Modeling and Analysis of Large Biological Regulatory Networks thanks to the "Process Hitting" Framework

Maxime FOLSCHETTE[1]
maxime.folschette@irccyn.ec-nantes.fr
http://www.irccyn.ec-nantes.fr/~folschet/

Joint work with: Olivier ROUX[1], Morgan MAGNIN[1], Loïc PAULEVÉ[2]

[1]: MeForBio team / IRCCyN / École Centrale de Nantes (Nantes, France)
morgan.magnin@irccyn.ec-nantes.fr
olivier.roux@irccyn.ec-nantes.fr

[2]: AMIB team / LIX / École Polytechnique (Palaiseau, France)
pauleve@lix.polytechnique.fr

AtlanSTIC sojourn financed by NII & Centrale Initiatives

The MeForBio Team

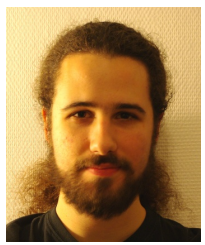
MeForBio (team): formal methods for bioinformatics
IRCCyN (laboratory): domains related to computer science
École Centrale de Nantes (engineer school)

Olivier ROUX
Professor & team leader

Morgan MAGNIN
Associate professor

Philippe BORDRON
Post-doc

Julien GRAS
Research engineer



Maxime FOLSCHETTE
PhD student

Loïc PAULEVÉ
Post-doc at LIX
former member of MeForBio

The MeForBio Team

Team activities (computer science / bioinformatics):

- Using formal & algebraic models to study the dynamics of biological systems
- Projects: CirClock (CNRS), BioTempo (French agency ANR), BIL (region)
- Collaborations: several projects with German teams, Microsoft Research (Cambridge)

Loïc PAULEVÉ's PhD thesis: [Paulevé11]

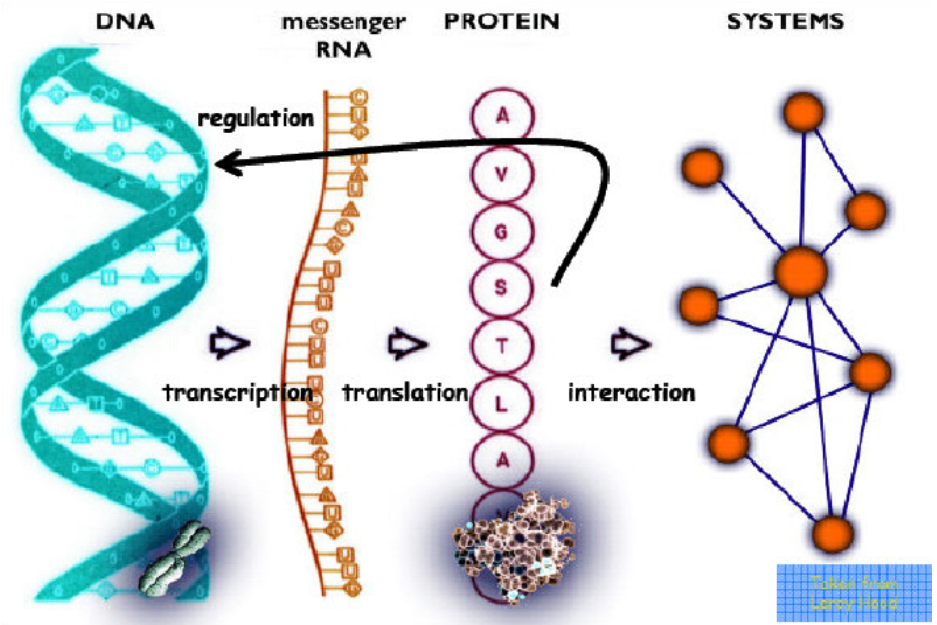
- Modeling, Simulation and Checking of Large Biological Regularoty Networks with Process Hitting
  - → Discrete approach, chronological data

My PhD subject:

- Adding a temporal dimension to the Process Hitting by introducing chronometric data
  - → Discrete approach, continuous data/constraints

Context and Aims

Algebraic modeling to study complex dynamical biological systems:



Context and Aims

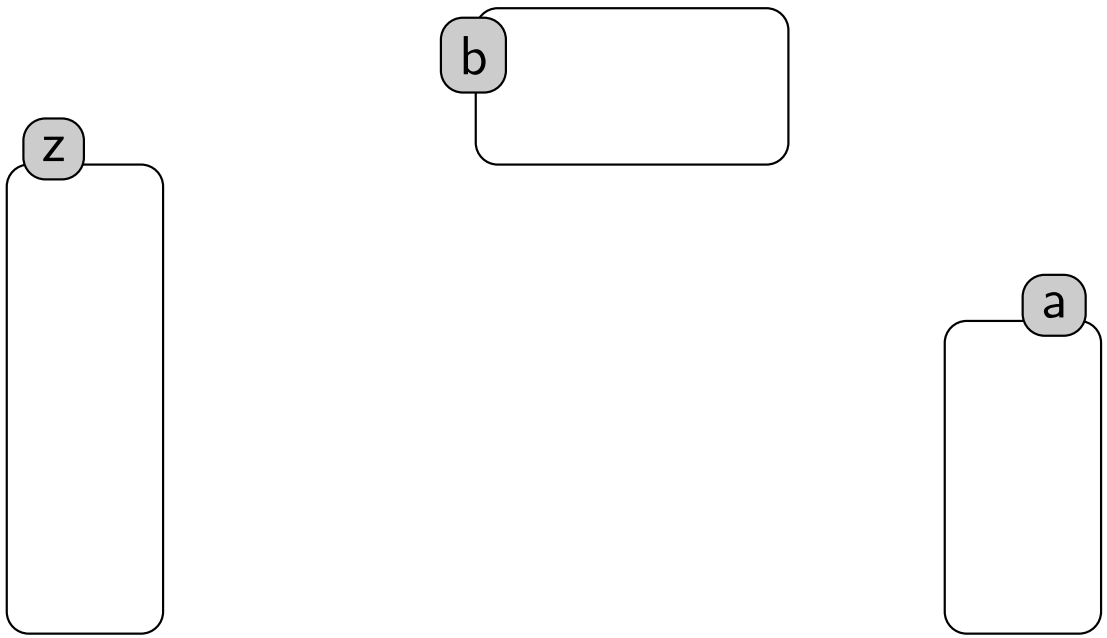
Algebraic modeling to study complex dynamical biological systems:

Some related works:

- Timed Petri Nets [Heiner, ...]
- Continuous-Time Markov Chains (PRISM) [Kwiatkowska, Parker, ...]
- Biocham [Fages]
- Kappa [Danos, Feret, Fontana, Harmer, Krivine]
- Timed Automata [Siebert, Bockmayr, ...]
- Linear Hybrid Automata [Ahmad, Roux]
- Hoare logic [Bernot, Comet, Roux]
- ...

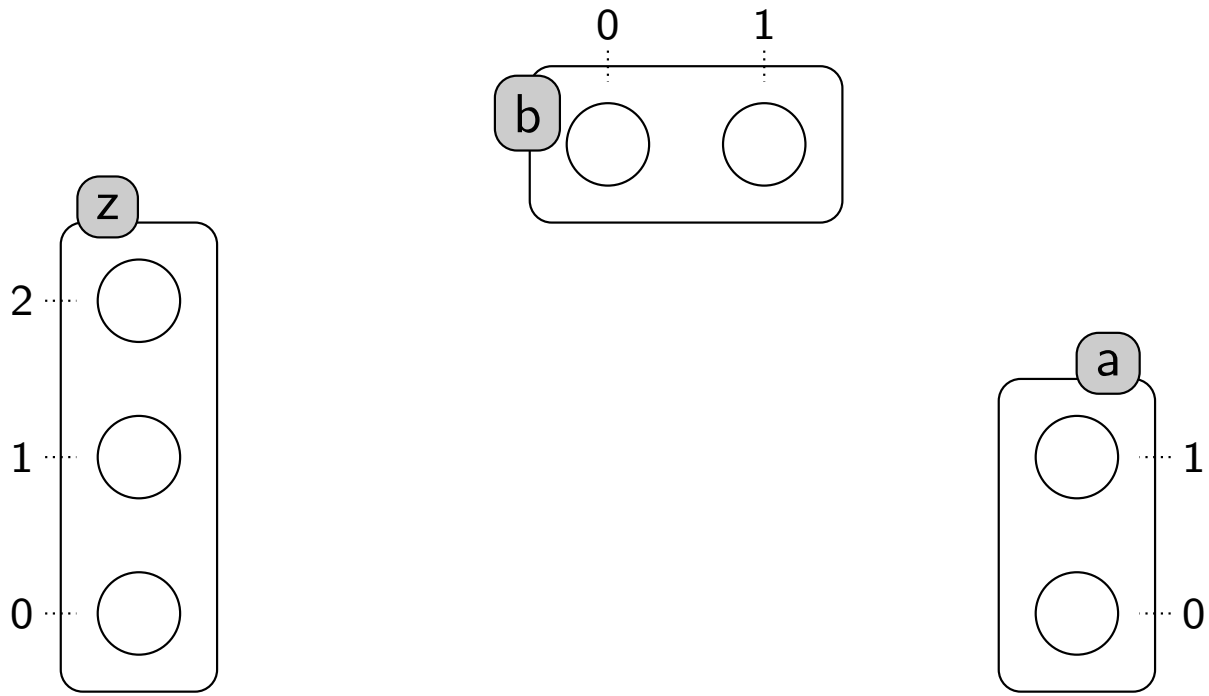
$\rightarrow$ How to cope with **very large systems** & combinatorial explosion?

