



HAL
open science

BoNesis: a Python-based declarative environment for the verification, reprogramming, and synthesis of Most Permissive Boolean networks

Stéphanie Chevalier, Déborah Boyenval, Gustavo Magaña-López, Théo Roncalli, Athénaïs Vaginay, Loïc Paulevé

► To cite this version:

Stéphanie Chevalier, Déborah Boyenval, Gustavo Magaña-López, Théo Roncalli, Athénaïs Vaginay, et al.. BoNesis: a Python-based declarative environment for the verification, reprogramming, and synthesis of Most Permissive Boolean networks. CMSB 2024: 22nd International Conference on Computational Methods in Systems Biology, 2024, Pisa, Italy. 10.1007/978-3-031-71671-3_6 . hal-04629083

HAL Id: hal-04629083

<https://hal.science/hal-04629083v1>

Submitted on 28 Jun 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

BoNesis: a Python-based declarative environment for the verification, reprogramming, and synthesis of Most Permissive Boolean networks

Stéphanie Chevalier^{1,2}, Déborah Boyenval³, Gustavo Magaña-López³, Théo Roncalli³, Athénaïs Vaginay³, and Loïc Paulevé³[0000-0002-7219-2027]

¹ Translational Medicine, Servier, France

² LISN, Univ. Paris-Saclay, CNRS, France

³ Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400 Talence, France

loic.pauleve@labri.fr

Abstract. BoNesis is a Python library which offers a declarative framework for the synthesis of Boolean networks from advanced dynamical properties, such as reachability, bifurcation, minimal trap spaces, stable states, and mutations. It combines recent theoretical advances on Boolean networks with the Most Permissive update mode and efficient resolution of logic programs expressed in Answer-Set Programming. Its main application domain is the inference of Boolean models from bulk and single-cell gene expression data of cell-fate, differentiation and reprogramming processes.

BoNesis is distributed under the GPLv3-compatible free software license CeCILL and is available at <https://bnediction.github.io/bonesis>.

1 Introduction

Boolean networks (BNs) are executable models with an extensive record of applications in biology and medicine, with the modeling of cellular processes such as cell cycle [9, 10, 21], lineage differentiation [14, 20, 22, 32], fate decision [18, 23, 35], and reprogramming [1, 8]. Yet, the design of BNs that are able to reproduce observed behaviors while satisfying desired structural properties (typically on its influence graph), and reflecting background and expert knowledge, is an outstanding challenge at different levels.

First, the model properties need to be specified, both in terms of structure and dynamics. In case of data-driven modeling, e.g. from multiscale sequencing data, a preliminary interpreting step is necessary in order to translate quantitative data into dynamic Boolean properties. This modeling task requires biological and statistical expertise, and an upstream well-defined goal about the BN reconstruction. Moreover, dealing with quantitative observations of the system typically involves uncertainty, e.g. measurement noise or lack of statistical significance for gene activity classification or cell clustering.

Then, provided a specification of what characterizes a good model arises the challenge of constructing such a model. Most of BNs described in systems

biology literature have been designed by hand, either entirely, or by combining previously handmade models. This involves a huge workload, expertise, rules of thumb, trials and errors. Obviously, the automatic construction, also known as logical model inference or *synthesis*, is a long-lived goal [33]. However, the combinatorics of the search space and the complexity of BN verification form a strong bottleneck for realistic applications. Provided a specification and an algorithm for BN synthesis, one would potentially face an issue with a huge number of solutions. Indeed, the experimental observations of cellular dynamics are often derived from few initial conditions and few perturbations with regard to the number of genes, resulting in a largely under-specified synthesis problem. There is consequently a pressing need for efficient computational methods to explore the solution space of BNs.

Recently, there have been several noticeable advances to provide application-focused tools that are tailored for specific type of data and biological and dynamical interpretation of those data [2, 11, 16, 17, 19, 25, 27]. Efforts for providing more generic engines for BN synthesis led to notable frameworks, such as RE:IN [36], involving domain specific language (DSL) and Satisfiability-Modulo-Theory (SMT) technologies, or BRE:IN [13] and AEON [3, 4], involving temporal logics and model-checking technologies.

