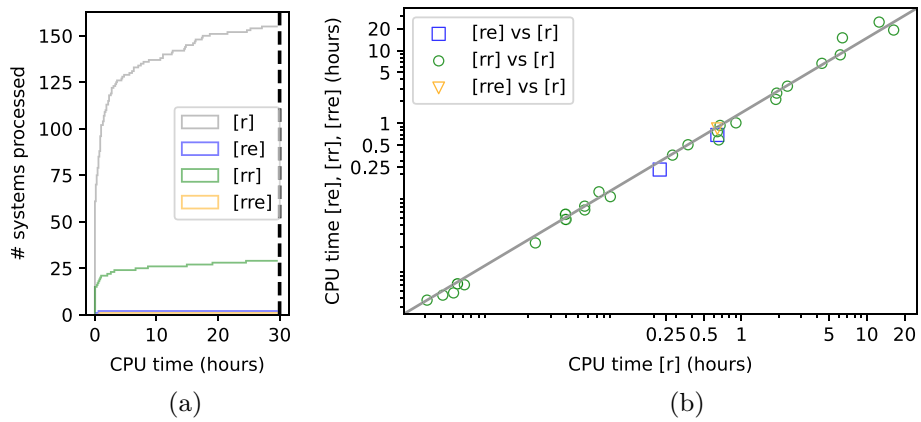



**Fig. 5** Runtime of the BN synthesis step in the four settings (**a**) and in comparison with [r] (**b**)

- the monotonicity of the transition functions ("Monotonicity analysis" section): attests the parsimony of the influences used by the BNs.

Note that we do not evaluate the compliance of the synthesised BN with the PKN, because they are compliant by construction ("Local inference of transition functions" section). Compared to our previous paper (Vaginay et al. 2022), we added the analysis of the monotonicity as well as the study of how the results are impacted according to the different settings of PKN construction, and a discussion on the well-formedness of the models. For a more detailed analysis of the results concerning the use of ASKeD-BN to synthesise BNs from given PKN and TS (not automatically extracted from an SBML model), the reader is invited to read (Vaginay et al. 2021).

#### *Runtime*

Despite the complexity of the BN synthesis step, about three fourth (187) of the experiments terminated in less than 30 h. Table 2 and Fig. 5a show how many models were processed in less than 30 h, as well as CPU time of the BN synthesis step in each setting. From Fig. 5b, we can see that the BN synthesis step stopped processing an interesting number of models after 10 h. In the following, we report the results for the 187 experiments terminated in less than 30 h.

**Table 3** Number of BNs generated, coverage and number of clauses

| Setting | Median # BNs |
|---|---|
| [r] | 1 |
| [re] | 1.5 |
| [rr] | 1 |
| [rre] | 1 |
| All | 1 |

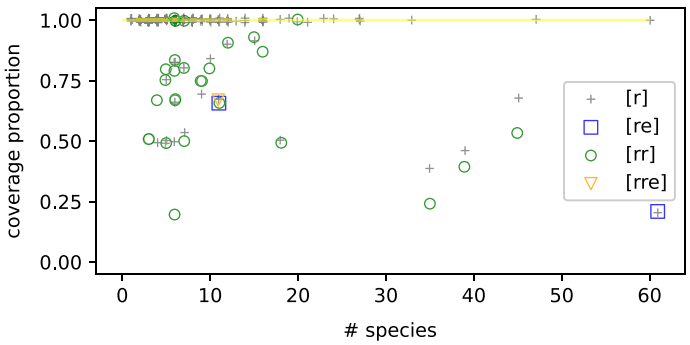



**Fig. 6** Coverage evaluation for the BNs synthesised by SBML2BN for 155 SBML models. Each dot represents the set of BNs returned for a given SBML model in a given setting. Its coordinates are the coverage ratio median (ordinate) and the number of species of the SBML model (abscissa). The yellow line shows where the dots are when the pipeline only returns BNs with a perfect coverage. The points are slightly jittered on x and y axes with a Gaussian noise of variance 0.2 and 0.02 respectively to ensure readability

To see how the addition of rules and/or events impacts the runtime of the local synthesis for a given model, we plot the runtimes for settings [re], [rr] and [rre] against the runtimes obtained with the setting [r] (Fig. 5b). The dots are on the diagonal when there is no change, and above (resp. below) the diagonal when the addition of rules and/or events leads to bigger (resp. smaller) runtime. Surprisingly, adding rules and/or events does not necessarily increase the runtime.