The Process Hitting Framework



Sorts: components a, b, z

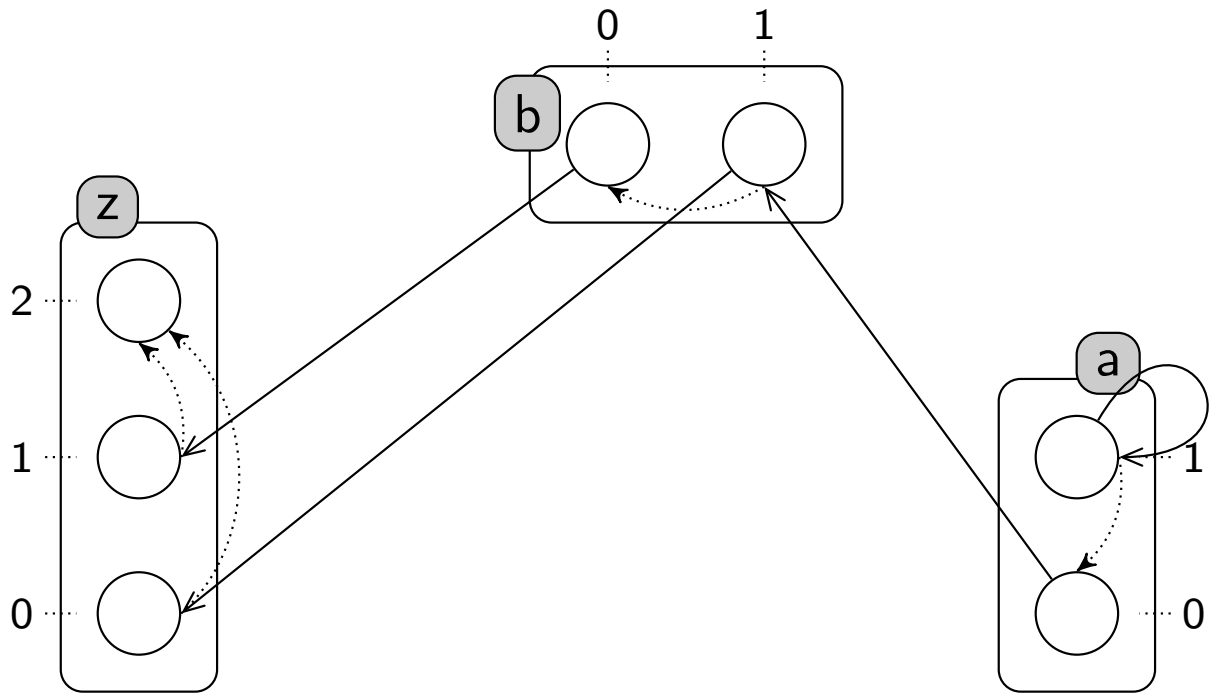
The Process Hitting Framework



Sorts: components a, b, z

Processes: local states / levels of expression 0, 1, 2

The Process Hitting Framework

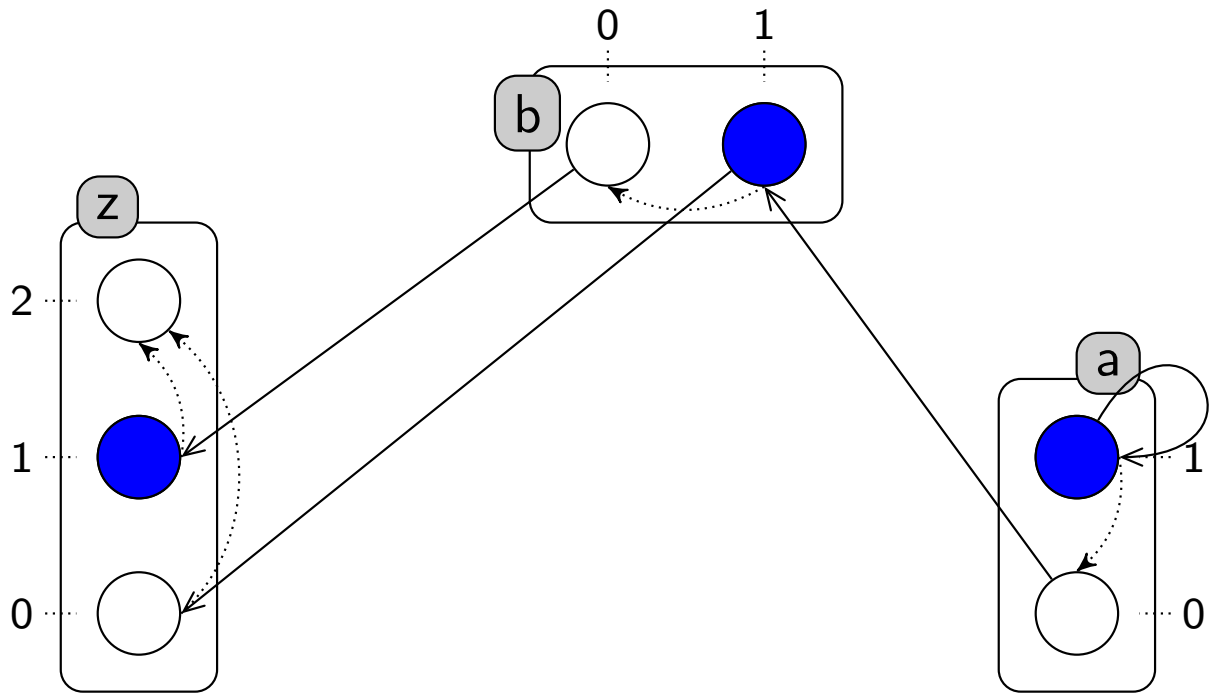


Sorts: components a, b, z

Processes: local states / levels of expression $0, 1, 2$

Actions: dynamics $a_0 \to b_1 \curvearrowright b_0$

The Process Hitting Framework



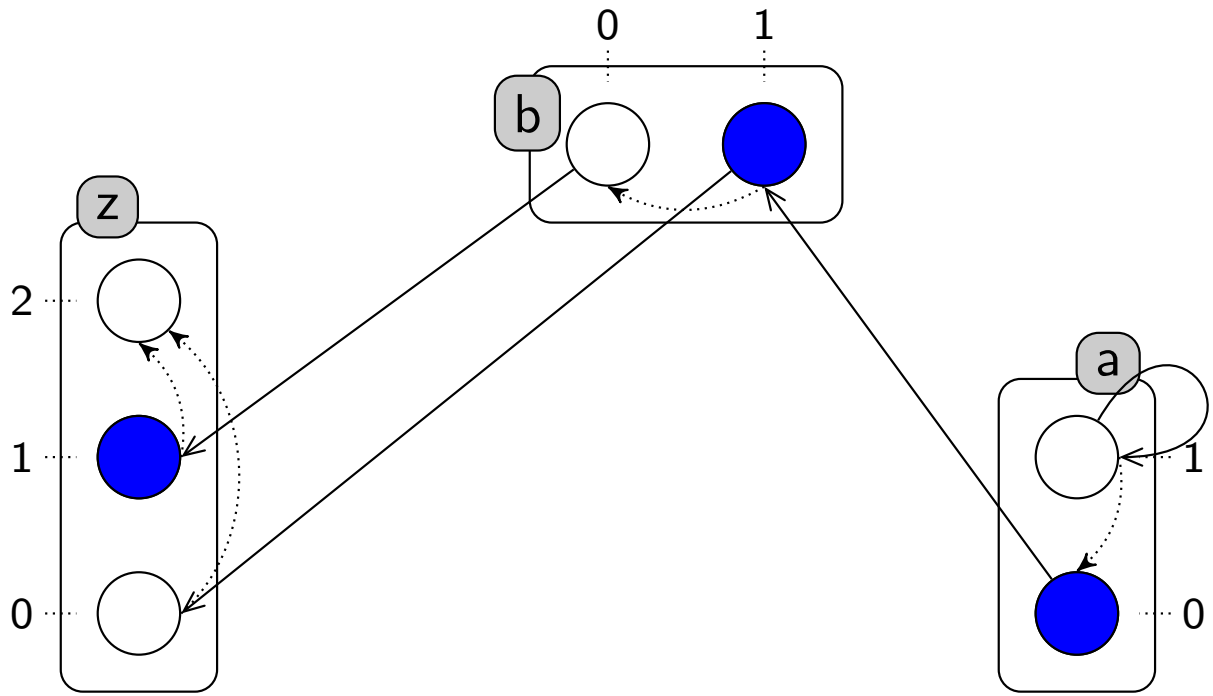
Sorts: components $a,\ b,\ z$

Processes: local states / levels of expression $0,\ 1,\ 2$

Actions: dynamics $a_0 \rightarrow b_1 \upharpoonright b_0$

States: sets of active processes $\langle a_1, b_1, z_1 \rangle$

The Process Hitting Framework



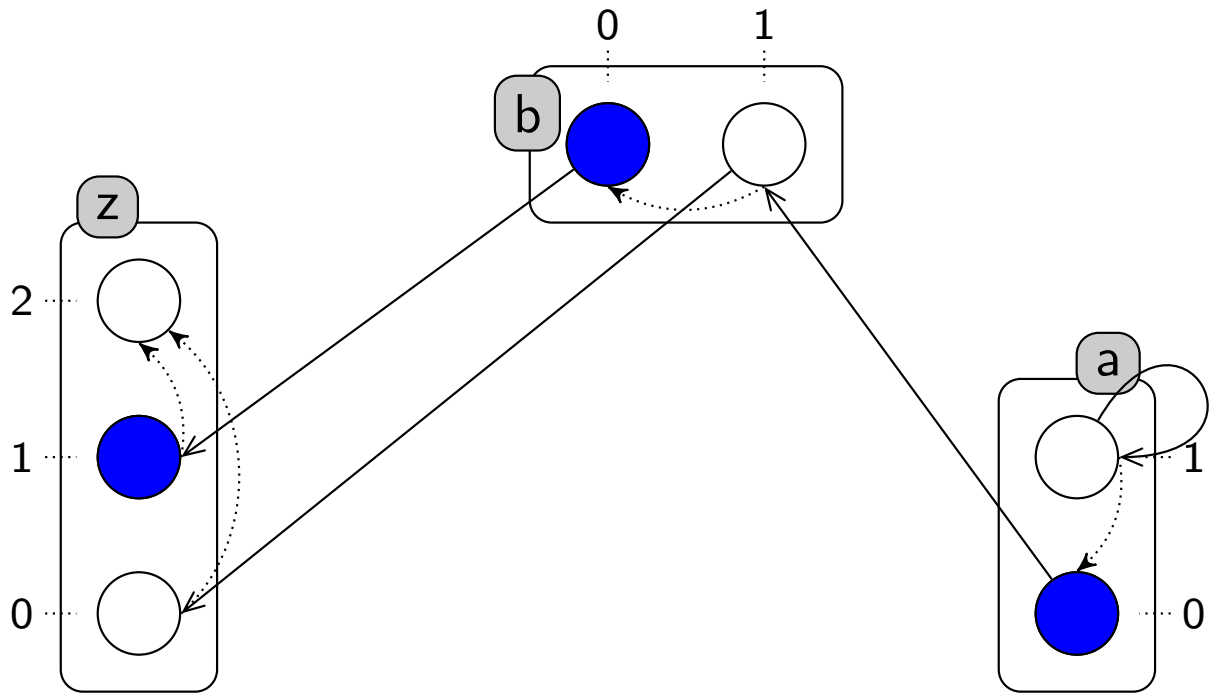
Sorts: components $a,\ b,\ z$

Processes: local states / levels of expression $0,\ 1,\ 2$

Actions: dynamics $a_0 \to b_1 \curvearrowright b_0$

States: sets of active processes $\langle a_0, b_1, z_1 \rangle$

The Process Hitting Framework



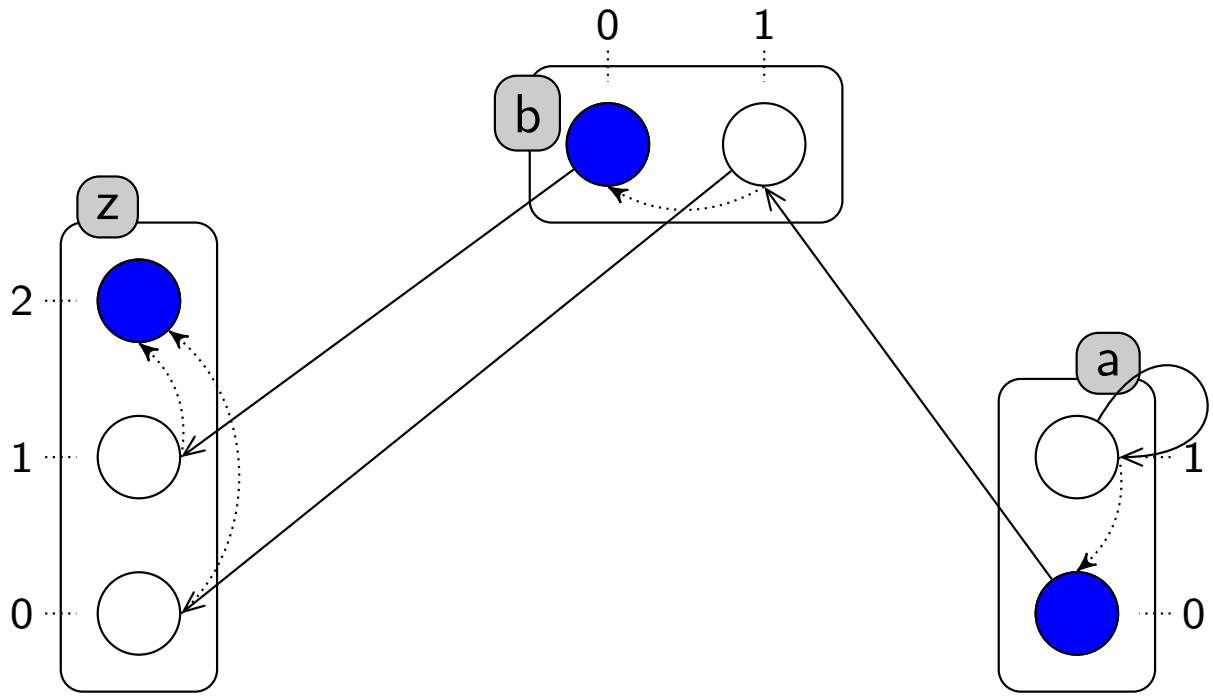
Sorts: components $a,\ b,\ z$

Processes: local states / levels of expression $0,\ 1,\ 2$

Actions: dynamics $a_0 \rightarrow b_1 \curvearrowright b_0$

States: sets of active processes $\langle a_0, b_0, z_1 \rangle$

The Process Hitting Framework



Sorts: components $a,\ b,\ z$

Processes: local states / levels of expression $0,\ 1,\ 2$

Actions: dynamics $a_0 \to b_1 \upharpoonright b_0$

States: sets of active processes $\langle a_0, b_0, z_2 \rangle$

Stochastic Features

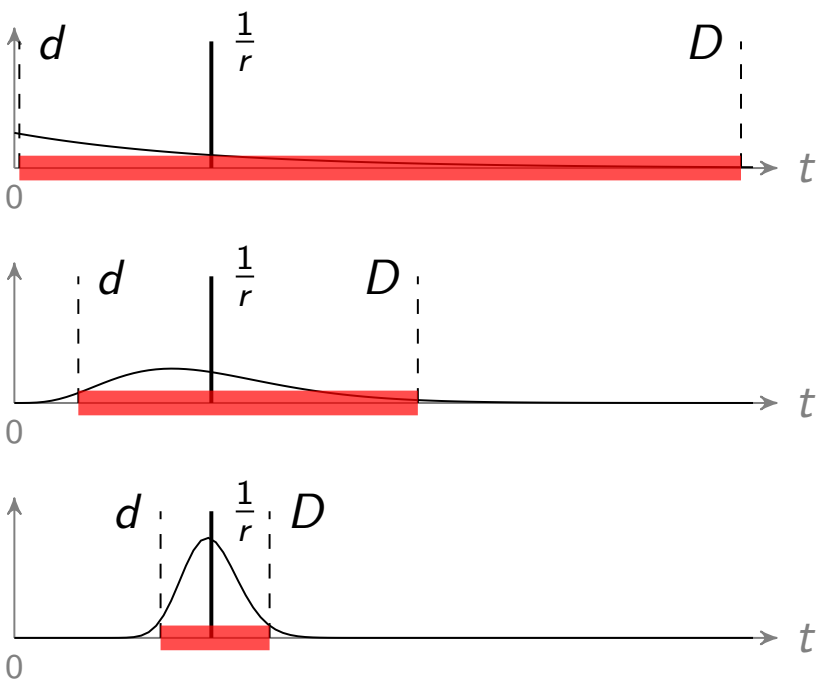
[PMR10-TSE]

- Introduces time features
- Parameters: either (r, sa), or the **firing interval** $[d; D]$.
  $\rightarrow$ Tests by simulation

Stochastic Features

[PMR10-TSE]

- Introduces time features
- Parameters: either (r, sa), or the **firing interval** $[d; D]$.
  $\rightarrow$ Tests by simulation

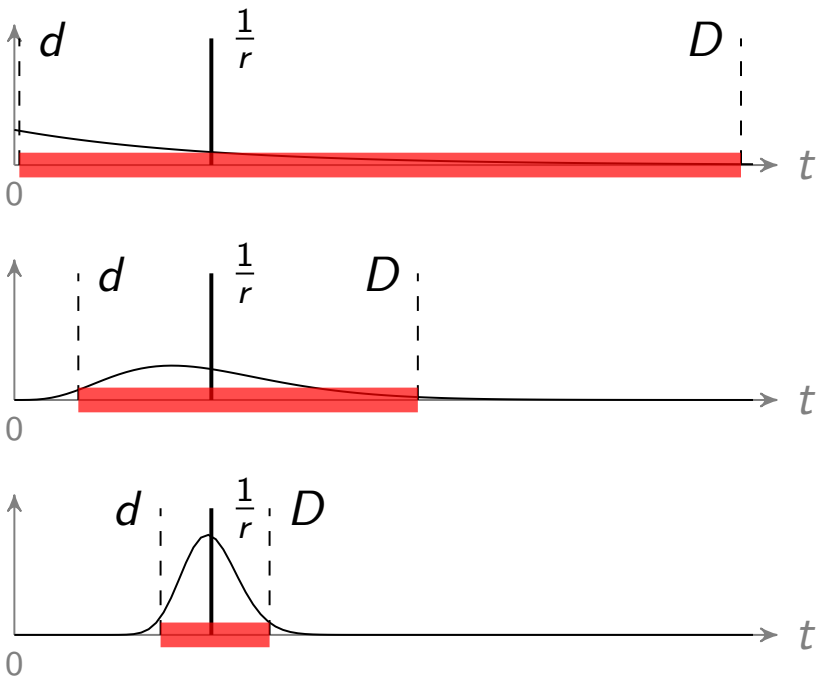


Stochastic Features

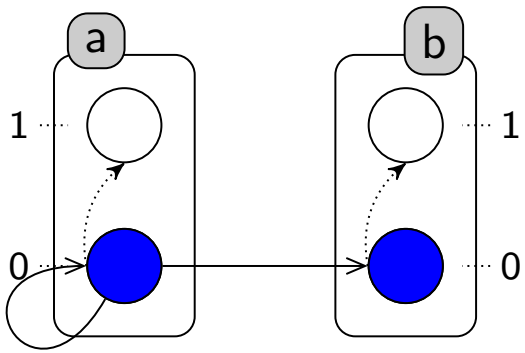
[PMR10-TSE]

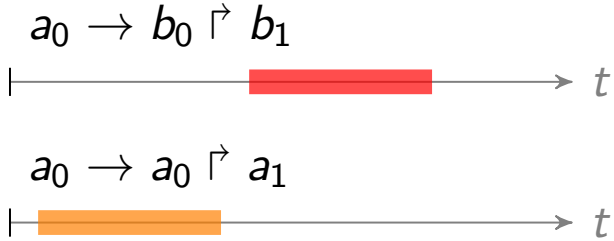
- Introduces time features
- Parameters: either (r, sa), or the **firing interval** $[d; D]$.

  $\rightarrow$ Tests by simulation



action duration





$\rightarrow$ b_1 reached with a **very low probability**.

Static Analysis

Problem: the simulation still faces combinatorial explosion

Idea: study the model **without running it**

$\rightarrow$ avoiding combinatorial explosion

$\rightarrow$ focusing on the interesting parts of the model

Two main results:

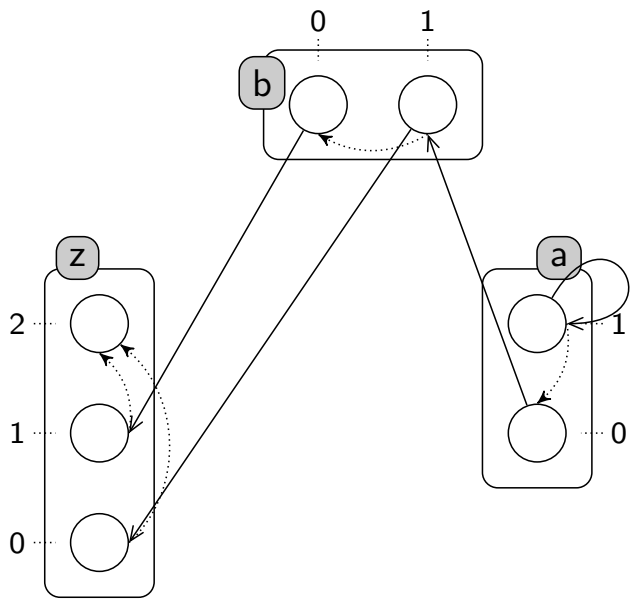
- **Fixed points**
  $\rightarrow$ Deadlocks
- **Process reachability**
  $\rightarrow$ From a state where we have processes p_1, p_2, ... can we reach the process q_1, then q_2, then ... ?

Static Analysis: Fixed Points

[PRM10-TCSB]

Fixed point = state where no action can be fired

$\rightarrow$ avoid couples of processes bounded by an action



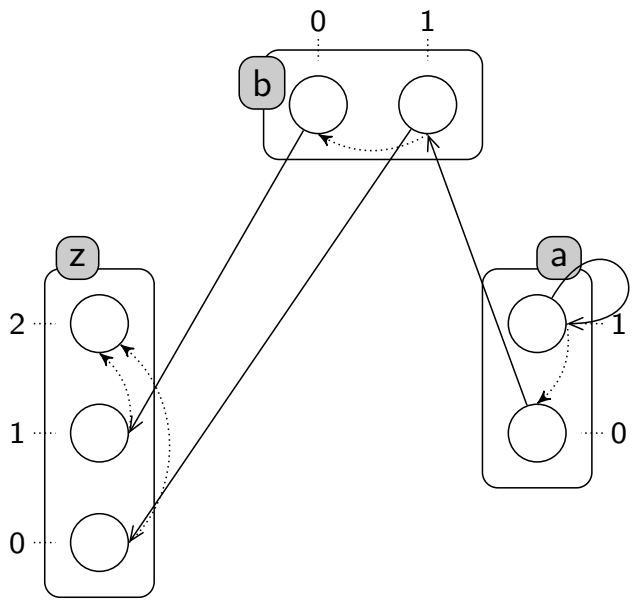
Static Analysis: Fixed Points

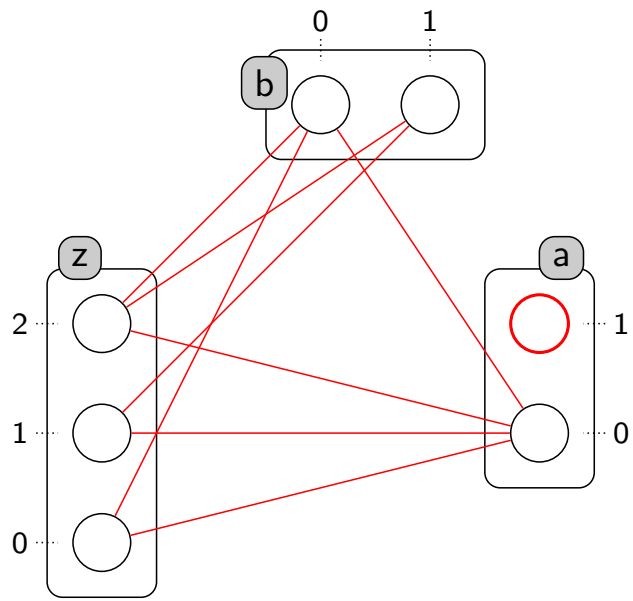
[PRM10-TCSB]

Fixed point = state where no action can be fired

→ avoid couples of processes bounded by an action

→ Hitless Graph





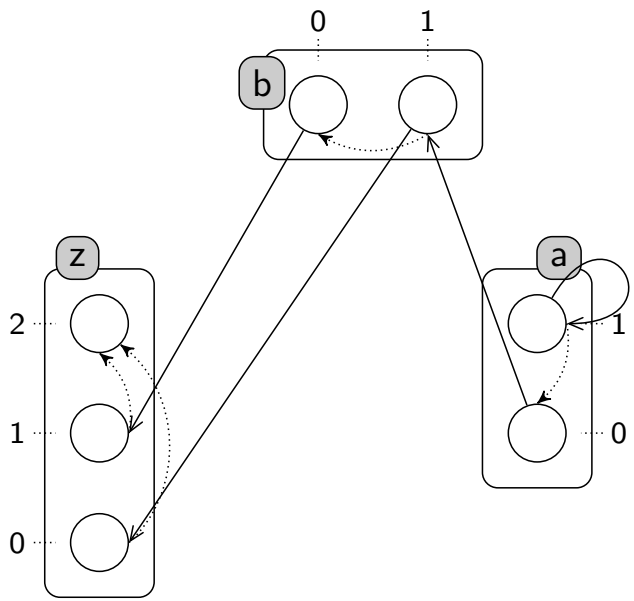
Static Analysis: Fixed Points

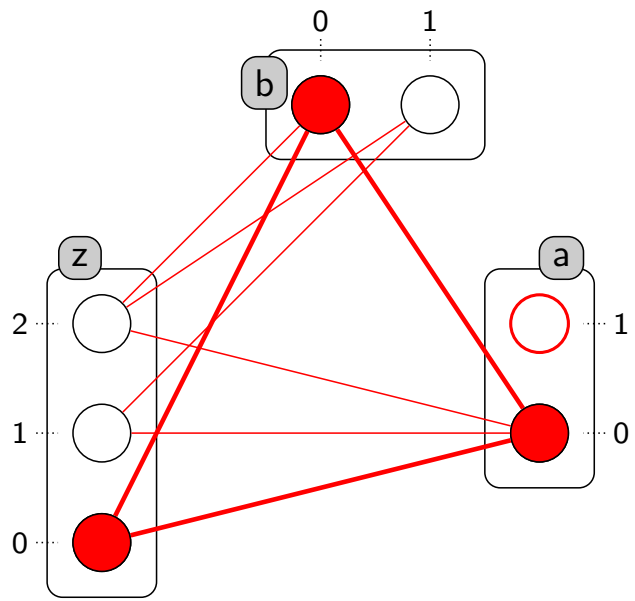
[PRM10-TCSB]

Fixed point = state where no action can be fired

→ avoid couples of processes bounded by an action

→ Hitless Graph → **n-cliques** = fixed points





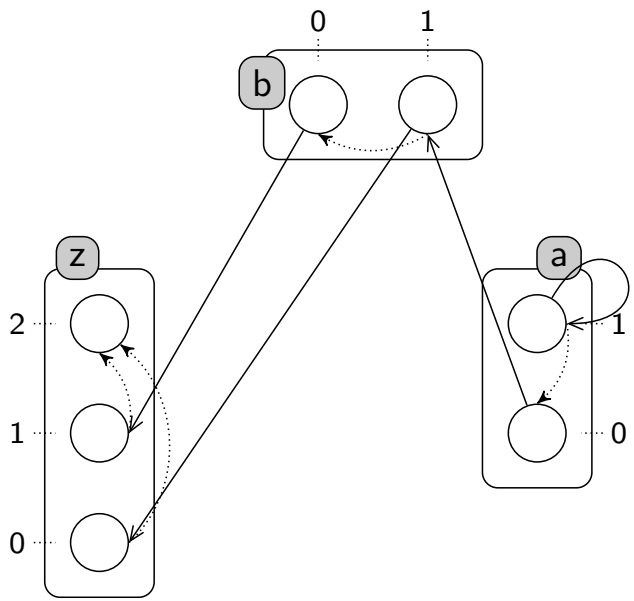
Static Analysis: Fixed Points

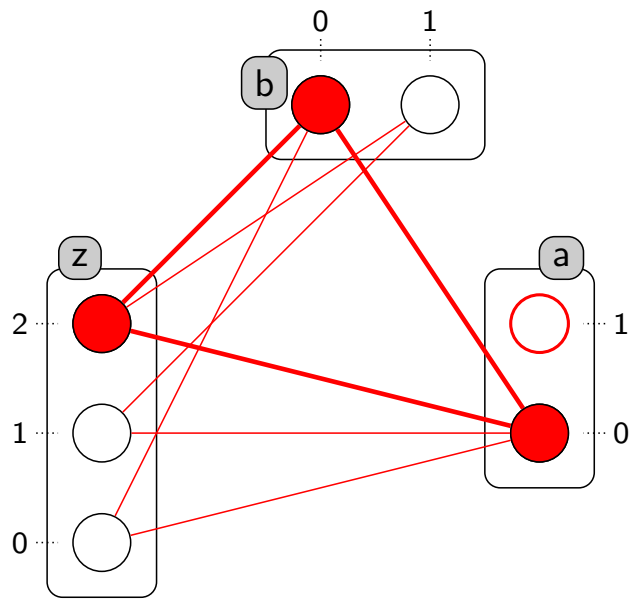
[PRM10-TCSB]