We present the software BONESIS, which aims at providing a general logical modeling environment and solving engine for the synthesis of BNs from rich dynamical properties. It takes the form of a Python library and offers a declarative interface to specify the synthesis problem, together with various methods for browsing the solution space. The synthesis itself relies on logic Answer-Set Programming (ASP) solving. The flexibility of the framework enables to exploit different types of data (bulk/single-cell RNA-seq, perturbation data) with different interpretations, depending on the biological context of the observations. Moreover, BONESIS can also be employed for the verification and reprogramming of a given BN, to identify mutations enforcing a specified dynamical property. Compared to the prior mentioned softwares, one of the main specificity of BONESIS is that it accounts for the *Most Permissive* (MP) dynamics of BNs, which bring a strong connection with quantitative models and a lower complexity of analysis [29]. BONESIS is available at <https://bnediction.github.io/bonesis>, with tutorials and documentation. It is distributed as part of the CoLoMoto Docker image [26], as well as standard `pip` and `conda` packages. It can be tried online at <https://bnediction.github.io/bonesis/online-demo.html>.

2 Features

BONESIS is dedicated to the synthesis of BNs from the combination of two kinds of inputs (Fig. 1): the *domain* of candidate BNs, describing the model search space, and the *specification of dynamical properties* that must be verified. BONESIS offers various methods to browse the BNs which belong to the input domain and possess the declared dynamical properties: enumeration of solutions, sampling of diverse solutions, projections over subsets of components, enumeration

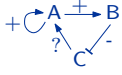
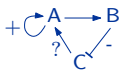
	Domain of Boolean networks (BNs)	Specifications of dynamics
INPUT	<ul style="list-style-type: none"> • Single BN from .bnet, GINsim, ... { • • • } Set of BNs  Influence graph - exact or subgraph - possibly unknown signs <i>Locally-monotone BNs</i>  Partially specified BN from AEON files or API <i>Unate undef/partial functions</i> 	<p>Python-based declarative language with variables</p> <ul style="list-style-type: none"> - existence/absence of MP trajectories - fixed points and (minimal) trap spaces - mutations - link with observations (partial configurations) - universal properties on (reachable) attractors <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <p>E.g.</p> <pre>x = ~obs("init") y = ~obs("steady") x >= fixed(y) with mutant ({"A":0}): x / y</pre> </div> <p>+ Optimisation criteria</p> <ul style="list-style-type: none"> - on satisfiable components - on influence graph complexity
	OUTPUT	<p>Satisfiability at least one BN of the input domain verify the given dynamical properties</p> <hr style="border-top: 1px dashed black;"/> <p>Enumeration:</p> <p>Boolean networks (BNs) can be exported to standard formats and tools</p> <ul style="list-style-type: none"> - non-redundant sampling - sampling with diversity - projections per components <p>Influence graphs of compatible BNs</p> <ul style="list-style-type: none"> - subset minimal - intersection / union over all solutions (cautious/brave reasoning)

Fig. 1. Overview of BoNesis features

of involved influence graphs, etc., with the possibility to specify optimization criteria, such as model size.

The input domain of BNs is typically defined from an influence graph which determines the components of the network and the allowed directed pairwise signed dependencies between them. BoNesis also supports partially defined BNs, thanks to the AEON framework [3]. In these cases, BoNesis will account for any BN matching with the influence graph and partially specified Boolean functions which are locally monotone, i.e. no component has both a positive and negative influence on another one. The domain can also be an explicit set of BNs, or a single BN, without any restriction. This latter case is useful for *verification*, i.e., for checking whether the input BN verifies the specified dynamical properties, and for *reprogramming*, i.e., the computation of permanent mutations ensuring desired dynamical properties [28].

A BN maps each of its components with a Boolean function modeling its target Boolean state with respect to the state of its regulators. A *configuration*

of the BN associates to each component a Boolean state. Then, provided an *update mode*, one can compute possible transitions and trajectories between these configurations. *Attractors* are the smallest sets of configurations from which no transition can leave. They represent the long-term dynamics of the BN, and can either be a single configuration (fixed points), or a set of configurations. In the case of MP update mode, transitions and trajectories can be computed by an iterative process over subcubes. There, the attractors correspond to the *minimal trap spaces* of the BN, that are the smallest subcubes for which no transition can leave, whatever the update mode [24, 29]. It has been show in [29] that MP is able to capture trajectories from any quantitative refinement of the BN, contrary to traditional update modes that can miss possible behaviors.