#### *Number of BNs synthesised*

Around 8 BNs are generated in average in each experiment. This number hides a strong disparity, since a single BN was synthesised for almost 70% (126) of the experiments. Table 3 shows the details for each setting. We can see that the choice of the setting does not have an impact on the number of BNs generated.

#### *Compatibility of the BNs with the TS (coverage analysis)*

To assess the coverage ratio criterion, we plot the median of the BNs synthesised for each SBML model in Fig. 6. As said before, all the BNs returned by the pipeline for a given SBML model would ideally have a perfect coverage ratio, hence with a median of 1 and a standard deviation of 0. The pipeline synthesises only BNs with maximal coverage

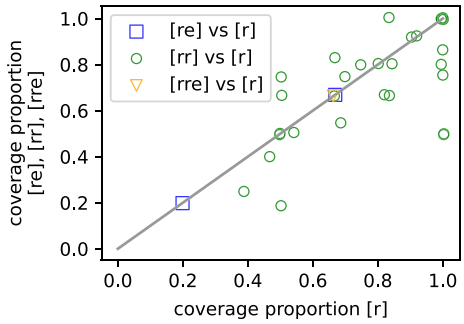



**Fig. 7** Evaluation of the impact on the coverage of constructing the PKN with rules and/or events. The points are jittered with a Gaussian noise of variance 0.01 on both axes to ensure readability

**Table 4** Impact of the setting on the coverage of the synthesised BNs

| Setting | # xp | ↗ | ↘ | = |
|---|---|---|---|---|
| [re] | 2 | | | 2 |
| [rr] | 29 | 13 | 8 | 8 |
| [rre] | 1 | | | 1 |
| Total | 32 | 13 | 8 | 11 |

ratio for almost three forth (139) of the experiments. The mean, median and standard deviation of the median coverage ratios of the BNs synthesised are of 0.90, 1 and 0.19 respectively. There are only 4 experiments for which the standard deviation is not 0 (max = 0.22). Overall, the pipeline is efficient at finding Boolean networks with good coverage median and small standard deviation, whatever the considered setting. Nevertheless, there are experiments for which the coverage of the synthesised BNs is not good. In particular, there is a significant loss of performance correlated to the number of nodes in the systems (Kendall' $\tau$ value of $-0.19$, $p$ value of 0.001).

We are currently investigating possible reasons of this correlation, and reasons of poor coverage ratio in general. One reason could simply be that Boolean networks cannot explain all phenomena ("Synthesis of BNs compatible with a structure and a dynamics" section): in some cases, the maximum achievable coverage ratio is smaller than 1, but our quality evaluation of the synthesised BNs does not take this fact into account. We could use Boolean networks with the most-permissive semantics (Chatain et al. 2020) to overcome this limitation, but no implementation is available for BNs having non-monotonous transition functions (such as the ones our pipeline might produce). In the previous version of the paper (Vaginay et al. 2022), we speculated that another reason could be that the specifications of SBML leave open the possibility for a model to contain contradictory information. It has been showed in Fages et al. (2012) that more than 60% of the SBML models tested in 2012 were not well-formed ("Completeness and well-formedness" section). For example, the model n°44[5] has reactions with species used in the kinetics which are not listed as reactants nor modifiers. This has a bad impact on the

[5] https://www.ebi.ac.uk/biomodels/BIOMD0000000044.