Fixed point = state where no action can be fired

→ avoid couples of processes bounded by an action

→ Hitless Graph → **n-cliques** = fixed points





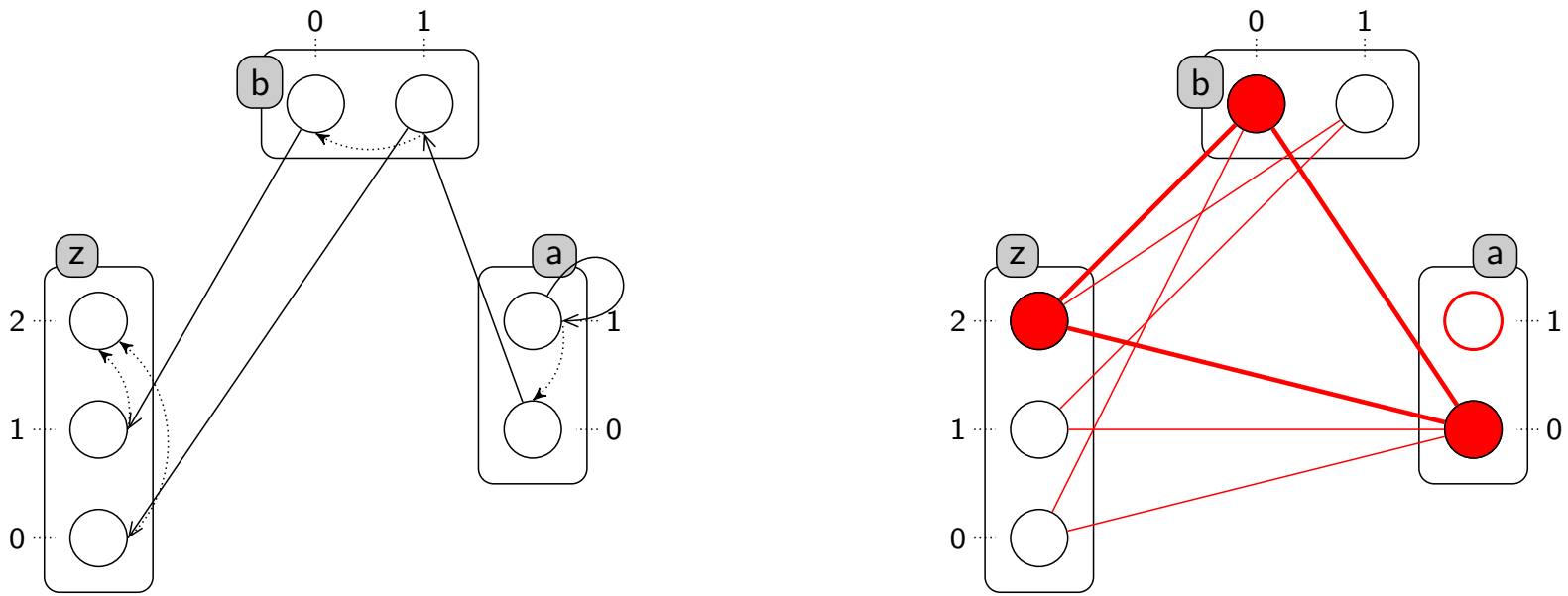
Static Analysis: Fixed Points

[PRM10-TCSB]

Fixed point = state where no action can be fired

→ avoid couples of processes bounded by an action

→ Hitless Graph → **n-cliques** = fixed points

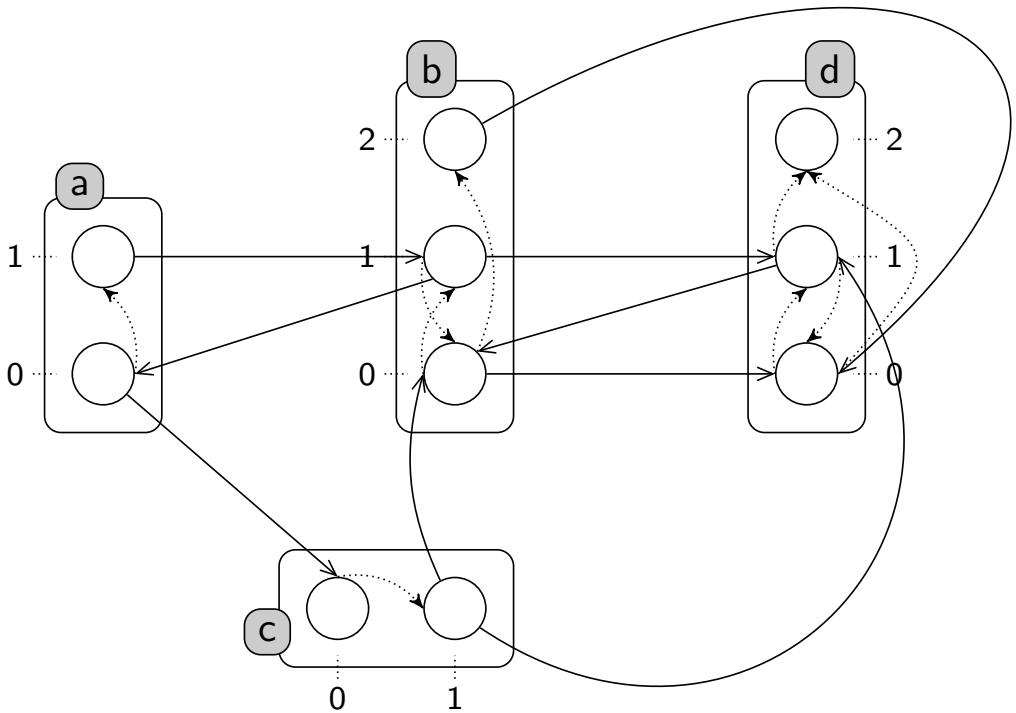


Exponential complexity w.r.t. the number of sorts

Static Analysis: Successive Reachability

[PMR12-MSCS]

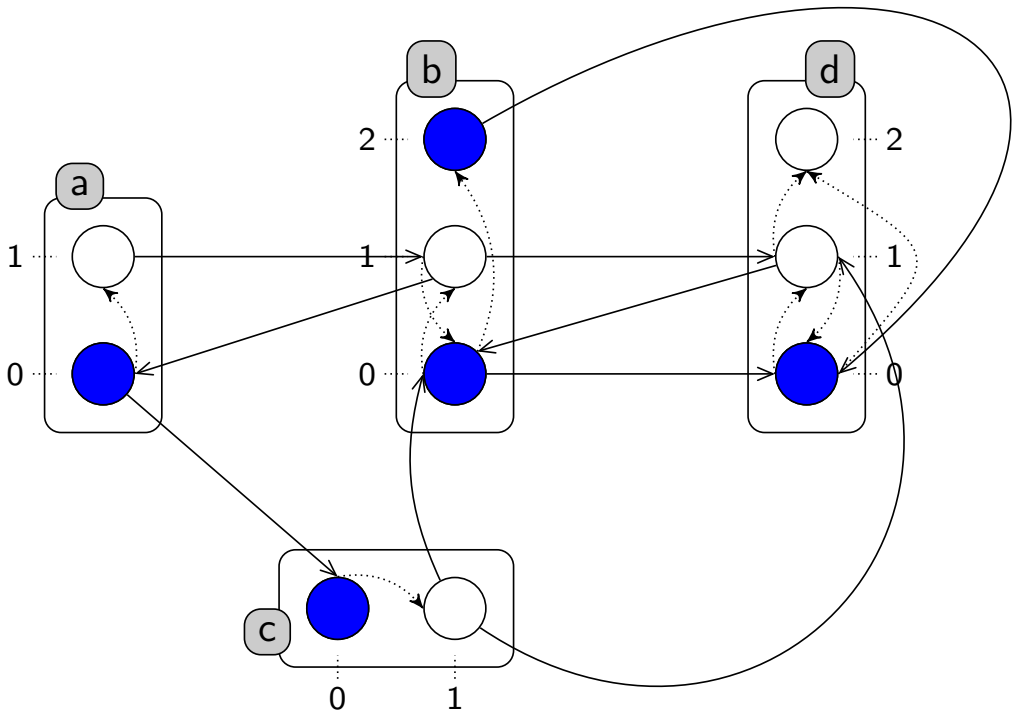
Successive reachability of processes:



Static Analysis: Successive Reachability

[PMR12-MSCS]

Successive reachability of processes:



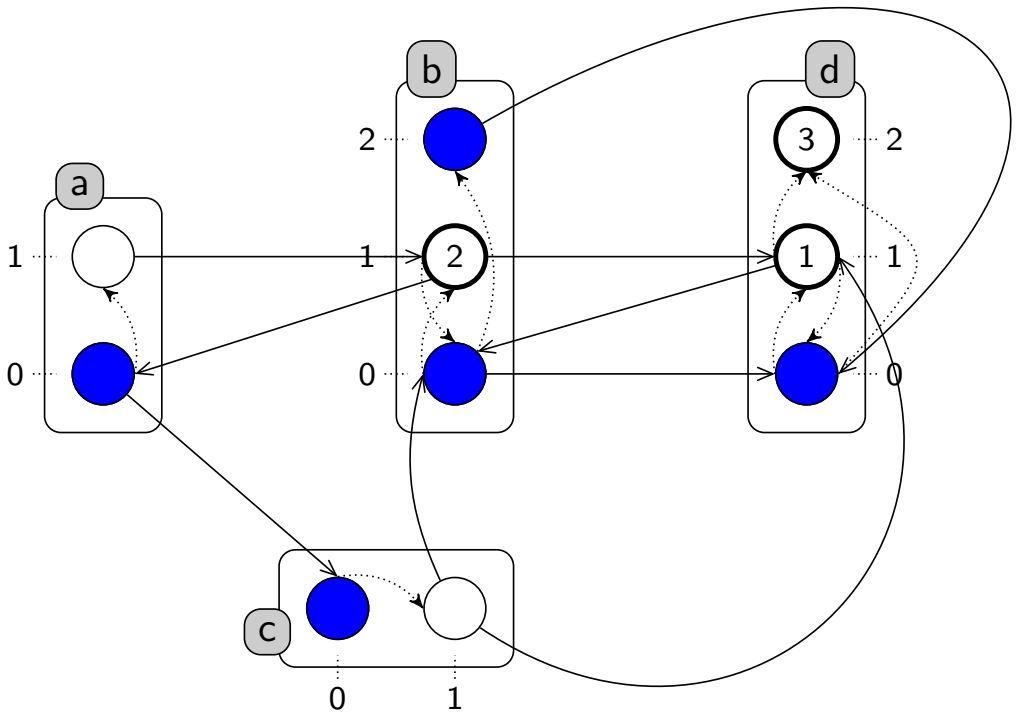
- Initial context

$$\langle a_1, \{b_0, b_1\}, c_0, z_0 \rangle$$

Static Analysis: Successive Reachability

[PMR12-MSCS]

Successive reachability of processes:



- Initial context

  $\langle a_1, \{b_0, b_1\}, c_0, z_0 \rangle$

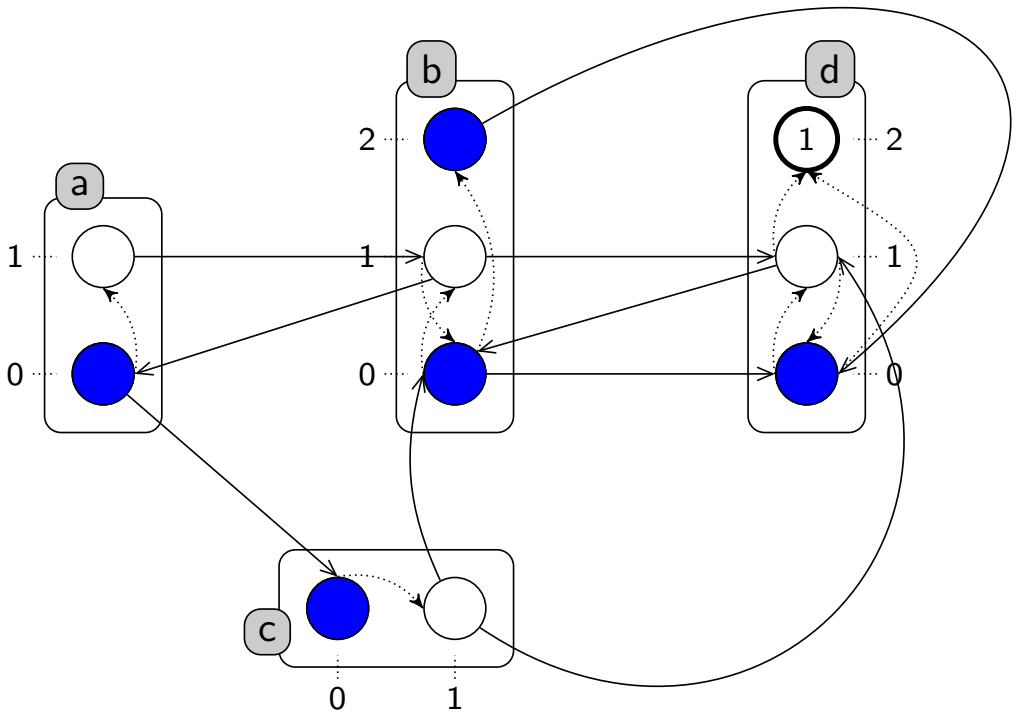
- Objectives

  $d_0 \Rsh^* d_1 :: b_0 \Rsh^* b_1 :: d_1 \Rsh^* d_2$

Static Analysis: Successive Reachability

[PMR12-MSCS]

Successive reachability of processes:



- Initial context

$$\langle a_1, \{b_0, b_1\}, c_0, z_0 \rangle$$

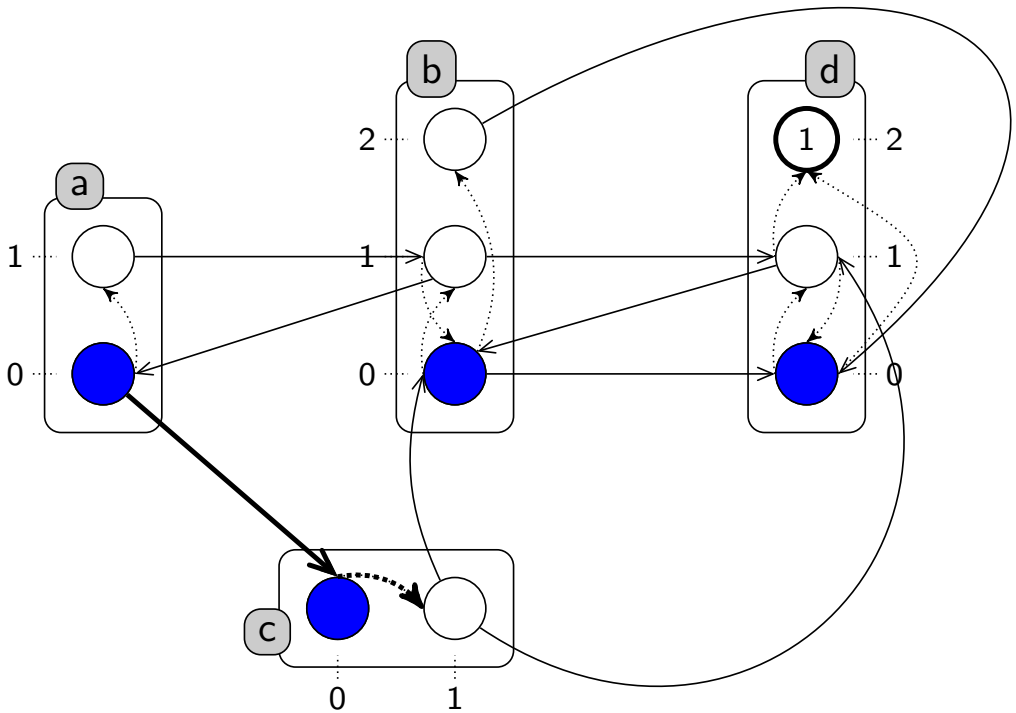
- Objectives

$$d_0 \Rsh^* d_1 :: b_0 \Rsh^* b_1 :: d_1 \Rsh^* d_2$$
$$d_0 \Rsh^* d_2$$

Static Analysis: Successive Reachability

[PMR12-MSCS]

Successive reachability of processes:



- Initial context
  $\langle a_1, \{b_0, b_1\}, c_0, z_0 \rangle$
- Objectives
  $d_0 \Rsh^* d_1 :: b_0 \Rsh^* b_1 :: d_1 \Rsh^* d_2$
  $d_0 \Rsh^* d_2$

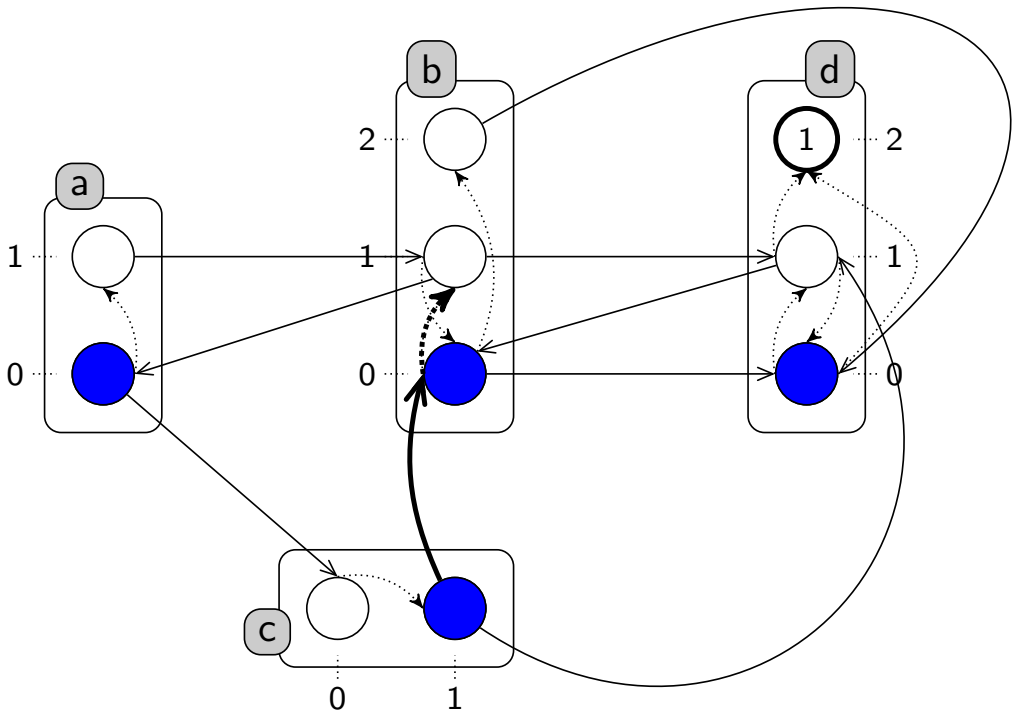
$\rightarrow$ Concretization of the objective = scenario