The expected dynamical properties are specified using predicates, summarized in Table 1. Currently, BONESIS supports properties related to observations, fixed points, minimal trap spaces, existence and absence of reachability between configurations, and universal properties on fixed points and reachable fixed points. The properties can be conditioned with mutations, that replace the function of specified components with a constant value. This enables to specify jointly wild-type and mutant properties. E.g., one can specify that, without any mutation, all the fixed points match with an observation A or B, and that in a mutant where g is knocked out, the fixed points match only with B. The properties can involve partially defined configurations, e.g., by requiring they match with observations over a subset of components, as well as placeholders mutations (**Some** objects). In these cases, BONESIS will search for satisfying assignments for complete configuration and mutations so that the dynamical properties are fulfilled. This is notably employed to identify reprogramming strategies: one can declare that under a **Some** mutation, all fixed points satisfy a given property. Then, BONESIS can enumerate all mutations that can replace **Some**.

In practice, BONESIS builds on the Python language to offer a declarative language for such properties. Fig. 2 illustrates BONESIS usage for the modeling of a simple dynamical property: the reachability of a fixed point matching with a specific marker in the wild-type condition, and the absence of such behavior in a given mutant.

3 Implementation and use cases

BONESIS relies on Answer-Set Programming (ASP) logic framework and the solver CLINGO [12]. When requesting for solutions, BONESIS compiles the predicates in ASP using the encodings defined in [6, 34] and relies on CLINGO’s Python API to perform the different resolution mechanisms (sampling with diversity, projections, etc.). The ASP encodings are designed such that each solution of the logic program corresponds to a distinct BN verifying the input properties.

Currently, for each component, the Boolean functions to synthesize are internally represented in disjunctive normal form (DNF), i.e. by a list of clauses, each clause being a conjunction of at most m literals, where m is the number of regulators of the component. Because the number of clauses grows exponentially

Table 1. Overview of the main predicates to declare dynamical properties in the scope of a Python object `BoNesis(dom, data)`, where `dom` specifies the domain of BNs, including the set of components, and `data` defines the named observations. For a complete reference, see <https://bnediction.github.io/bonesis/language.html>.

Objects	
Observation: maps a subset of components to 0 or 1	
<code>0 = obs({a:0, b:1, ...})</code> <code>0 = obs("name")</code>	Obs from partial mapping of components to 0/1 Named obs defined in the <code>data</code> dictionary
Configuration: maps each component to 0 or 1	
<code>C = cfg()</code> <code>C = +0</code> <code>C = ~0</code>	Allocates a fresh configuration Fresh configuration matching with obs <code>0</code> Default configuration matching with obs <code>0</code>
<code>S = Some(max_size=k)</code>	Represents a mutation of at most <code>k</code> components. Use <code>S.assignments()</code> to get satisfying valuations.
Properties	
Constraints on configurations	
<code>C != C'</code> <code>C[a] == C'[b] 0 1</code> <code>C[a] != C'[b] 0 1</code> <code>C != 0</code>	<code>C</code> and <code>C'</code> differ on at least one component State of component <code>a</code> in <code>C</code> is equal (resp. different) to state of component <code>b</code> in <code>C'</code> (resp. 0 and 1) <code>C</code> does <i>not</i> match with <code>0</code>
Attractor properties	
<code>fixed(C)</code> <code>fixed(0)</code> <code>in_attractor(C)</code>	<code>C</code> is a fixed point there exists a trap space where <code>0</code> is fixed <code>C</code> belongs to an attractor
Reachability	
<code>C >= C' or reach(C, C')</code> <code>C >= 0</code> <code>C >= fixed(0)</code> <i>Note: can be composed, e.g., C1 >= C2 >= fixed(C3)</i> <code>C / □ or nonreach(C, □)</code>	Exists an MP trajectory from <code>C</code> to <code>C'</code> ... to a config matching with <code>0</code> (equiv to <code>C >= +0</code>) ... to a config in a trap space where <code>0</code> is fixed Absence of MP traj from <code>C</code> to ... (same as <code>reach</code>)
Universal properties	
<code>all_fixpoints({0,0',...})</code> <code>C >> "fixpoints" ^ {0,0',...}</code>	All the fps match with at least one given obs All the fps <i>reachable from</i> <code>C</code> ...
Contexts	
<code>with mutant({a:0, ...} S):</code> ...	Properties within the <code>with</code> block are subject to mutation specified by <code>{a:0, ...}</code> (resp. <code>S</code>)

```

import bonesis

# use complete graph as domain
dom = bonesis.InfluenceGraph.complete(["A","B","C"])
# named observations
data = {
  "init": {"A": 0, "B": 0, "C": 0},
  "marker": {"A": 1, "C": 0}
}

bo = bonesis.BoNesis(dom, data)
x = -bo.obs("init")
y = -bo.obs("marker")
x >= bo.fixed(y) # y is a fixed point and can be reached from x
with bo.mutant({"C": 0}):
  x / y

bo.is_satisfiable()
True

solutions = list(bo.boolean_networks(limit=3))
solutions[0] # show one solution
A <- A|(!B&C)
B <- !B|(A&!C)
C <- !B

# list attractors
import pandas as pd
pd.DataFrame(solutions[0].attractors())

  A B C
0 1 1 0

solutions[0].save("solution.bnet")