**Table 5** Impact of the setting on the search space and synthesised functions in terms of monotonicity and constancy

| Setting | # models concerned | # species concerned | # species in PKN, compared to [r] | | # species from IG of merged BNs, compared to [r] | |
|---|---|---|---|---|---|---|
| | | | w/ potential parents | w/ only monotonous incoming influence | Constant | Monotonous |
| [re] versus [r] | 2 | 72 | 50 versus 46 → + 4 | 25 versus 29 → − 4 | 70 versus 70 → 0 | 72 versus 72 → 0 |
| [rr] versus [r] | 29 | 333 | 309 versus 234 → + 75 | 122 versus 197 → − 77 | 306 versus 310 → − 4 | 333 versus 333 → 0 |
| [rre] versus [r] | 1 | 11 | 10 versus 7 → + 3 | 2 versus 5 → − 3 | 10 versus 10 → 0 | 11 versus 11 → 0 |

construction of the PKN by our pipeline ("Extraction of the PKN from the SBML model" section), since potential parents of some species are not identified. For this particular model in setting [r], one BN was generated, with a poor coverage of 0.55. Among the 187 experiments analysed, 131 concern not well-formed models and 56 well-formed models. However, the coverages obtained from well-formed models are not really different from the coverages obtained from not well-formed (means and median of 0.9 and 1 in both cases). We also hypothesised contradictions were most likely to occur in bigger models, but this is not the case. Indeed, the median size of the 64 well-formed models in the complete set of models is of 9.5 versus 7 for the 145 not well-formed models.

Let us now we consider specifically the impact of a PKN built with rules and/or reactions on the coverage results. For a given SBML model, we check how the coverages in setting [re], [rr] and [rre] differ from the ones obtained in setting [r] (Fig. 7 and Table 4). We can see that adding events does not impact the coverage of the synthesised BNs. The synthesised BNs are actually the same. Adding rules, however, has a mixed impact. There are 8 experiments for which it changed nothing, but 13 for which is improves the coverage and 8 for which it decreases the coverage. We are planning to investigate automatic ways to determine in advance which rules are worthy to be considered for the PKN construction.

#### *Monotonicity analysis*

Table 5 reports the impact of the introduction of rules and events on the PKN and the synthesised functions in terms of parenthood and monotonicity. The analysis of constancy and monotonicity of the synthesized functions is done on the interaction graph of a BN obtained by merging the BNs solutions (as explained in "Global Boolean networks assembly" section).

The number of species without any potential parent in the PKN (for which the synthesised function is then the constant False by default) decreases when adding rules and/or events. As a result, 4 species in the setting [rr] led to synthesised functions which are not the constant function (the default one).

Concerning the monotonicity, the introduction of rules and events leads by construction to non-monotonous influences in the PKN. Hence, in settings [re], [rr] and [rre], the number of species for which the PKN contains at least one non-monotonous influence increases compared to what is observed for the setting [r]. However, the synthesised functions are all monotonous.

## Related works

A mathematical method to convert an ODE system to a partial function from $\mathbb{B}^n \to \mathbb{B}^n$ (similar to a BN) has been explored in Davidich and Bornholdt (2008). It consists in a coarse-grain interpretation of the equations normalised between 0 and 1. It was successfully applied on equations modelling the cell division cycle of fission yeast. However, the generated partial function is not a strict BN as generated by our approach. There are no local transition functions. Moreover, it is impossible to apply it automatically on given ODE systems as the conversion relies on expert choices such as deduplication of some species, and the inclusion of some kinetic parameters (as if they were species).

If the SBML model under study corresponds to a Chemical Reaction Network (CRN) i.e., a set of reactions over a set of species, without rules nor events, one can automatically build a Boolean transition system using Biocham (Calzone et al. 2006). It implements the Boolean interpretation of a CRN defined in Fages and Soliman (2008b). A non-deterministic (asynchronous) transition system over the Boolean configurations of the CRN is build in the following way: if there is a reaction $\mathsf{A} + \mathsf{B} \to \mathsf{C}$ in the model, then the configurations 110 and 111 ($\mathsf{A}$ and $\mathsf{B}$ present, $\mathsf{C}$ don't care) are connected to the four following configurations: 001, 101, 011, 111 ($\mathsf{c}$ is for sure present, $\mathsf{A}$ and/or $\mathsf{B}$ might be consumed). The authors proved that this is a correct over-approximation of the quantitative behaviour of the CRN: the absence of a behaviour with this Boolean semantics entails its absence in the quantitative semantics of the original chemical reaction network, whatever the kinetic expressions are.

Several methods have been proposed in the literature for the BN synthesis from a PKN and a multivariate time series (Liang et al. 1998; Lähdesmäki et al. 2003; Ostrowski et al. 2016) as well as for the more general problem of BN synthesis from experimental data (such as omics data) and background knowledge (extracted from literature, or public databases) (Aghamiri and Delaplace 2021; Chevalier et al. 2019; Barman and Kwon 2018; Dorier et al. 2016). These methods exploit various strategies, especially regarding (i) the extraction method of the sequence of configurations and (ii) the fitting method of the transition functions to the observations. They all roughly amount to enforcing that the IG and STG of the synthesised BNs contains specific edges that corresponding to specific interactions and transitions of configurations.

Although they inspired us in our work, these studies differ from ours. Indeed, some methods such as caspo-TS (Ostrowski et al. 2016) work on explaining the *reachability* of the configurations instead of the transitions themselves. Hence, wildcard are thus added to the configurations sequence: $000 \to * \to 100 \to * \to 110 \to * \to 111 \to * \to 000$. This feature is an asset in the case of missing time points, but in our framework, the multivariate TS is complete, and this feature is not necessary [and even counter-productive (Vaginay et al. 2021)]. Some others perform a stochastic or greedy search for the candidate BNs. In contrast, our study aims at finding *all* solutions that satisfy the criteria we defined. Some of these methods assume that the data has been binarised beforehand and do not include a binarisation step. On the other side, they have to identify the correct sequence of configurations which will constrain the BN construction. Some of these methods are ASP-based as well (Chevalier et al. 2020; Videla et al. 2015) and were validated on synthetic data or on targeted complex biological systems.

## Conclusion and perspectives

In this paper, we presented SBML2BN, a pipeline for the automatic transformation of a *complete* quantitative SBML model into a set of compatible Boolean networks. The transformation of biological models from a formalism to another has been investigated in several papers (Aghamiri et al. 2020; Fages and Soliman 2008b) in particular from ODE system to Boolean networks (Davidich and Bornholdt 2008). Yet, to the best of our knowledge, our study is the first to be dedicated to the *automatic* transformation of a complete quantitative SBML model into Boolean networks. As a complete and automatic process, our pipeline reduces the risk of errors and saves effort and time of biologists. Our results show that SBML2BN succeeds most of the time at recovering small sets of BNs compatible with both the structure and dynamics extracted from the input SBML model. By construction, the Boolean networks synthesised by our pipeline are compatible with the structure of the input SBML model. They also tend to maximize the coverage ratio towards the observed dynamics of the system.

Overall, SBML2BN is an important building block on which we can build upon. So far, we take reactions, rules and events to retrieve the influences among species, and we use a deterministic simulation of the model to get the behaviour of the species. To go further, other SBML elements could be taken into account (such as the ones introduced in the last version of SBML (Zhang et al. 2020) to model species with multiple components or states). Moreover, certain handcrafted BNs contain tricks to fit to the data. For example, using nodes that are parameters and not species *per se*. These nodes are not exploitable by the automatic pipeline, as it is difficult to identify such tricks. One interesting perspective would be to take external expert knowledge into account in future versions of the pipeline, such as known fixed points and cyclic attractors. We are also investigating strategies to make the pipeline more efficient, particularly on more complex models. Finally, we plan to take benefit of the *set* of BNs synthesised for a given SBML model by combining and simulating them together, as recently proposed in Chevalier et al. (2020). We are also investigating how to aggregate BNs from several SBML models when they concern distinct parts of the same biological system.