```

Fig. 2. Example use of BoNESIS in a Jupyter session

with m , BoNESIS allows setting an arbitrary upper bound on the number of clauses for Boolean functions. This enables synthesizing functions that depend on numerous components, but have a limited number of clauses. As shown in [5, 6], BoNESIS can tackle the synthesis of BNs with thousands of components.

BoNESIS has been employed for different concrete biological applications. In [7], authors revisited a published BN of cell fate in cancer (32 components) with ensembles of diverse alternative models synthesized by BoNESIS from the same observations in different mutation conditions. In [30], authors included BoNESIS in a pipeline to infer an ensemble of BNs over 232 genes from gene perturbation experiments for the study of epilepsy. To model early hematopoiesis aging, [15] employed BoNESIS from scRNA-seq data and an expert influence graph (15 components) and results in the discovery of new explanatory regulatory mechanisms. [5] also integrated scRNA-seq with BoNESIS but with a large influence graph from a public database (>1,000 nodes) to identify key genes involved in hematopoiesis and analyze variability of candidate BNs.

4 Conclusion

BoNESIS aims at offering a flexible and efficient framework for BN synthesis from rich dynamical properties, for instance generated from experimental data. Thanks to its flexibility, BoNESIS can also be employed for the verification and reprogramming of BNs. The defined predicates form a domain-specific language that enables to express a large variety of dynamical properties over BNs with a higher abstraction level than classical temporal logic. Moreover, the intertwining with Python greatly facilitates the programmatic generation of the dynamical properties (loops, variables), and the combination with other tools, both as pre-processing of input data and post-processing of obtained models. In future work, we plan to extend support for the synthesis of non-monotone BNs, and for further constraints over attractors and trajectories, notably by generalizing the universal reasoning over minimal trap spaces introduced in [31].

References

1. Abou-Jaoudé, W., Monteiro, P.T., Naldi, A., Grandclaoudon, M., Soumelis, V., Chaouiya, C., Thieffry, D.: Model checking to assess t-helper cell plasticity. *Frontiers in Bioengineering and Biotechnology* **2** (2015). <https://doi.org/10.3389/fbioe.2014.00086>
2. Aghamiri, S.S., Delaplace, F.: Taboon boolean network synthesis based on tabu search. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* p. 1–1 (2021). <https://doi.org/10.1109/TCBB.2021.3063817>
3. Beneš, N., Brim, L., Kadlecak, J., Pastva, S., Šafránek, D.: AEON: Attractor bifurcation analysis of parametrised boolean networks. In: *Computer Aided Verification*, pp. 569–581. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-53288-8_28
4. Beneš, N., Brim, L., Huvar, O., Pastva, S., Šafránek, D.: Boolean network sketches: a unifying framework for logical model inference. *Bioinformatics* **39**(4) (2023). <https://doi.org/10.1093/bioinformatics/btad158>
5. Chevalier, S.: Inférence logique de réseaux booléens à partir de connaissances et d'observations de processus de différenciation cellulaire. Ph.D. thesis, Université Paris-Saclay (Sep 2022), https://theses.hal.science/tel-03917566/file/106218_CHEVALIER_2022_archivage.pdf
6. Chevalier, S., Froidevaux, C., Paulevé, L., Zinovyev, A.: Synthesis of boolean networks from biological dynamical constraints using answer-set programming. In: *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. pp. 34–41. IEEE (2019). <https://doi.org/10.1109/ICTAI.2019.00014>
7. Chevalier, S., Noël, V., Calzone, L., Zinovyev, A., Paulevé, L.: Synthesis and Simulation of Ensembles of Boolean Networks for Cell Fate Decision. In: *CMSB 2020 - 18th International Conference on Computational Methods in Systems Biology. Lecture Notes in Computer Science*, vol. 12314, pp. 193–209. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60327-4_11
8. Collombet, S., van Oevelen, C., Sardina Ortega, J.L., Abou-Jaoudé, W., Di Stefano, B., Thomas-Chollier, M., Graf, T., Thieffry, D.: Logical modeling of lymphoid and myeloid cell specification and transdifferentiation. *Proceedings of the National Academy of Sciences* **114**(23), 5792–5799 (2017). <https://doi.org/10.1073/pnas.1610622114>
9. Davidich, M.I., Bornholdt, S.: Boolean network model predicts cell cycle sequence of fission yeast. *PLoS ONE* **3**(2), e1672 (2008). <https://doi.org/10.1371/journal.pone.0001672>
10. 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), e124–e131 (2006). <https://doi.org/10.1093/bioinformatics/btl210>
11. Gao, S., Sun, C., Xiang, C., Qin, K., Lee, T.H.: Learning asynchronous Boolean networks from single-cell data using multiobjective cooperative genetic programming. *IEEE Trans. Cybern.* **52**(5), 2916–2930 (2022). <https://doi.org/10.1109/TCYB.2020.3022430>
12. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Clingo = ASP + control: Preliminary report. *CoRR* (2014). <https://doi.org/10.48550/arXiv.1405.3694>
13. Goldfeder, J., Kugler, H.: BRE:IN - a backend for reasoning about interaction networks with temporal logic. In: *Computational Methods in Systems Biology*, pp. 289–295. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-31304-3_15