**Abbreviations**

| | |
|---|---|
| ASP | Answer set programming |
| BN | Boolean networks |
| CRN | Chemical reactions network |
| DNF | Disjunctive normal form |
| IG | Interaction graph |
| ODE | Ordinary differential equation |
| PKN | Prior knowledge network |
| SBML | Systems Biology Markup Language |
| SIG | Syntactical interaction graph |
| STG | State transition graph |
| TS | Time-series |

## Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1007/s41109-022-00505-8.

**Additional file 1.** SBML model for the running example.

**Acknowledgements**

A subset of this work was originally presented at the tenth edition of the conference CNA. We are grateful to the attendees, organisers and reviewers of the conference, and thankful for the opportunity to contribute to this special issue. We

also thank Hans-Jörg Schurr for his valuable comments and suggestions on the manuscript, Joachim Niehren for the discussion on semantics of SBML models, and Guilhem Gamard for his precious help on the formalisation of the BN synthesis problem.

**Authors' contributions**
AV prepared the draft manuscript, developed the pipeline and analysed the results. TB and MS supervised the development and analysis. All authors read and approved the final manuscript.

**Availability of data and materials**
All data and programs needed to reproduce the presented results are accessible at https://gitlab.inria.fr/avaginay/CNA2021_extension.

## Declarations

**Ethics approval and consent to participate**
Not applicable.

**Competing interests**
The authors declare that they have no competing interests.

Received: 15 March 2022 Accepted: 5 July 2022
Published online: 26 October 2022

**References**

Aghamiri SS, Delaplace F (2021) TaBooN Boolean network synthesis based on Tabu search. IEEE/ACM Trans Comput Biol Bioinform 19:1. https://doi.org/10.1109/TCBB.2021.3063817

Aghamiri SS, Singh V, Naldi A, Helikar T, Soliman S, Niarakis A (2020) Automated inference of Boolean models from molecular interaction maps using CaSQ. Bioinformatics 36(16):4473–4482. https://doi.org/10.1093/bioinformatics/btaa484

Barman S, Kwon Y-K (2018) A Boolean network inference from time-series gene expression data using a genetic algorithm. Bioinformatics (Oxford, England) 34(17):927–933. https://doi.org/10.1093/bioinformatics/bty584

Biane C, Delaplace F, Melliti T (2018) Abductive network action inference for targeted therapy discovery. Electron Notes Theor Comput Sci 335:3–25. https://doi.org/10.1016/j.entcs.2018.03.006

Bornstein BJ, Keating SM, Jouraku A, Hucka M (2008) LibSBML: an API library for SBML. Bioinformatics 24(6):880–881. https://doi.org/10.1093/bioinformatics/btn051

Calzone L, Fages F, Soliman S (2006) BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge. Bioinformatics 22(14):1805–1807. https://doi.org/10.1093/bioinformatics/btl172

Chatain T, Haar S, Kolčák J, Paulevé L (2020) Most permissive semantics of boolean networks. Research Report, Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400 Talence, France; LSV, ENS Cachan, CNRS, INRIA, Université Paris-Saclay, Cachan (France)

Chevalier S, Froidevaux C, Paulevé L, Zinovyev A (2019) Synthesis of Boolean networks from biological dynamical constraints using answer-set programming. arXiv:1909.04309 [cs, q-bio]

Chevalier S, Noël V, Calzone L, Zinovyev A, Paulevé L (2020) Synthesis and simulation of ensembles of Boolean networks for cell fate decision. In: Abate A, Petrov T, Wolf V (eds) Computational methods in systems biology. Lecture notes in computer science. Springer International Publishing, Cham, pp 193–209. https://doi.org/10.1007/978-3-030-60327-4_11

Conda (2021) Anaconda software distribution

Courtot M, Juty N, Knüpfer C, Waltemath D, Zhukova A, Dräger A, Dumontier M, Finney A, Golebiewski M, Hastings J, Hoops S, Keating S, Kell DB, Kerrien S, Lawson J, Lister A, Lu J, Machne R, Mendes P, Pocock M, Rodriguez N, Villeger A, Wilkinson DJ, Wimalaratne S, Laibe C, Hucka M, Novère NL (2011) Controlled vocabularies and semantics in systems biology. Mol Syst Biol 7(1):543. https://doi.org/10.1038/msb.2011.77

Davidich M, Bornholdt S (2008) The transition from differential equations to Boolean networks: a case study in simplifying a regulatory network model. J Theor Biol 255(3):269–277. https://doi.org/10.1016/j.jtbi.2008.07.020

Dorier J, Crespo I, Niknejad A, Liechti R, Ebeling M, Xenarios I (2016) Boolean regulatory network reconstruction using literature based knowledge with a genetic algorithm optimization method. BMC Bioinform 17:410. https://doi.org/10.1186/s12859-016-1287-z

Fages F, Soliman S (2008a) From reaction models to influence graphs and back: a theorem. In: Fisher J (ed) Formal methods in systems biology. Lecture notes in computer science. Springer, Berlin, pp 90–102

Fages F, Soliman S (2008b) Abstract interpretation and types for systems biology. Theor Comput Sci 403(1):52–70. https://doi.org/10.1016/j.tcs.2008.04.024

Fages F, Gay S, Soliman S (July 2012) Automatic curation of SBML models based on their ODE semantics. Research Report RR-8014, INRIA

Gebser M, Kaminski R, Kaufmann B, Schaub T (2012) Answer set solving in practice. Morgan & Claypool Publishers, Williston

Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, Singhal M, Xu L, Mendes P, Kummer U (2006) COPASI—a COmplex PAthway SImulator. Bioinformatics 22(24):3067–3074. https://doi.org/10.1093/bioinformatics/btl485

Kauffman SA (1969) Metabolic stability and epigenesis in randomly constructed genetic nets. J Theor Biol 22(3):437–467. https://doi.org/10.1016/0022-5193(69)90015-0

Keating SM, Waltemath D, König M, Zhang F, Dräger A, Chaouiya C, Bergmann FT, Finney A, Gillespie CS, Helikar T, Hoops S, Malik-Sheriff RS, Moodie SL, Moraru II, Myers CJ, Naldi A, Olivier BG, Sahle S, Schaff JC, Smith LP, Swat MJ, Thieffry D, Watanabe L, Wilkinson DJ, Blinov ML, Begley K, Faeder JR, Gómez HF, Hamm TM, Inagaki Y, Liebermeister W, Lister AL, Lucio D, Mjolsness E, Proctor CJ, Raman K, Rodriguez N, Shaffer CA, Shapiro BE, Stelling J, Swainston N, Tanimura N, Wagner J, Meier-Schellersheim M, Sauro HM, Palsson B, Bolouri H, Kitano H, Funahashi A, Hermjakob H, Doyle JC, Hucka M (2020) SBML level 3 community members: SBML level 3: an extensible format for the exchange and reuse of biological models. Mol Syst Biol 16(8):9110. https://doi.org/10.15252/msb.20199110

Klarner H, Streck A, Siebert H (2016) PyBoolNet: a python package for the generation, analysis and visualization of Boolean networks. Bioinformatics 33:682. https://doi.org/10.1093/bioinformatics/btw682

Klarner H, Heinitz F, Nee S, Siebert H (2020) Basins of attraction, commitment sets, and phenotypes of Boolean networks. IEEE/ACM Trans Comput Biol Bioinform 17(4):1115–1124. https://doi.org/10.1109/TCBB.2018.2879097

Lähdesmäki H, Shmulevich I, Yli-Harja O (2003) On learning gene regulatory networks under the Boolean network model. Mach Learn 52(1):147–167. https://doi.org/10.1023/A:1023905711304

Liang S, Fuhrman S, Somogyi R (1998) REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. In: Pacific symposium on biocomputing, pp 18–29