14. Hamey, F.K., Nestorowa, S., Kinston, S.J., Kent, D.G., Wilson, N.K., Göttgens, B.: Reconstructing blood stem cell regulatory network models from single-cell molecular profiles. *Proceedings of the National Academy of Sciences* **114**(23), 5822–5829 (2017). <https://doi.org/10.1073/pnas.1610609114>
15. Héroult, L., Poplineau, M., Duprez, E., Remy, É.: A novel Boolean network inference strategy to model early hematopoiesis aging. *Computational and Structural Biotechnology Journal* **21**, 21–33 (2023). <https://doi.org/10.1016/j.csbj.2022.10.040>
16. Heydari, T., Langley, M.A., Fisher, C.L., Aguilar-Hidalgo, D., Shukla, S., Yachie-Kinoshita, A., Hughes, M., McNagny, K.M., Zandstra, P.W.: Iqcell: A platform for predicting the effect of gene perturbations on developmental trajectories using single-cell rna-seq data. *PLOS Computational Biology* **18**(2), e1009907 (2022). <https://doi.org/10.1371/journal.pcbi.1009907>
17. Hung-Cuong, T., Yung-Keun, K.: Cga-bni: A novel constrained genetic algorithm-based boolean network inference method from steady-state gene expression data. *Bioinformatics* **37**, i383–i391 (2021). <https://doi.org/10.1093/bioinformatics/btab295>
18. Ikononi, N., Kühlwein, S.D., Schwab, J.D., Kestler, H.A.: Awakening the HSC: Dynamic Modeling of HSC Maintenance Unravels Regulation of the TP53 Pathway and Quiescence. *Frontiers in Physiology* **11** (2020). <https://doi.org/10.3389/fphys.2020.00848>
19. Liu, X., Wang, Y., Shi, N., Ji, Z., He, S.: Gapore: Boolean network inference using a genetic algorithm with novel polynomial representation and encoding scheme. *Knowledge-Based Systems* **228**, 107277 (2021). <https://doi.org/10.1016/j.knosys.2021.107277>
20. Martínez-Sosa, P., Mendoza, L.: The regulatory network that controls the differentiation of t lymphocytes. *Biosystems* **113**(2), 96–103 (2013). <https://doi.org/10.1016/j.biosystems.2013.05.007>
21. Meyer, P., Maity, P., Burkovski, A., Schwab, J., Müssel, C., Singh, K., Ferreira, F.F., Krug, L., Maier, H.J., Wlaschek, M., Wirth, T., Kestler, H.A., Scharffetter-Kochanek, K.: A model of the onset of the senescence associated secretory phenotype after dna damage induced senescence. *PLOS Computational Biology* **13**(12), e1005741 (2017). <https://doi.org/10.1371/journal.pcbi.1005741>
22. Moignard, V., Woodhouse, S., Haghverdi, L., Lilly, A.J., Tanaka, Y., Wilkinson, A.C., Buettner, F., Macaulay, I.C., Jawaid, W., Diamanti, E., Nishikawa, S.I., Piterman, N., Kouskoff, V., Theis, F.J., Fisher, J., Göttgens, B.: Decoding the regulatory network of early blood development from single-cell gene expression measurements. *Nature biotechnology* **33**(3), 269–276 (2015). <https://doi.org/10.1038/nbt.3154>, pMC4374163[pmcid]
23. Montagud, A., Béal, J., Tobalina, L., Traynard, P., Subramanian, V., Szalai, B., Alföldi, R., Puskás, L., Valencia, A., Barillot, E., Saez-Rodriguez, J., Calzone, L.: Patient-specific boolean models of signalling networks guide personalised treatments. *eLife* **11** (2022). <https://doi.org/10.7554/elife.72626>
24. Moon, K., Lee, K., Paulevé, L.: Computational complexity of minimal trap spaces in boolean networks. *ArXiv e-prints* (2022). <https://doi.org/10.48550/ARXIV.2212.12756>
25. Muñoz, S., Carrillo, M., Azpeitia, E., Rosenblueth, D.A.: Griffin: A tool for symbolic inference of synchronous boolean molecular networks. *Frontiers in Genetics* **9** (2018). <https://doi.org/10.3389/fgene.2018.00039>