Malik-Sheriff RS, Glont M, Nguyen TVN, Tiwari K, Roberts MG, Xavier A, Vu MT, Men J, Maire M, Kananathan S, Fairbanks EL, Meyer JP, Arankalle C, Varusai TM, Knight-Schrijver V, Li L, Dueñas-Roca C, Dass G, Keating SM, Park YM, Buso N, Rodriguez N, Hucka M, Hermjakob H (2020) BioModels—15 years of sharing computational models in life science. Nucleic Acids Res 48(D1):407–415. https://doi.org/10.1093/nar/gkz1055

Mölder F, Jablonski KP, Letcher B, Hall MB, Tomkins-Tinch CH, Sochat V, Forster J, Lee S, Twardziok SO, Kanitz A, Wilm A, Holtgrewe M, Rahmann S, Nahnsen S, Köster J (2021) Sustainable data analysis with Snakemake. F1000 Research 10:33. https://doi.org/10.12688/f1000research.29032.2

Novak B, Pataki Z, Ciliberto A, Tyson JJ (2001) Mathematical model of the cell division cycle of fission yeast. Chaos Interdiscip J Nonlinear Sci 11(1):277. https://doi.org/10.1063/1.1345725

Ostrowski M, Paulevé L, Schaub T, Siegel A, Guziolowski C (2016) Boolean network identification from perturbation time series data combining dynamics abstraction and logic programming. Biosystems 149:139–153. https://doi.org/10.1016/j.biosystems.2016.07.009

Paulevé L, Kolčák J, Chatain T, Haar S (2020) Reconciling qualitative, abstract, and scalable modeling of biological networks. Nat Commun 11(1):4256. https://doi.org/10.1038/s41467-020-18112-5

Schwab JD, Kühlwein SD, Ikonomi N, Kühl M, Kestler HA (2020) Concepts in Boolean network modeling: What do they all mean? Comput Struct Biotechnol J 18:571–582. https://doi.org/10.1016/j.csbj.2020.03.001

Sontag ED (2007) Monotone and near-monotone biochemical networks. Syst Synth Biol 1(2):59–87. https://doi.org/10.1007/s11693-007-9005-9

Thomas R (1973) Boolean formalization of genetic control circuits. J Theor Biol 42(3):563–585. https://doi.org/10.1016/0022-5193(73)90247-6

Vaginay A, Boukhobza T, Smaïl-Tabbone M (2021) Automatic synthesis of Boolean networks from biological knowledge and data. In: Dorronsoro B, Amodeo L, Pavone M, Ruiz P (eds) Optimization and learning. Communications in computer and information science. Springer International Publishing, Cham, pp 156–170. https://doi.org/10.1007/978-3-030-85672-4_12

Vaginay A, Boukhobza T, Smaïl-Tabbone M (2022) From quantitative SBML models to Boolean networks. In: Benito RM, Cherifi C, Cherifi H, Moro E, Rocha LM, Sales-Pardo M (eds) Complex networks and their applications X, vol 1016. Springer International Publishing, Cham, pp 676–687. https://doi.org/10.1007/978-3-030-93413-2_56

Videla S, Guziolowski C, Eduati F, Thiele S, Gebser M, Nicolas J, Saez-Rodriguez J, Schaub T, Siegel A (2015) Learning Boolean logic models of signaling networks with ASP. Theor Comput Sci 599:79–101. https://doi.org/10.1016/j.tcs.2014.06.022

Zhang F, Smith LP, Blinov ML, Faeder J, Hlavacek WS, Juan Tapia J, Keating SM, Rodriguez N, Dräger A, Harris LA, Finney A, Hu B, Hucka M, Meier-Schellersheim M (2020) Systems biology markup language (SBML) level 3 package: multistate, multicomponent and multicompartment species, version 1, release 2. J Integr Bioinform 17(2–3):20200015. https://doi.org/10.1515/jib-2020-0015

## Publisher's Note