26. Naldi, A., Hernandez, C., Levy, N., Stoll, G., Monteiro, P.T., Chaouiya, C., Helikar, T., Zinovyev, A., Calzone, L., Cohen-Boulakia, S., Thieffry, D., Paulevé, L.: The CoLoMoTo Interactive Notebook: Accessible and Reproducible Computational Analyses for Qualitative Biological Networks. *Frontiers in Physiology* **9**, 680 (2018). <https://doi.org/10.3389/fphys.2018.00680>
27. Palli, R., Palshikar, M.G., Thakar, J.: Executable pathway analysis using ensemble discrete-state modeling for large-scale data. *PLoS Comput Biol* **15**(9), e1007317 (2019). <https://doi.org/10.1371/journal.pcbi.1007317>
28. Paulevé, L.: Marker and source-marker reprogramming of Most Permissive Boolean networks and ensembles with BoNesis . *Peer Community Journal* **3**, e30 (2023). <https://doi.org/10.24072/pcjournal.255>
29. Paulevé, L., Kolčák, J., Chatain, T., Haar, S.: Reconciling qualitative, abstract, and scalable modeling of biological networks. *Nature Communications* **11**(1), 4256 (2020). <https://doi.org/10.1038/s41467-020-18112-5>
30. Réda, C., Delahaye-Duriez, A.: Prioritization of candidate genes through boolean networks. In: *Computational Methods in Systems Biology*, pp. 89–121. Springer International Publishing (2022). https://doi.org/10.1007/978-3-031-15034-0_5
31. Riva, S., Lagniez, J.M., López, G.M., Paulevé, L.: Tackling universal properties of minimal trap spaces of boolean networks. In: *CMSB 2023: Proceedings of the 21st International Conference on Computational Methods in Systems Biology* (2023). <https://doi.org/10.48550/arXiv.2305.02442>
32. Schwab, J.D., Ikonomi, N., Werle, S.D., Weidner, F.M., Geiger, H., Kestler, H.A.: Reconstructing boolean network ensembles from single-cell data for unraveling dynamics in the aging of human hematopoietic stem cells. *Computational and Structural Biotechnology Journal* **19**, 5321–5332 (2021). <https://doi.org/10.1016/j.csbj.2021.09.012>
33. Thieffry, D., Thomas, R.: Dynamical behaviour of biological regulatory networks—ii. immunity control in bacteriophage lambda. *Bulletin of Mathematical Biology* **57**, 277–297 (1995). <https://doi.org/10.1007/BF02460619>
34. Trinh, V.G., Benhamou, B., Paulevé, L.: mpbn: a simple tool for efficient edition and analysis of elementary properties of Boolean networks. *arXiv* (2024)
35. Werle, S.D., Schwab, J.D., Tatura, M., Kirchhoff, S., Szekely, R., Diels, R., Ikonomi, N., Sipos, B., Sperveslage, J., Gress, T.M., Buchholz, M., Kestler, H.A.: Unraveling the molecular tumor-promoting regulation of cofilin-1 in pancreatic cancer. *Cancers* **13**(4), 725 (2021). <https://doi.org/10.3390/cancers13040725>
36. Yordanov, B., Dunn, S.J., Kugler, H., Smith, A., Martello, G., Emmott, S.: A method to identify and analyze biological programs through automated reasoning. *npj Systems Biology and Applications* **2**(1), 1–16 (2016). <https://doi.org/10.1038/npsba.2016.10>