$\underline{a_0 \rightarrow c_0 \Rsh c_1 :: b_0} \rightarrow d_0 \Rsh d_1 :: c_1 \rightarrow b_0 \Rsh b_1 :: b_1 \rightarrow d_1 \Rsh d_2$

Static Analysis: Successive Reachability

[PMR12-MSCS]

Successive reachability of processes:



- Initial context

$$\langle a_1, \{b_0, b_1\}, c_0, z_0 \rangle$$

- Objectives

$$d_0 \Rsh^* d_1 :: b_0 \Rsh^* b_1 :: d_1 \Rsh^* d_2$$
$$d_0 \Rsh^* d_2$$

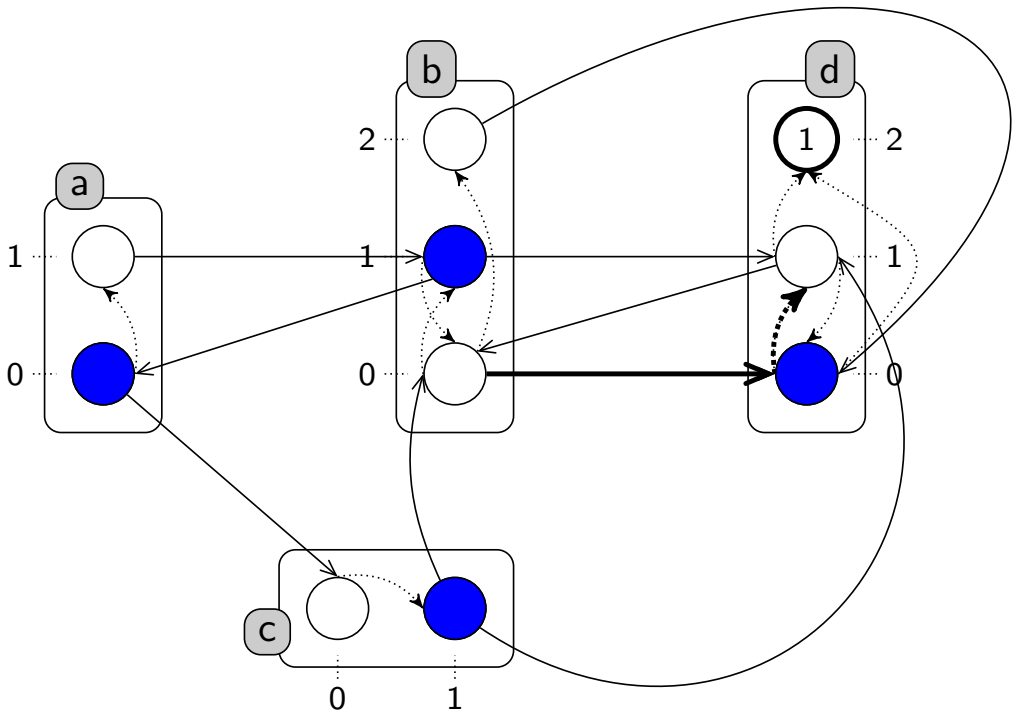
$\rightarrow$ Concretization of the objective = scenario

$$a_0 \rightarrow c_0 \Rsh c_1 :: \underline{b_0 \rightarrow d_0 \Rsh d_1} :: c_1 \rightarrow b_0 \Rsh b_1 :: b_1 \rightarrow d_1 \Rsh d_2$$

Static Analysis: Successive Reachability

[PMR12-MSCS]

Successive reachability of processes:



- Initial context

$$\langle a_1, \{b_0, b_1\}, c_0, z_0 \rangle$$

- Objectives

$$d_0 \Rsh^* d_1 :: b_0 \Rsh^* b_1 :: d_1 \Rsh^* d_2$$
$$d_0 \Rsh^* d_2$$

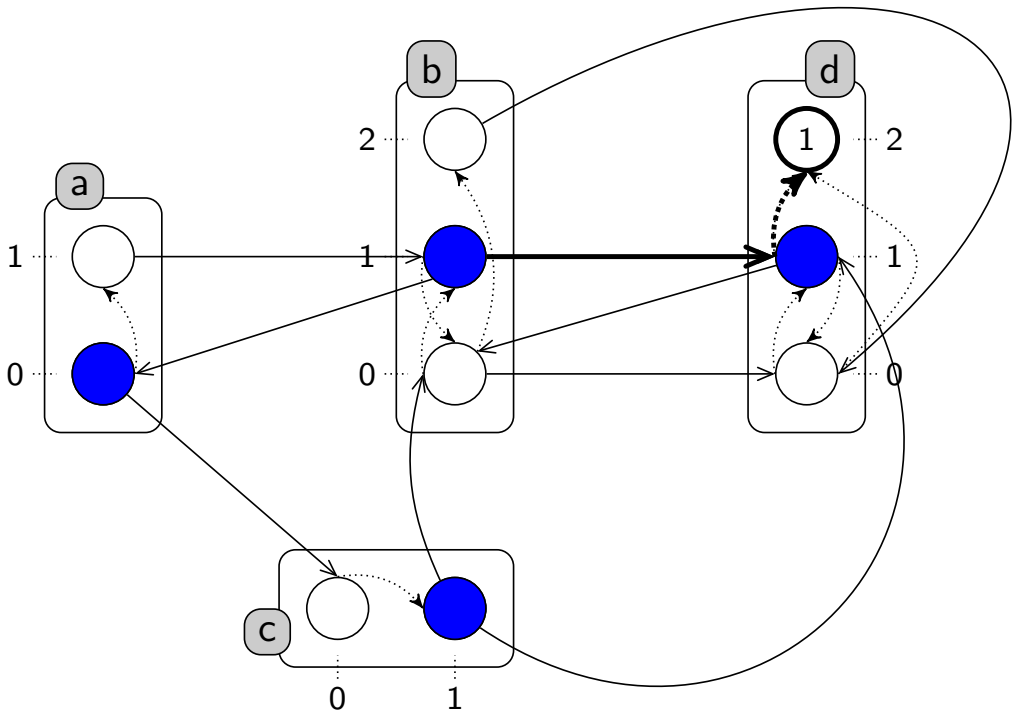
$\rightarrow$ Concretization of the objective = scenario

$$a_0 \rightarrow c_0 \Rsh c_1 :: b_0 \rightarrow d_0 \Rsh d_1 :: \underline{c_1 \rightarrow b_0 \Rsh b_1} :: b_1 \rightarrow d_1 \Rsh d_2$$

Static Analysis: Successive Reachability

[PMR12-MSCS]

Successive reachability of processes:



- Initial context

$$\langle a_1, \{b_0, b_1\}, c_0, z_0 \rangle$$

- Objectives

$$d_0 \Rsh^* d_1 :: b_0 \Rsh^* b_1 :: d_1 \Rsh^* d_2$$
$$d_0 \Rsh^* d_2$$

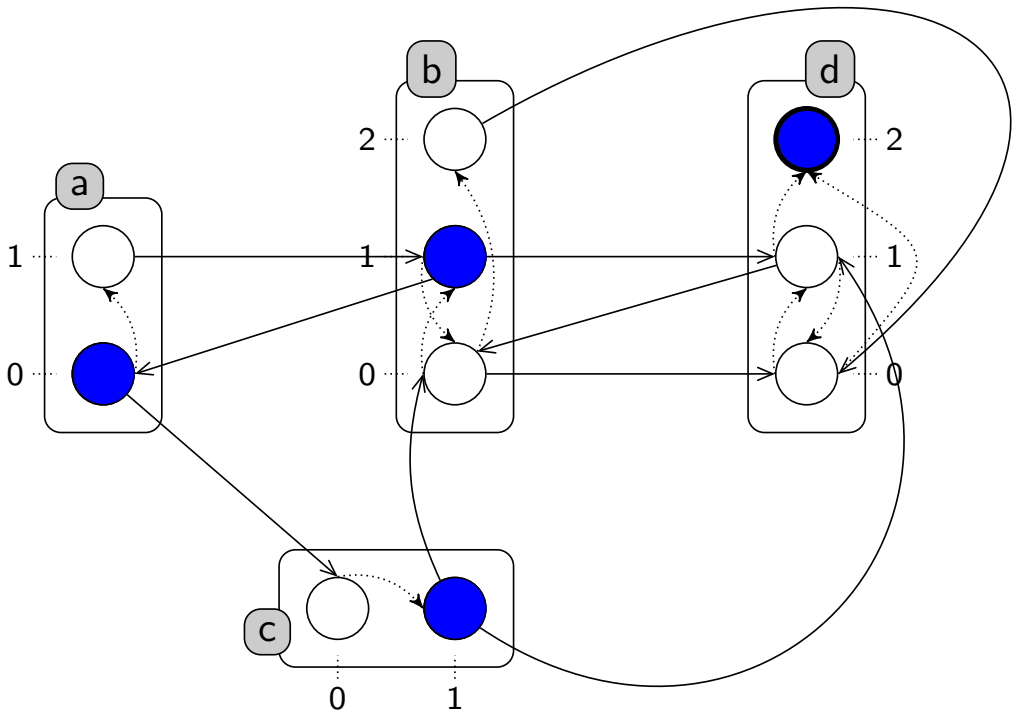
$\rightarrow$ Concretization of the objective = scenario

$$a_0 \rightarrow c_0 \Rsh c_1 :: b_0 \rightarrow d_0 \Rsh d_1 :: c_1 \rightarrow b_0 \Rsh b_1 :: \underline{b_1 \rightarrow d_1 \Rsh d_2}$$

Static Analysis: Successive Reachability

[PMR12-MSCS]

Successive reachability of processes:



- Initial context

$$\langle a_1, \{b_0, b_1\}, c_0, z_0 \rangle$$

- Objectives

$$d_0 \Rsh^* d_1 :: b_0 \Rsh^* b_1 :: d_1 \Rsh^* d_2$$
$$d_0 \Rsh^* d_2$$

$\rightarrow$ Concretization of the objective = scenario

$$a_0 \rightarrow c_0 \Rsh c_1 :: b_0 \rightarrow d_0 \Rsh d_1 :: c_1 \rightarrow b_0 \Rsh b_1 :: b_1 \rightarrow d_1 \Rsh d_2$$

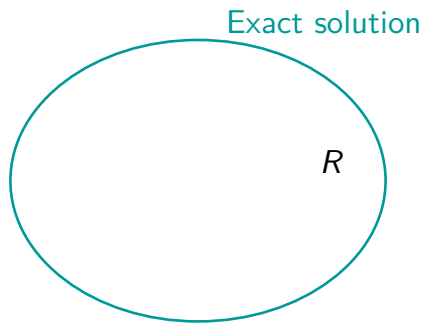
Over- and Under-approximations

[PMR12-MSCS]

Static analysis by abstractions:

$\rightarrow$ Directly checking an objective sequence R is hard

$\rightarrow$ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



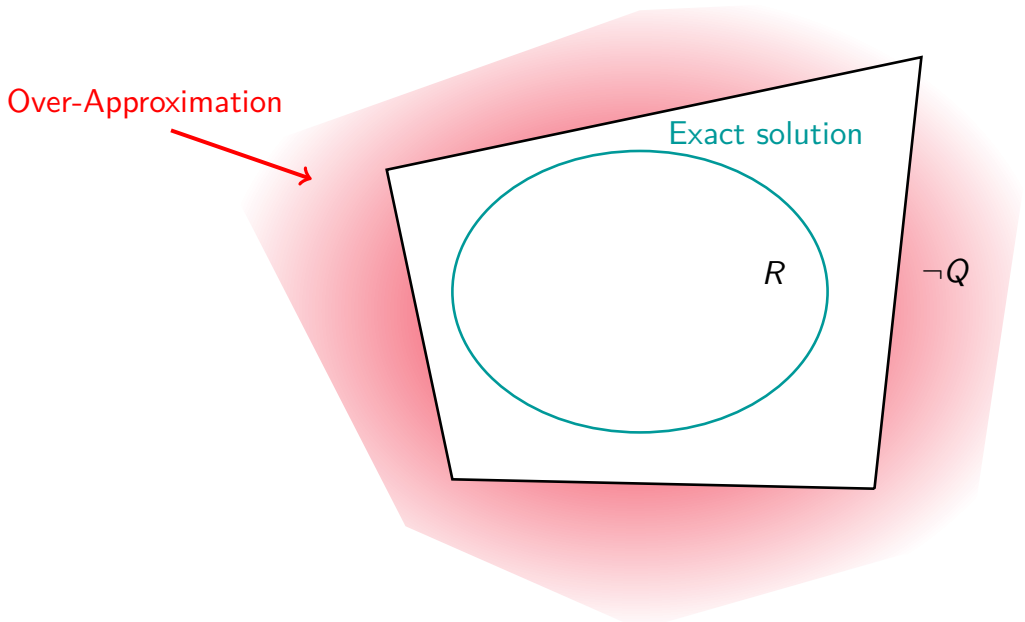
Over- and Under-approximations

[PMR12-MSCS]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard

→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



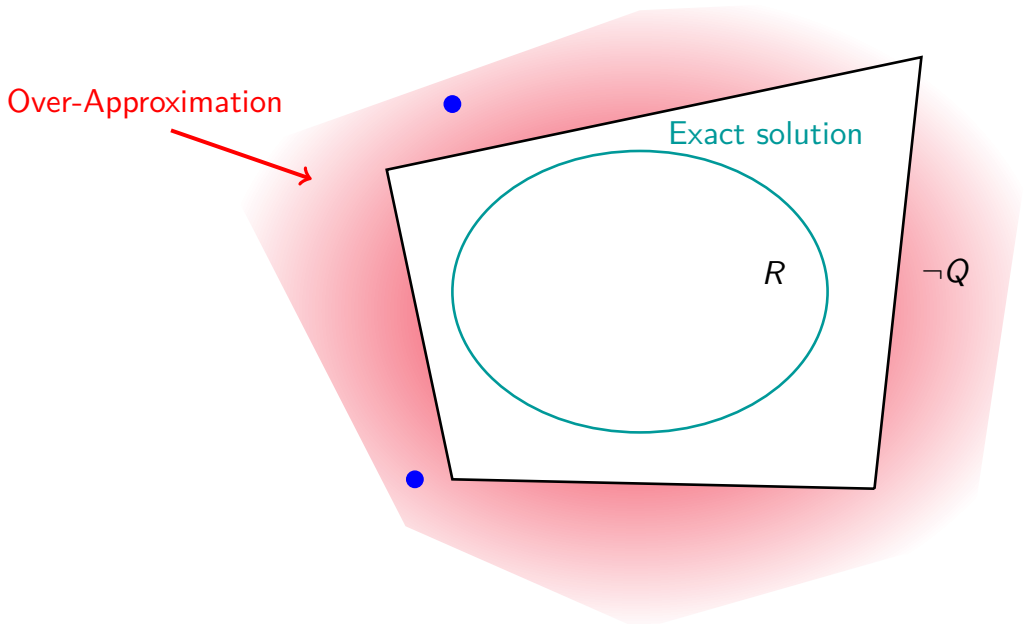
Over- and Under-approximations

[PMR12-MSCS]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard

→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



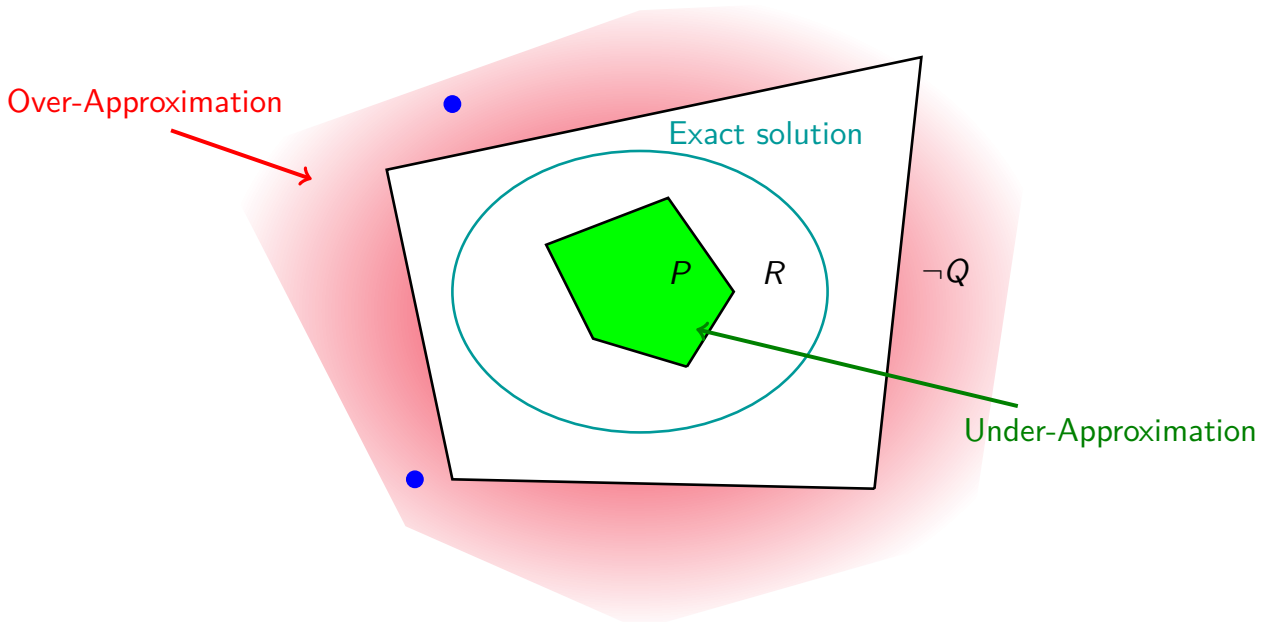
Over- and Under-approximations

[PMR12-MSCS]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard

→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



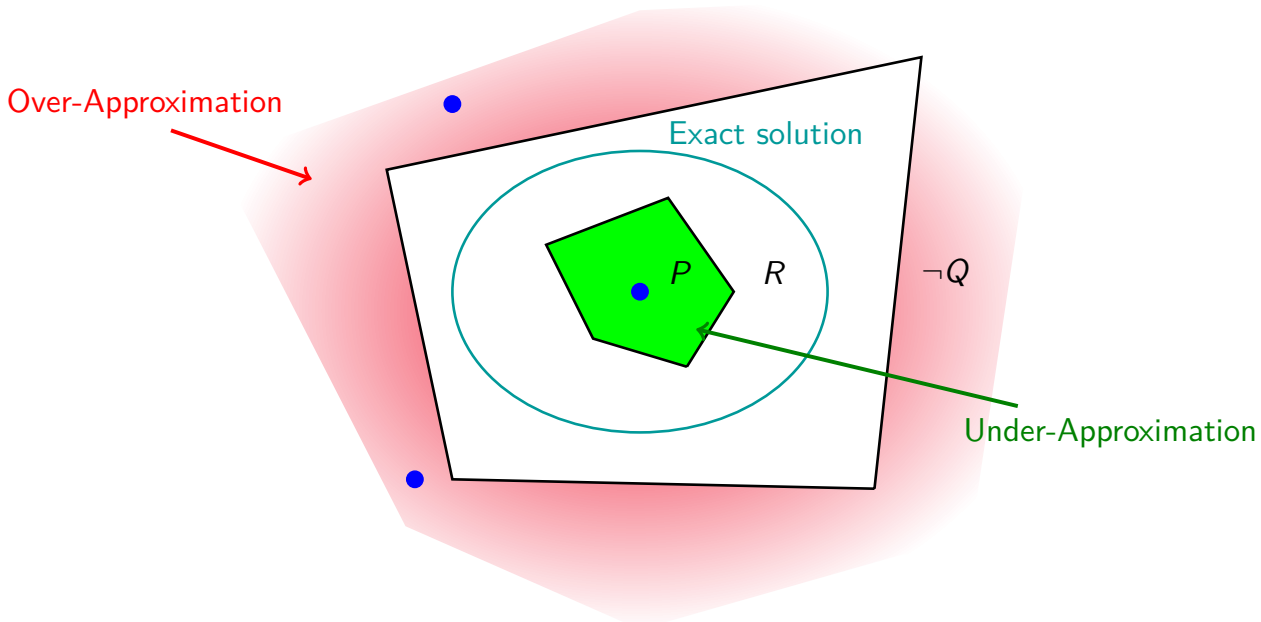
Over- and Under-approximations

[PMR12-MSCS]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard

→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



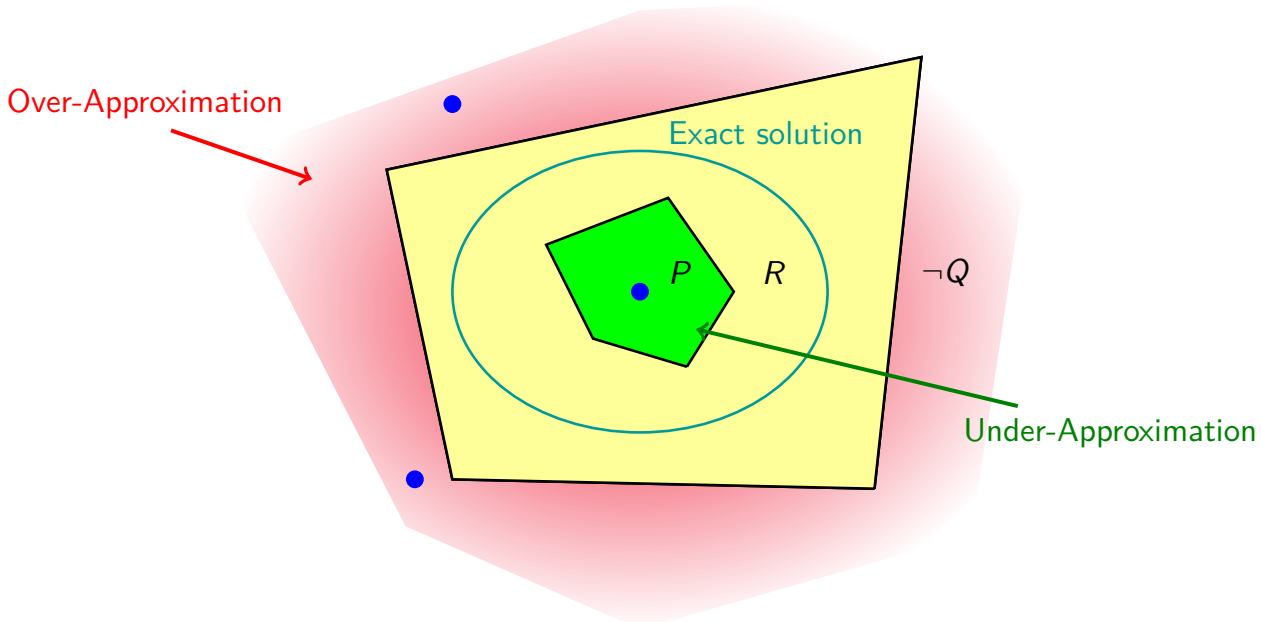
Over- and Under-approximations

[PMR12-MSCS]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard

→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



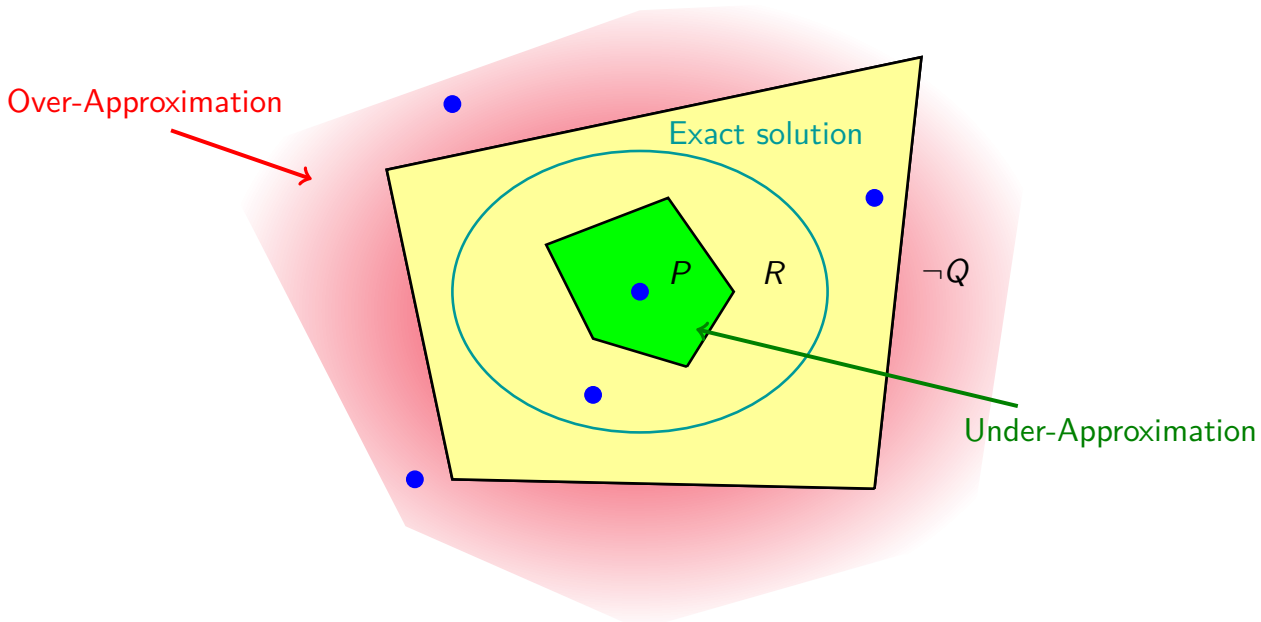
Over- and Under-approximations

[PMR12-MSCS]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard

→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



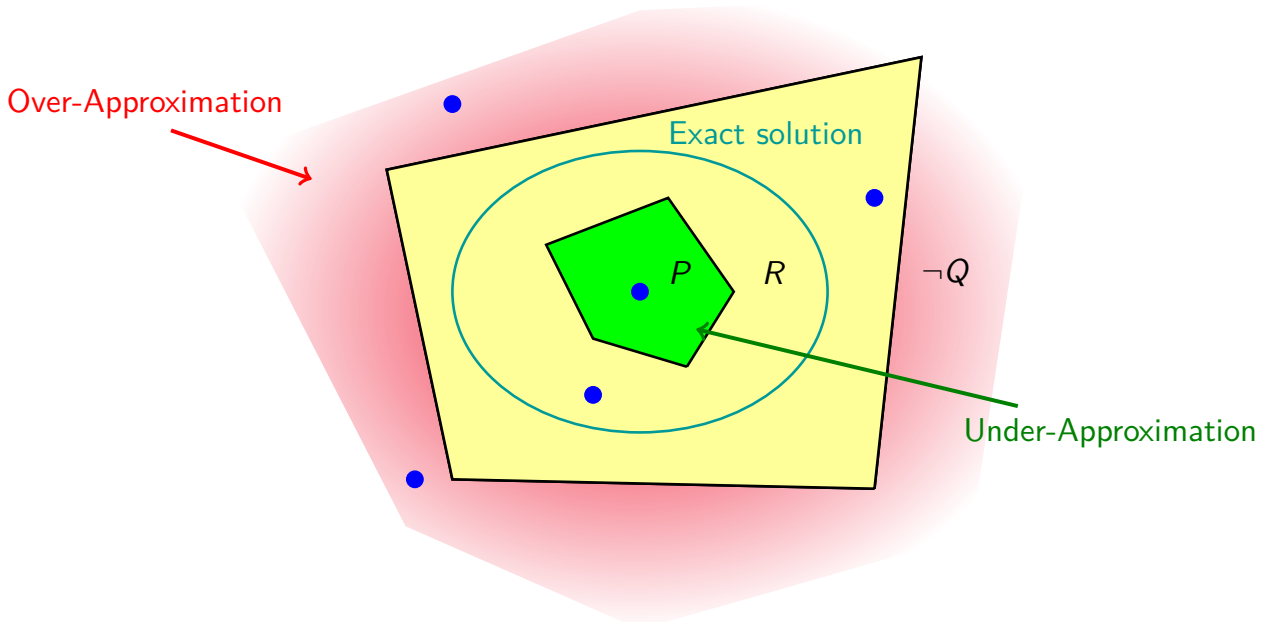
Over- and Under-approximations

[PMR12-MSCS]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard

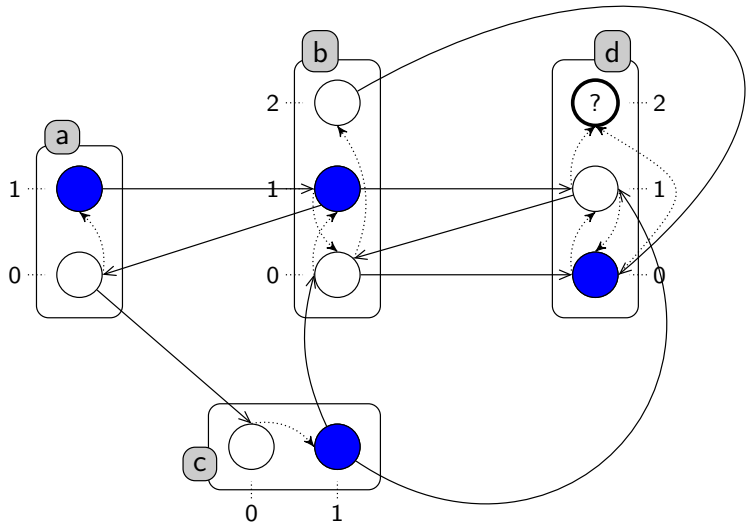
→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



Linear w.r.t. the number of sorts and
exponential w.r.t. the number of processes in each sort

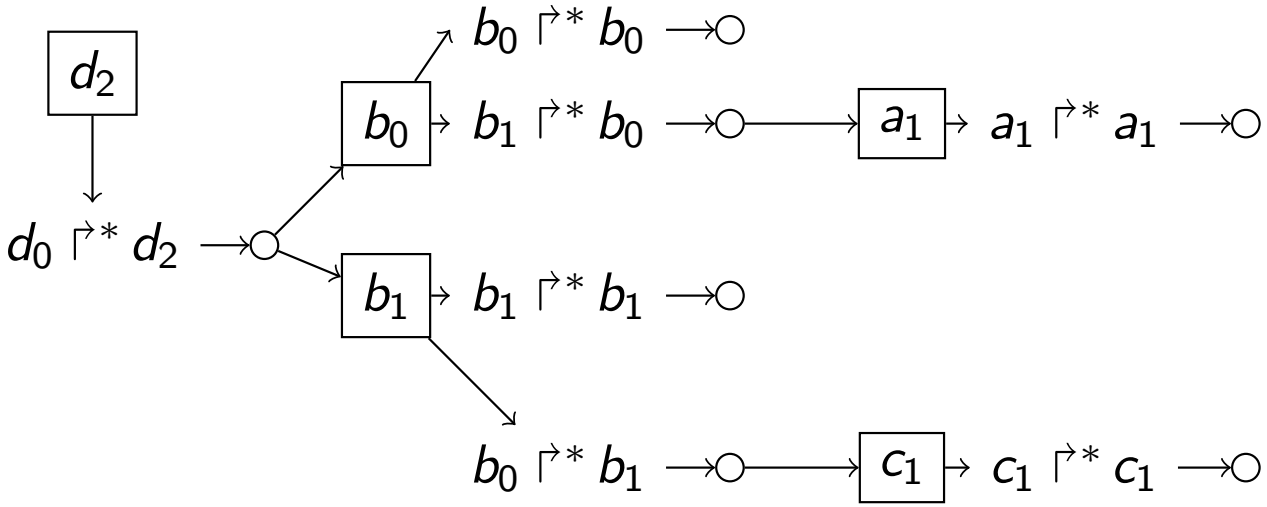
→ Efficient for big models with few levels of expression

Under-approximation

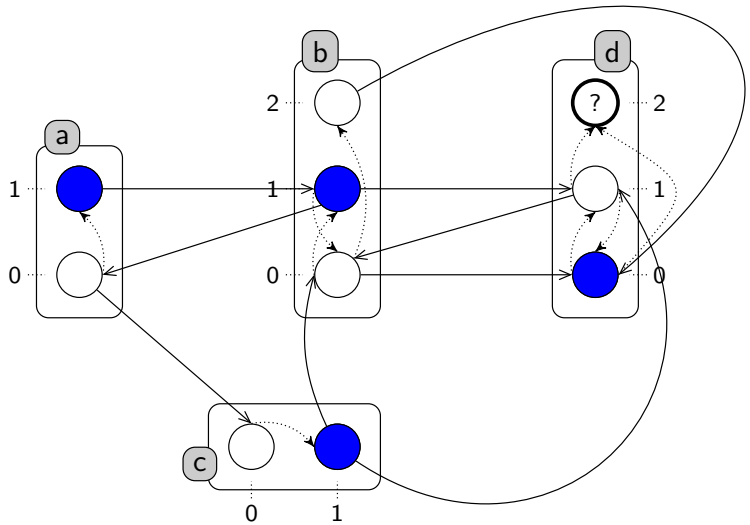


$\rightarrow$ New abstract structure

Sufficient condition:



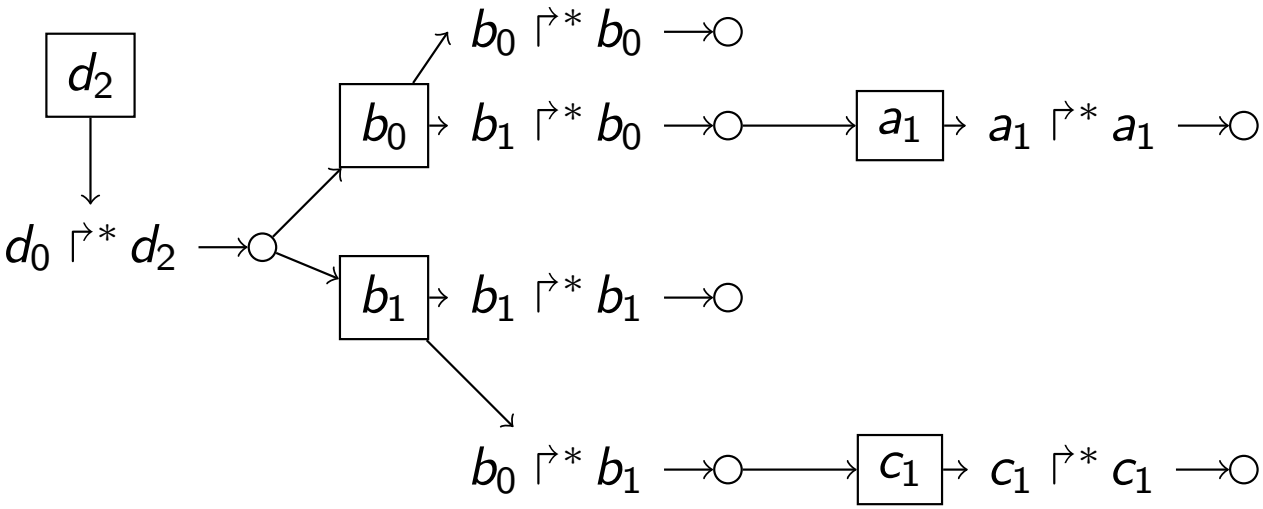
Under-approximation



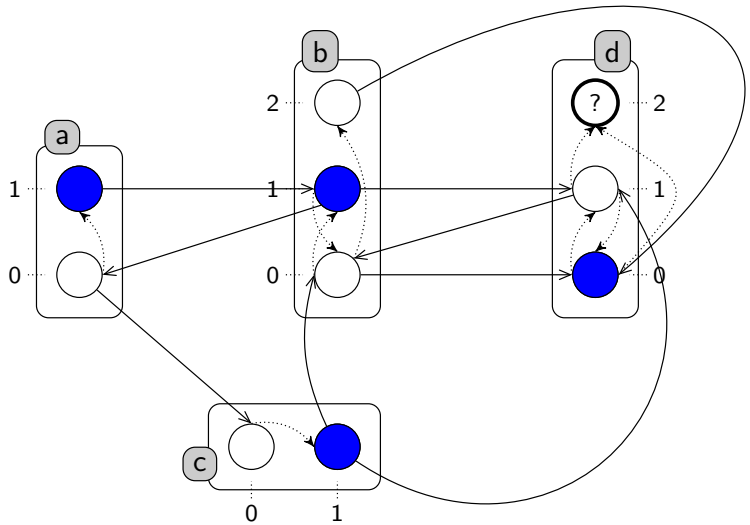
$\rightarrow$ New abstract structure

Sufficient condition:

- no cycle
- each objective has a solution



Under-approximation

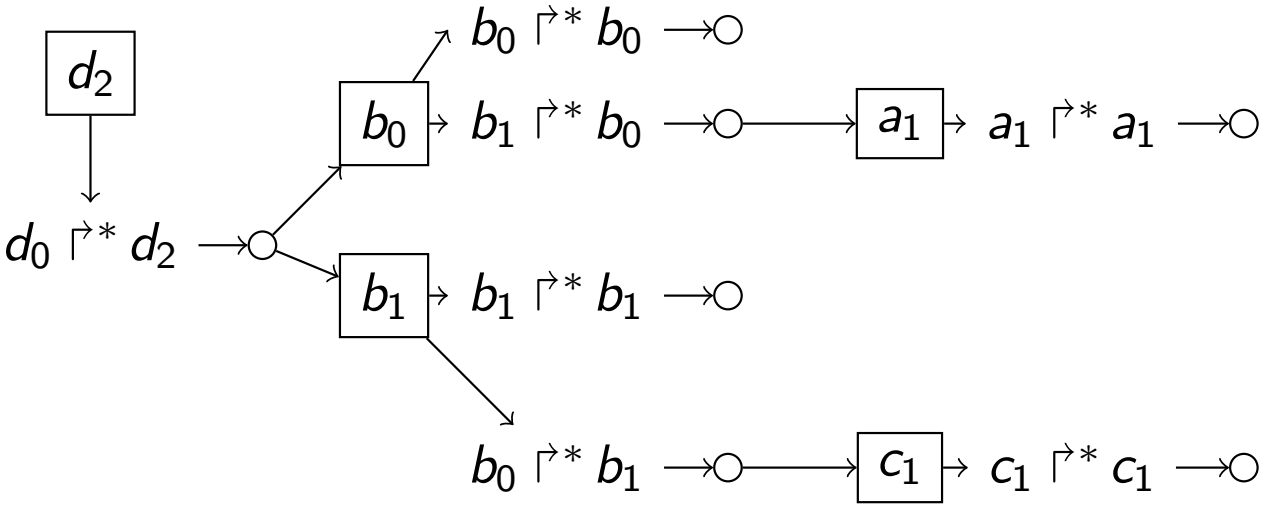


$\rightarrow$ New abstract structure

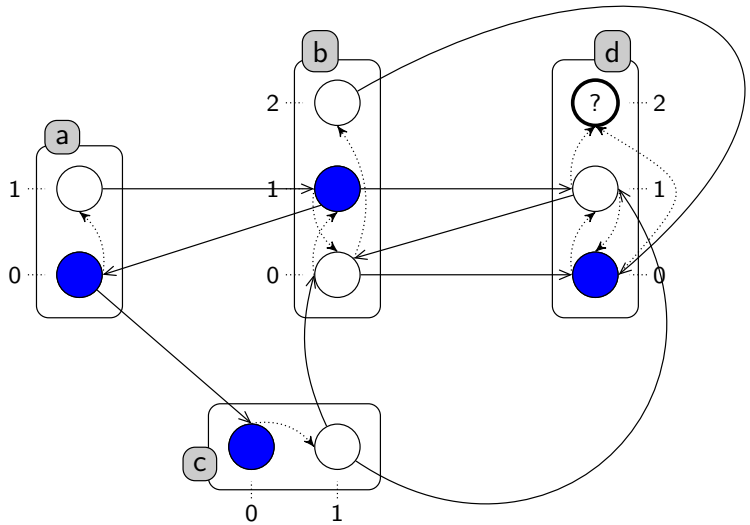
Sufficient condition:

- no cycle
- each objective has a solution

R is **true**



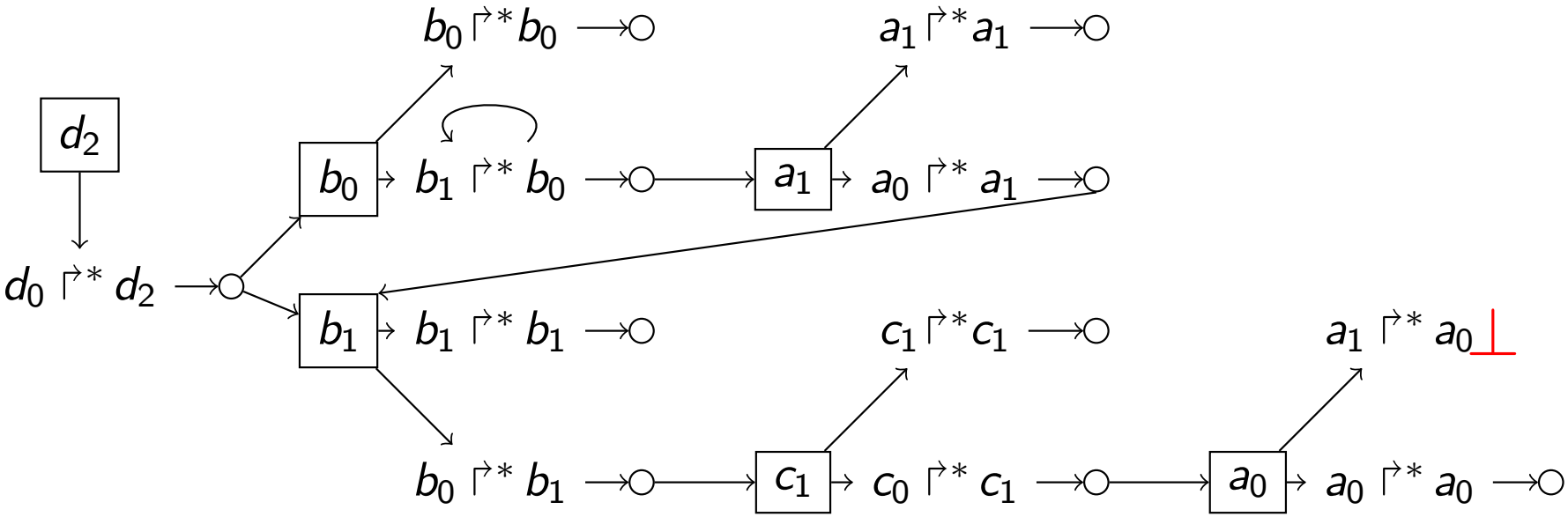
Under-approximation



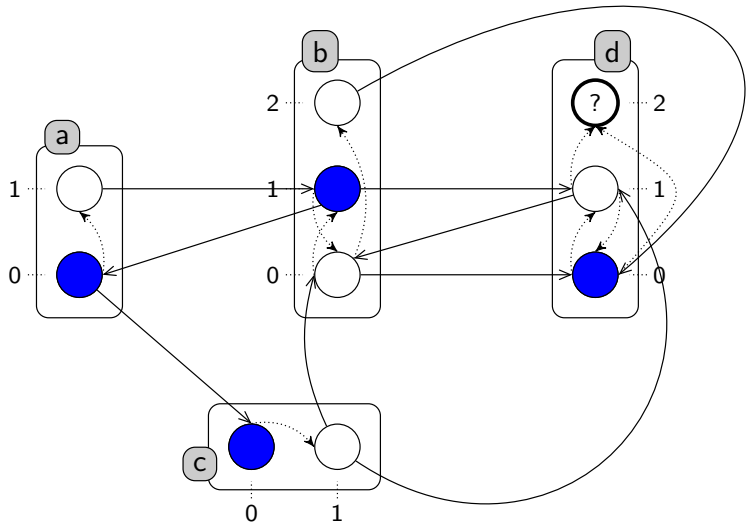
$\rightarrow$ New abstract structure

Sufficient condition:

- ~~no cycle~~
- ~~each objective has a solution~~



Under-approximation

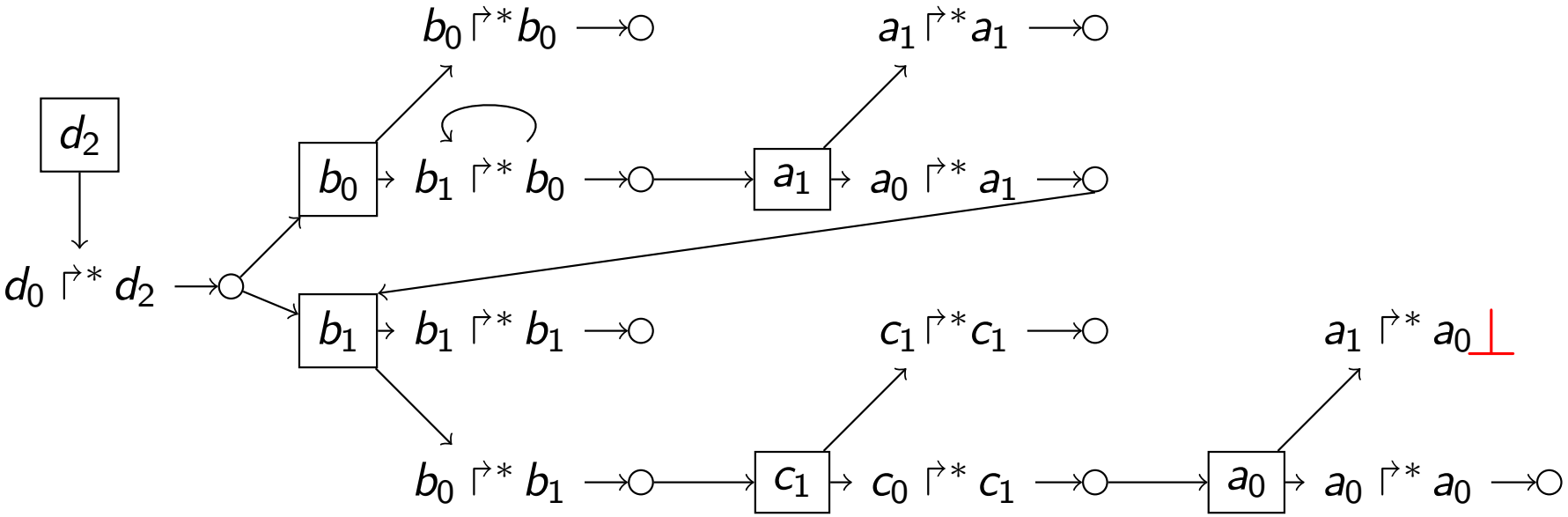


→ New abstract structure

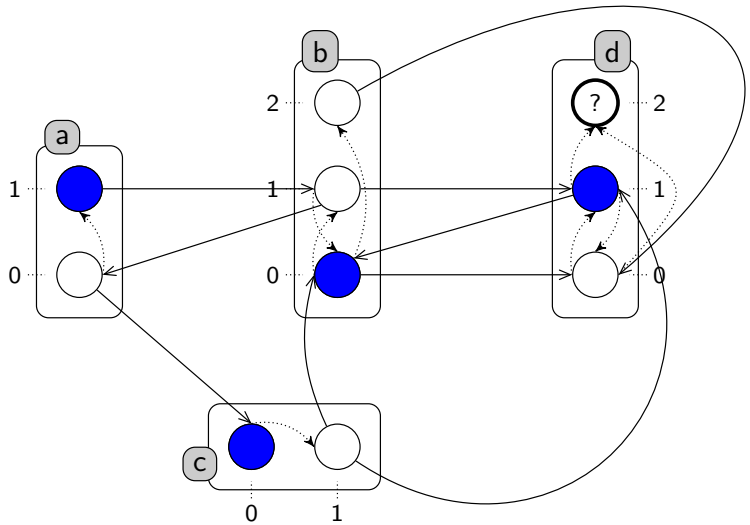
Sufficient condition:

- ~~no cycle~~
- ~~each objective has a solution~~

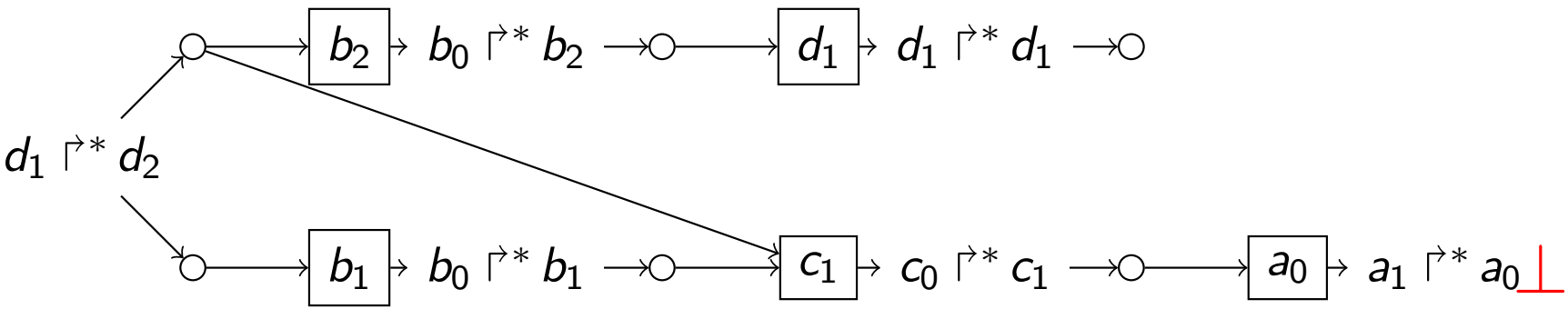
Inconclusive



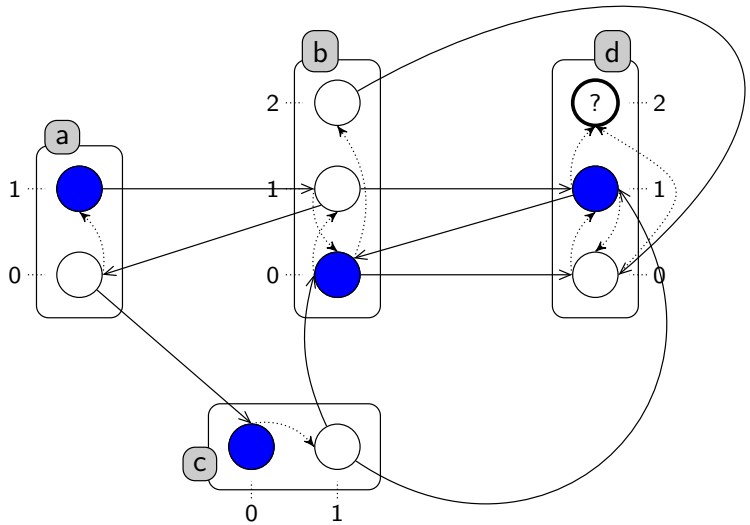
Over-approximation



Necessary condition:



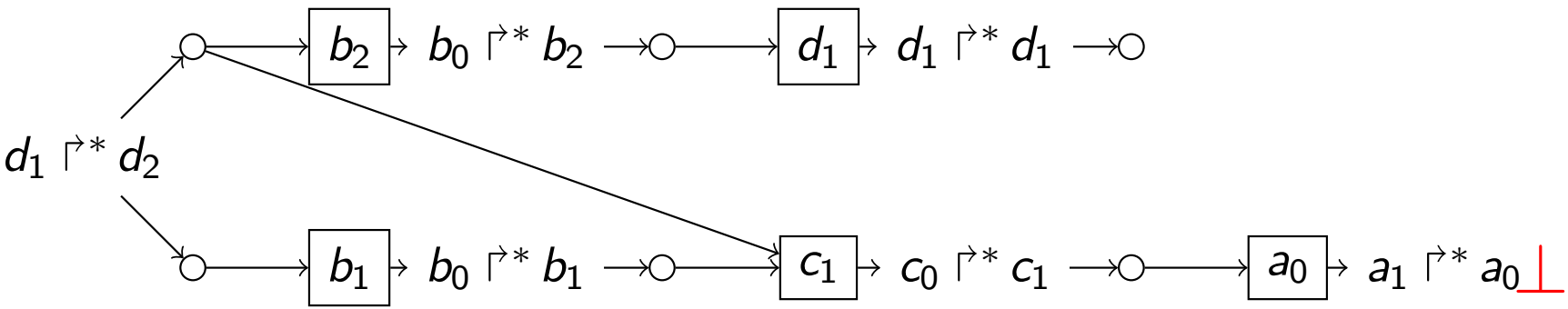
Over-approximation



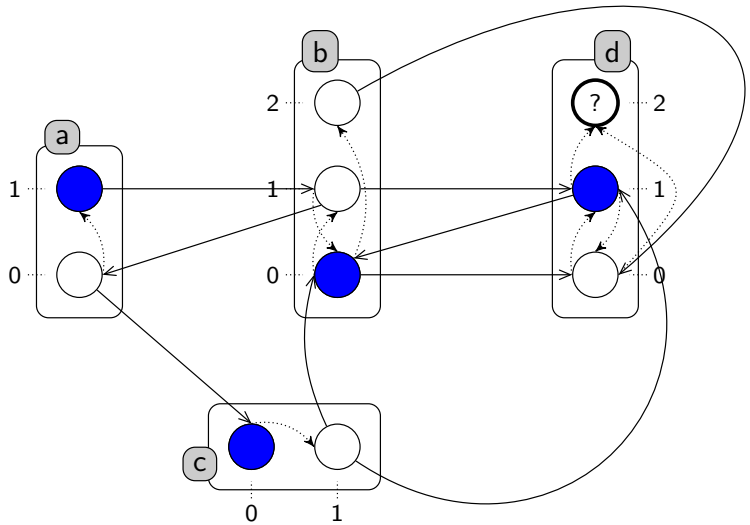
Necessary condition:

There exists a traversal with no cycle

- objective $\rightarrow$ follow one solution
- solution $\rightarrow$ follow all processes
- process $\rightarrow$ follow all objectives



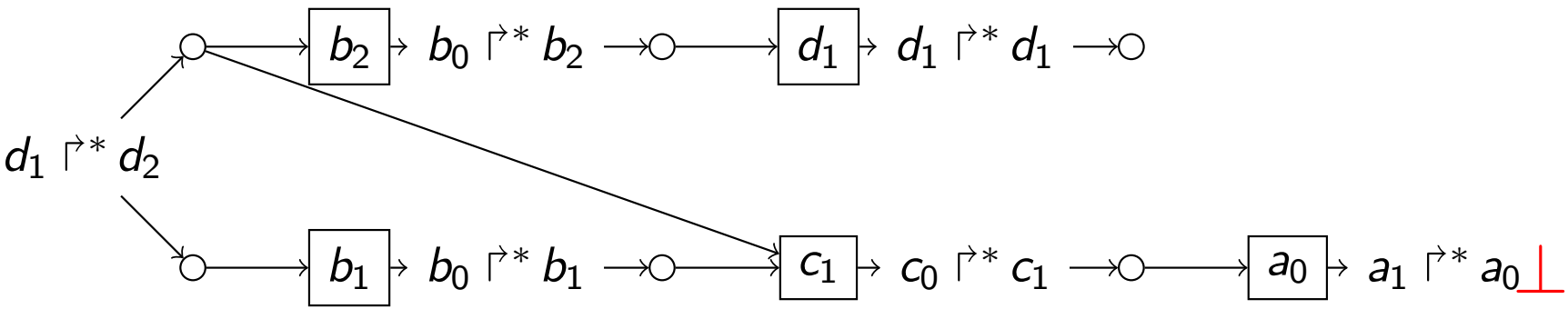
Over-approximation



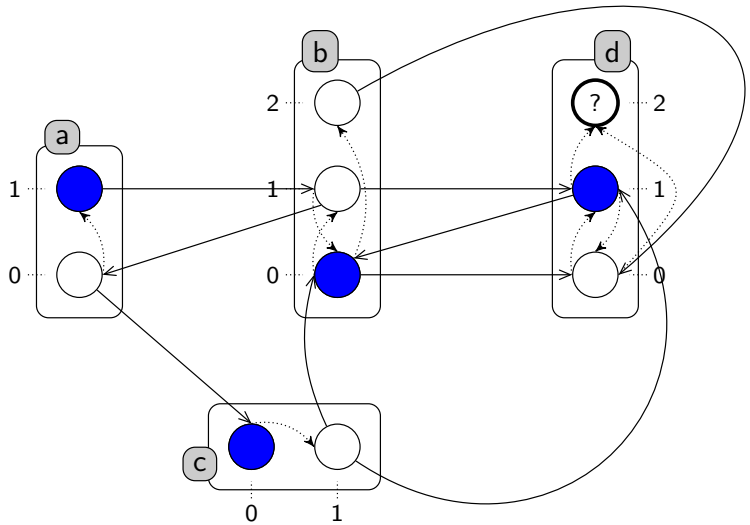
Necessary condition:

~~There exists a traversal~~ with no cycle

- ~~objective → follow one solution~~
- solution → follow all processes
- process → follow all objectives



Over-approximation

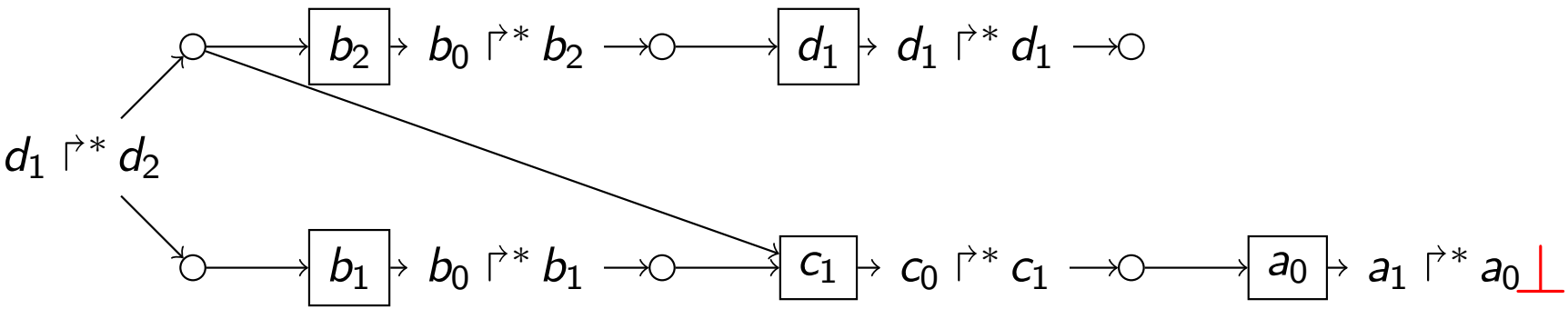


Necessary condition:

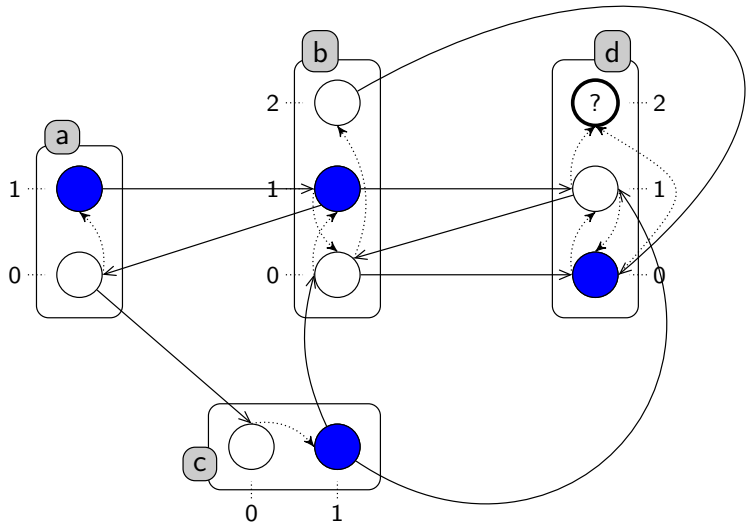
~~There exists a traversal~~ with no cycle

- ~~objective → follow one solution~~
- solution → follow all processes
- process → follow all objectives

R is **false**



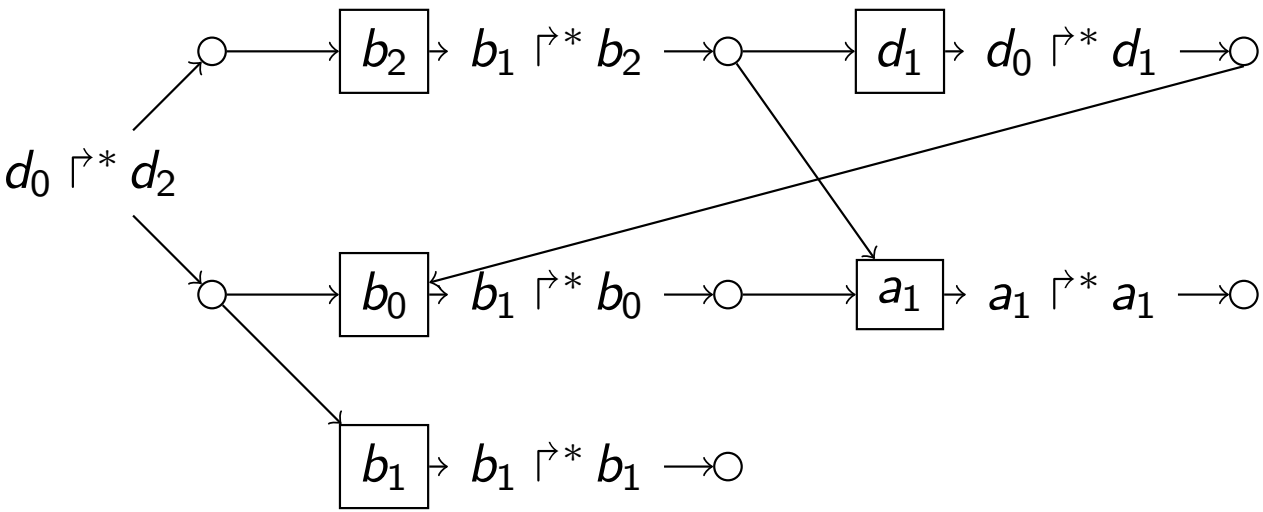
Over-approximation



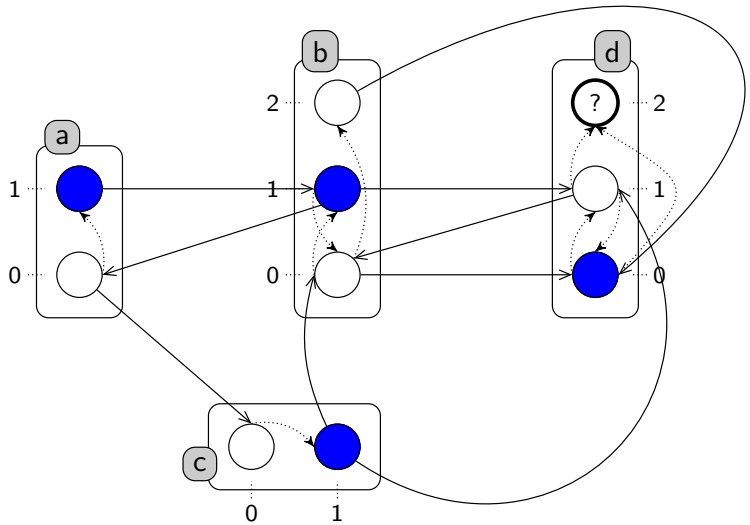
Necessary condition:

There exists a traversal with no cycle

- objective $\rightarrow$ follow one solution
- solution $\rightarrow$ follow all processes
- process $\rightarrow$ follow all objectives



Over-approximation

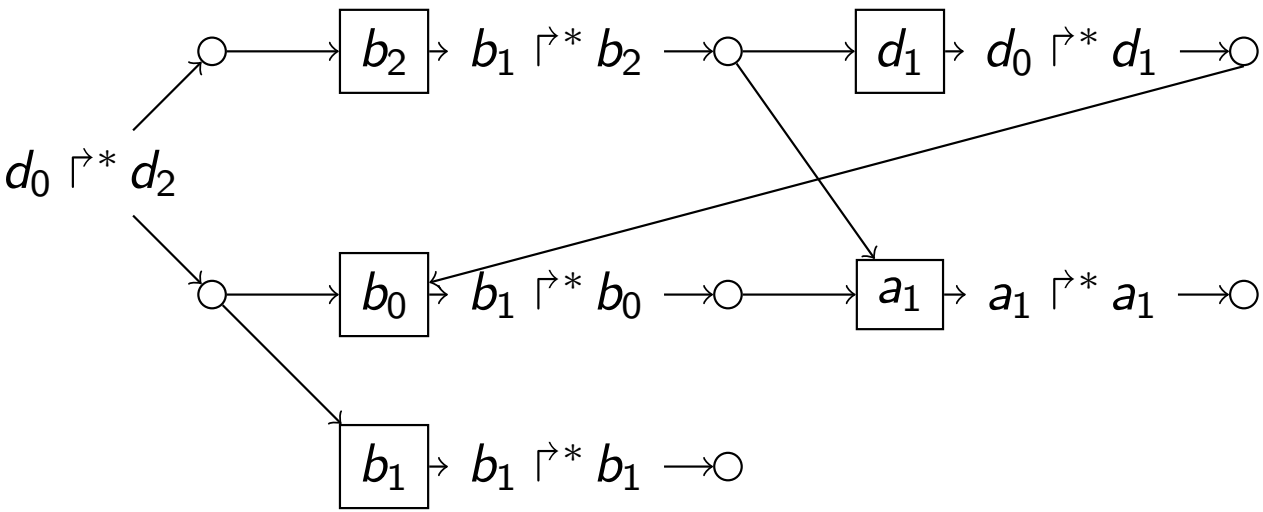


Necessary condition:

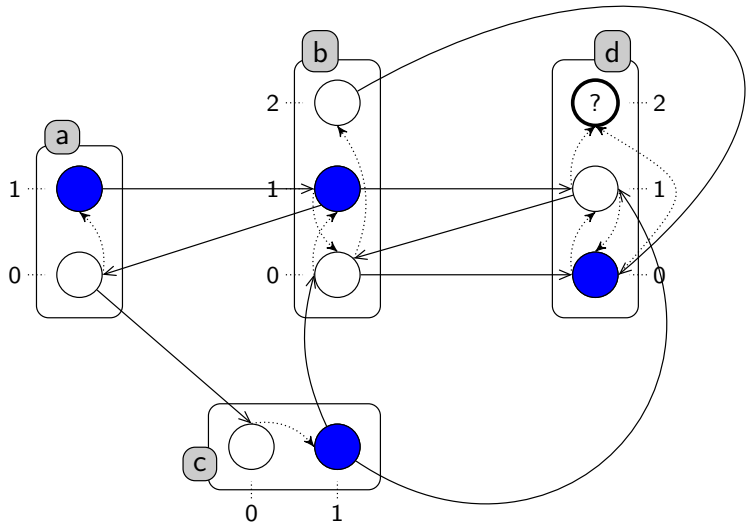
There exists a traversal with no cycle

- objective $\rightarrow$ follow one solution
- solution $\rightarrow$ follow all processes
- process $\rightarrow$ follow all objectives

Inconclusive



Over-approximation

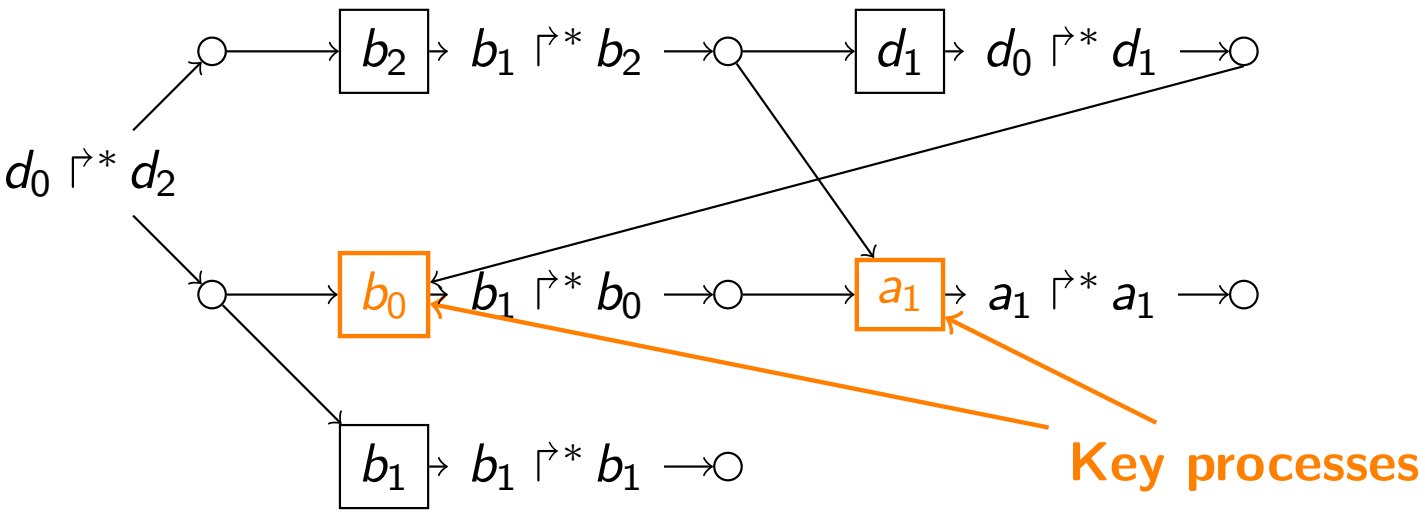


Necessary condition:

There exists a traversal with no cycle

- objective → follow one solution
- solution → follow all processes
- process → follow all objectives

Inconclusive



The Pint Tool

[processhitting.wordpress.com]

Textual language to describe a Process Hitting

Tools implemented:

- fixed points research
- stochastic simulation
- reachability checker
- translations from and to various other models

→ Free API available for future developments

The Pint Tool

[processhitting.wordpress.com]

Textual language to describe a Process Hitting

Tools implemented:

- fixed points research
- stochastic simulation
- reachability checker
- translations from and to various other models

→ Free API available for future developments

Results and performance (reachability analysis):

Model	sorts	procs	actions	states	Biocham[1]	libddd[2]	PINT
egfr20	35	196	670	2^{64}	[3s-KO]	[1s-150s]	**0.007s**
tcrsig40	54	156	301	2^{73}	[1s-KO]	[0.6s-KO]	**0.004s**
tcrsig94	133	448	1124	2^{194}	KO	KO	**0.030s**
egfr104	193	748	2356	2^{320}	KO	KO	**0.050s**

[1] [Inria Paris-Rocquencourt/Contraintes]
[2] [LIP6/Move]

Application to Biological Systems

The Process Hitting framework:

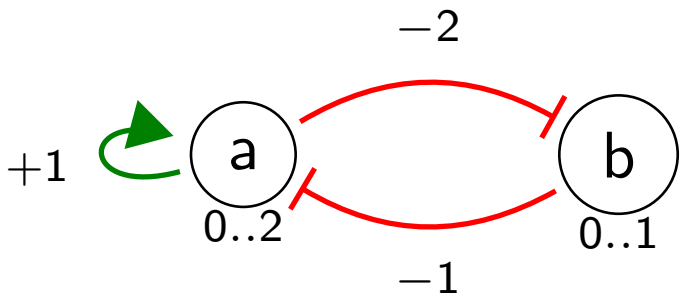
- new formalism with simple elements
- dynamic modeling
- efficient reachability checking
- general framework that could be applied to...

Biological systems:

- gene/protein couples
- Thomas' Modeling: Interaction Graphs
- hard to study $\rightarrow$ use Process Hitting

Interaction Graphs

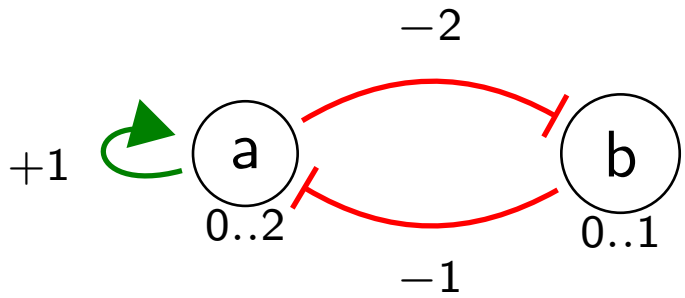
- Nodes = Genes
- Directed edges = Interactions



+ Parametrization

Interaction Graphs

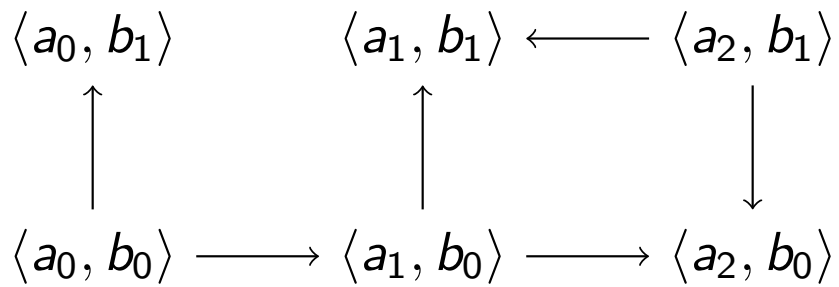
- Nodes = Genes
- Directed edges = Interactions



+ Parametrization

State Graphs

- One level at a time
- Asynchronous



Exponential number of states w.r.t. the number of genes

Tools for Interaction Graphs Study

Interaction Graphs [RCB08]:

- No positive circuit $\Rightarrow$ only 1 attractor
- No negative circuit $\Rightarrow$ no cyclic attractor
- Positive circuits $\Rightarrow$ criterion for max. number of attractors
- Temporal logics $\Rightarrow$ check properties (needs State Graph)
  $\rightarrow$ SM-BIONET

Boolean Networks [PR10-CRAS]:

- Results on topological fixed points

Tools for Interaction Graphs Study

Interaction Graphs [RCB08]:

- No positive circuit $\Rightarrow$ only 1 attractor
- No negative circuit $\Rightarrow$ no cyclic attractor
- Positive circuits $\Rightarrow$ criterion for max. number of attractors
- Temporal logics $\Rightarrow$ check properties (**needs State Graph**)
  $\rightarrow$ SM-BIONET

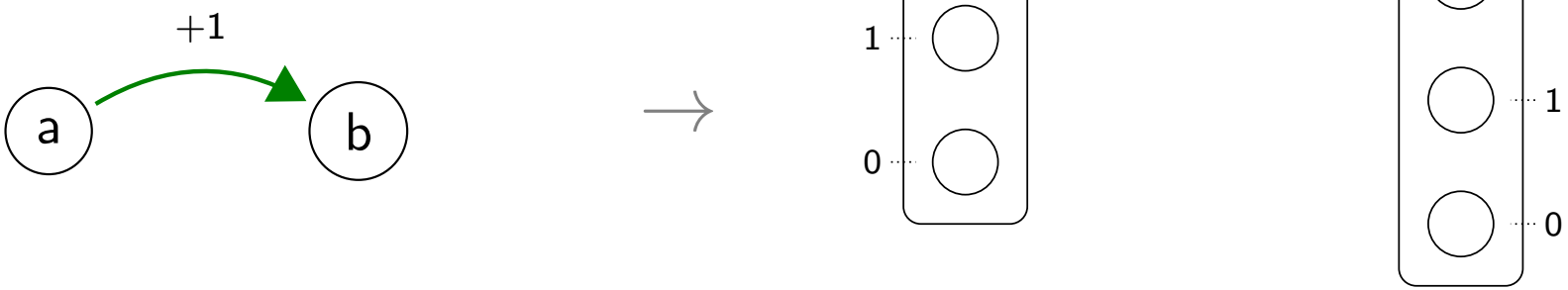
Boolean Networks [PR10-CRAS]:

- Results on topological fixed points

Problem: combinatorial explosion when computing the State Graph
$\rightarrow$ Need for static analysis $\rightarrow$ Use the Process Hitting

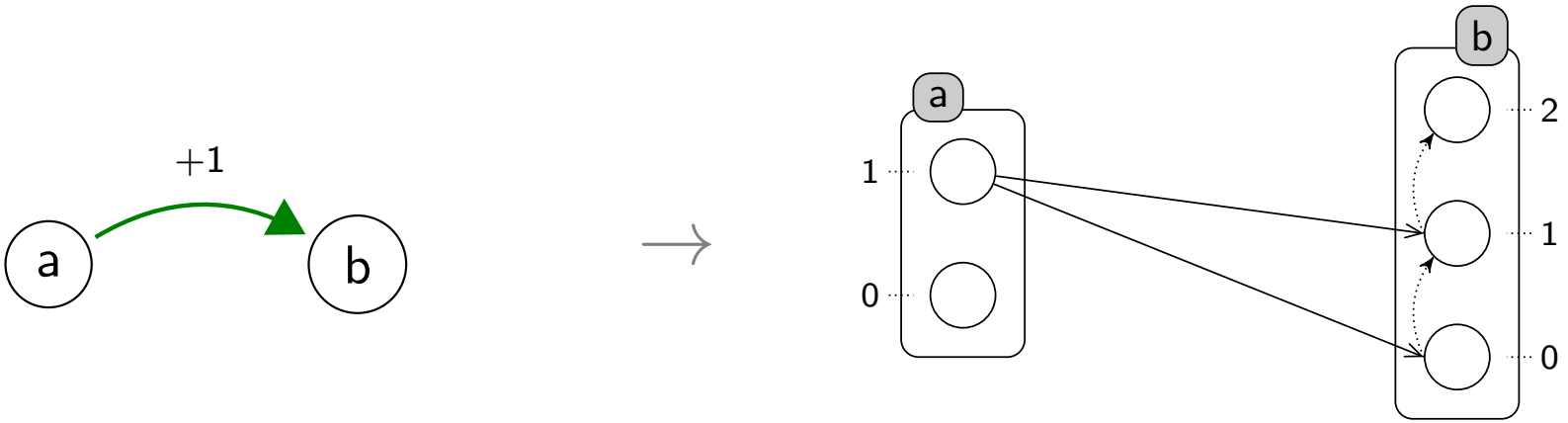
Translation of the Generalized Dynamics

Positive interaction:



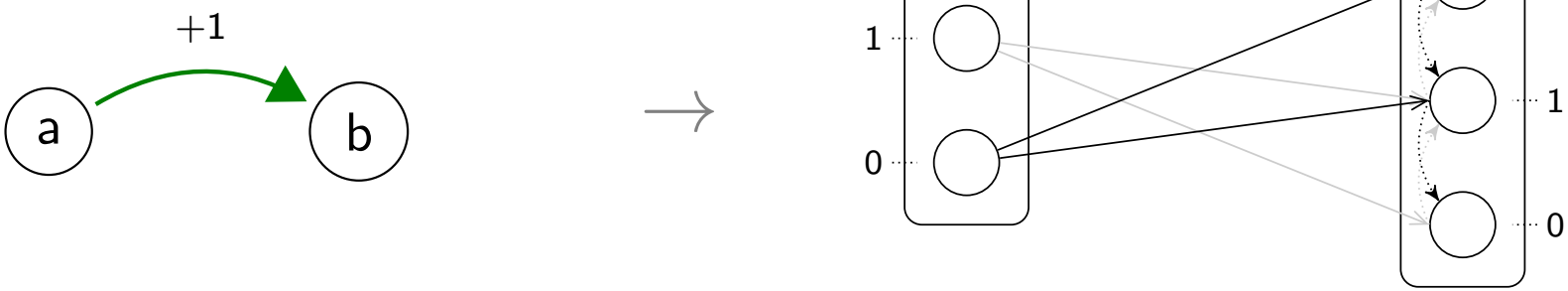
Translation of the Generalized Dynamics

Positive interaction:



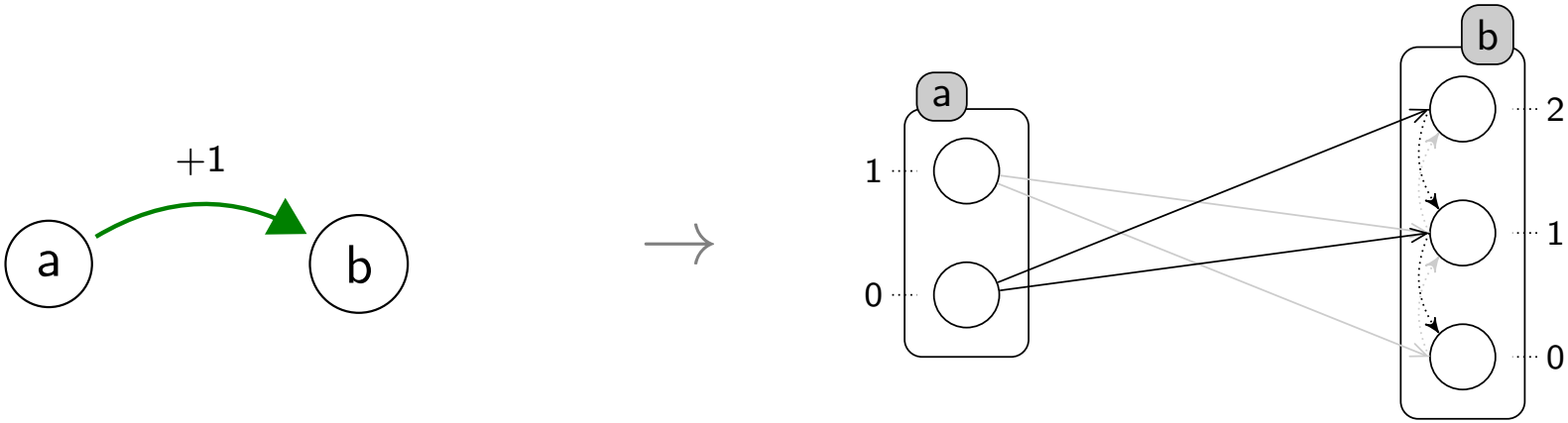
Translation of the Generalized Dynamics

Positive interaction:

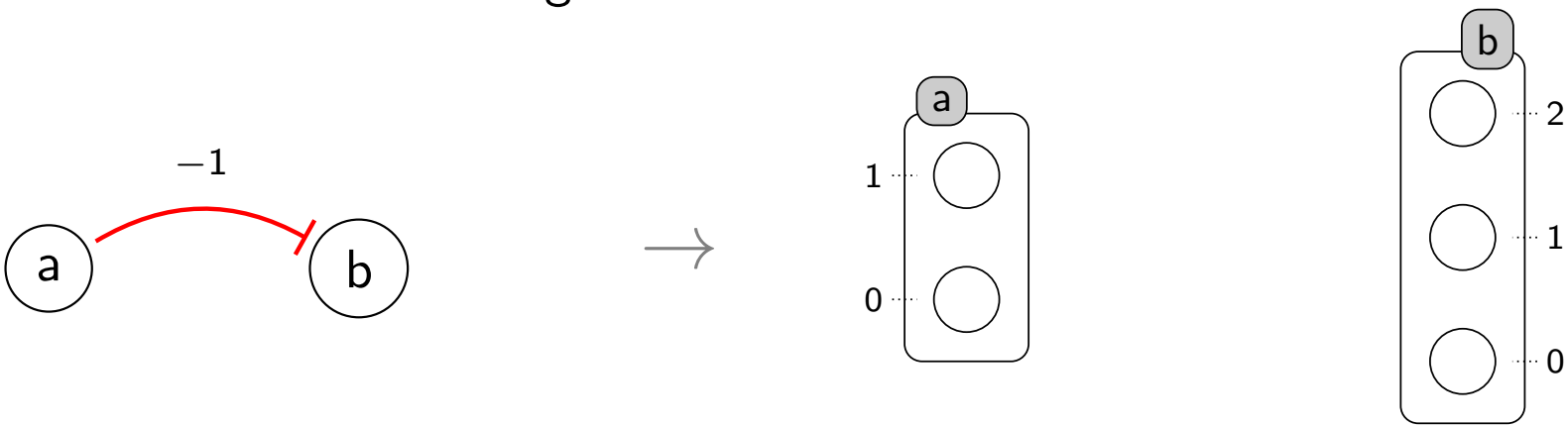


Translation of the Generalized Dynamics

Positive interaction:

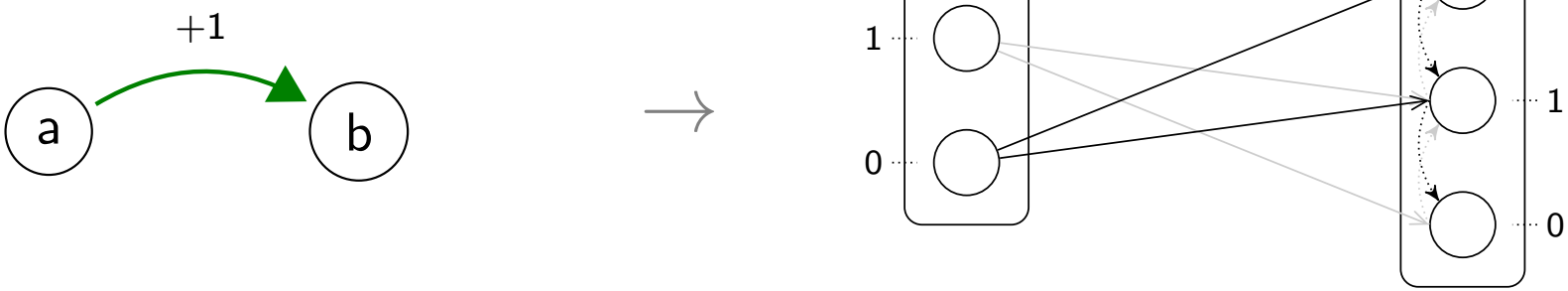


Negative interaction:

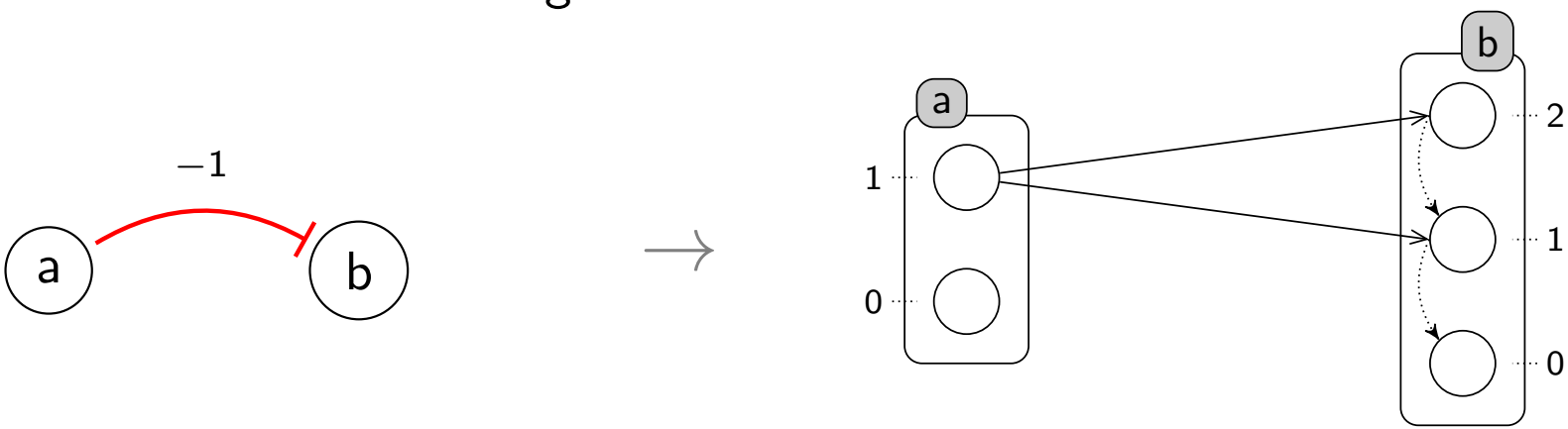


Translation of the Generalized Dynamics

Positive interaction:

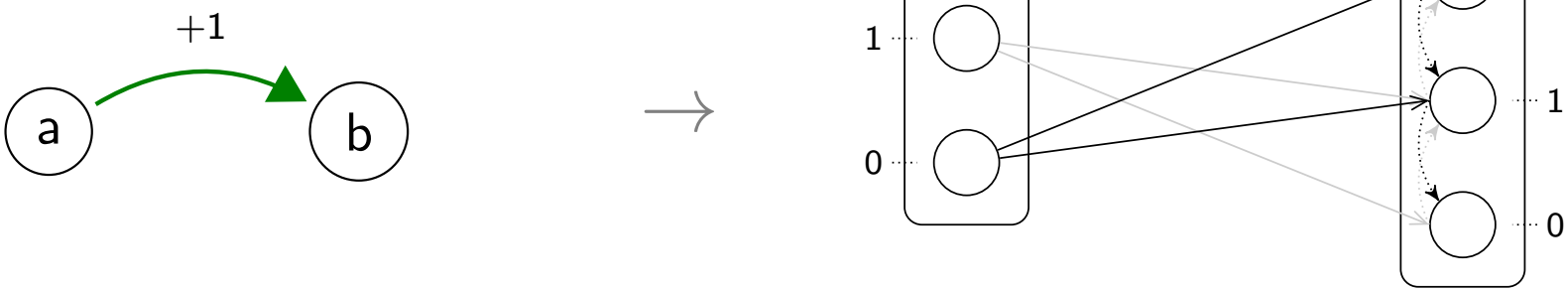


Negative interaction:

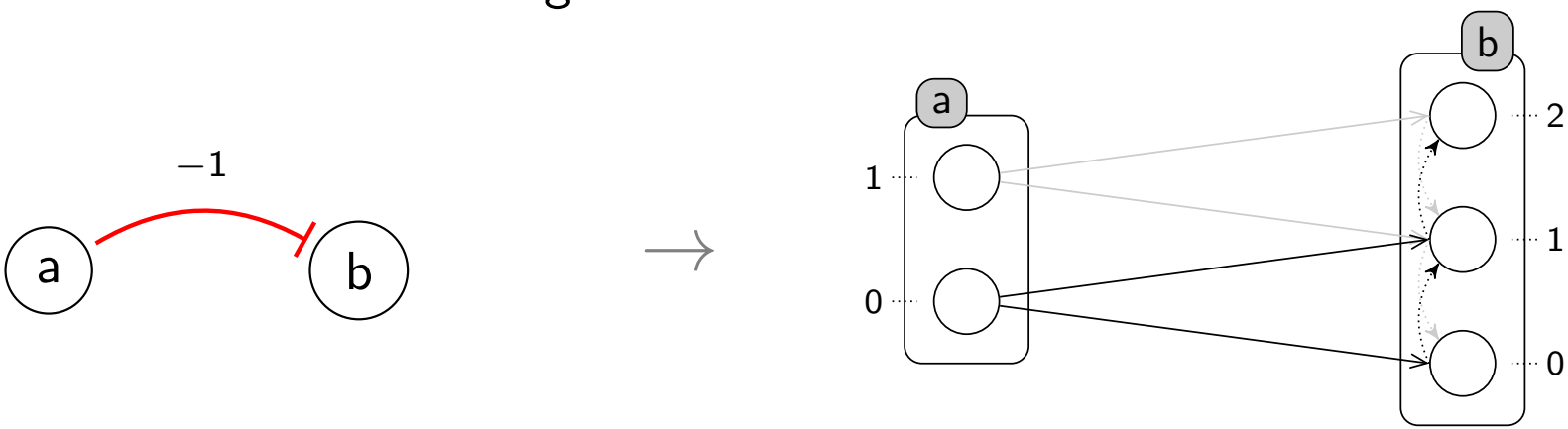


Translation of the Generalized Dynamics

Positive interaction:

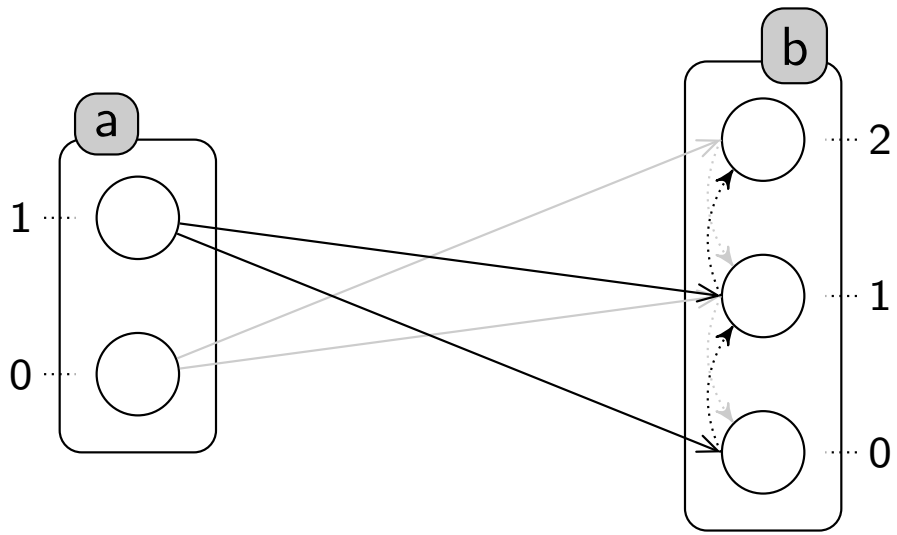


Negative interaction:



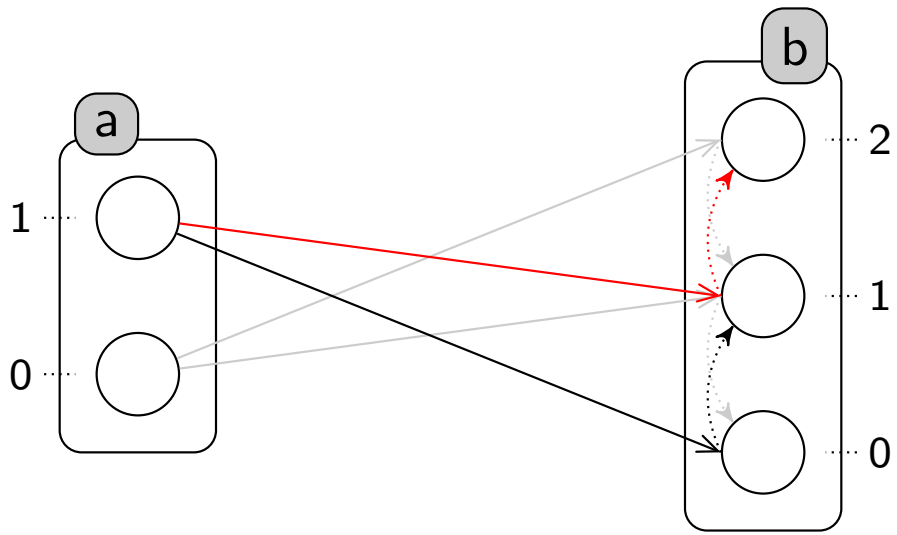
Refining with Actions Removal

Prevent behaviors by deleting **unrealistic actions**



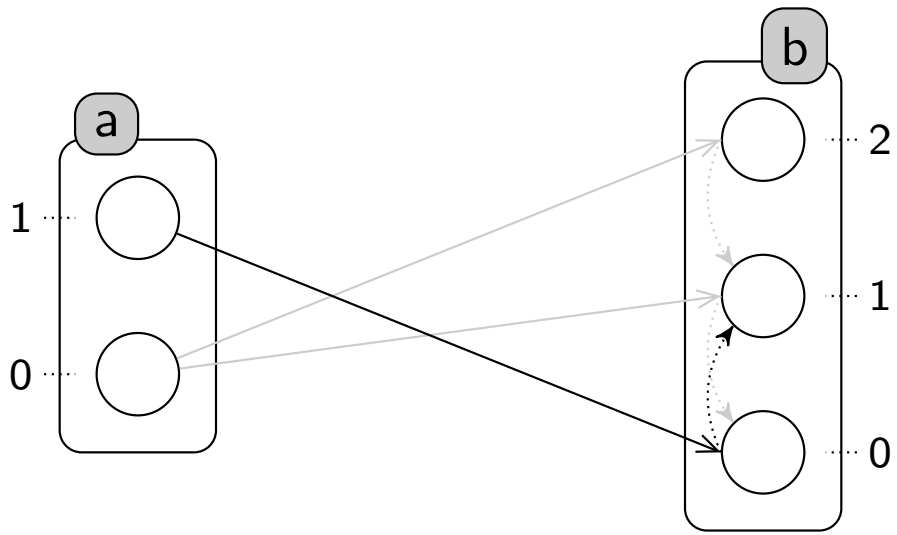
Refining with Actions Removal

Prevent behaviors by deleting **unrealistic actions**



Refining with Actions Removal

Prevent behaviors by deleting **unrealistic actions**

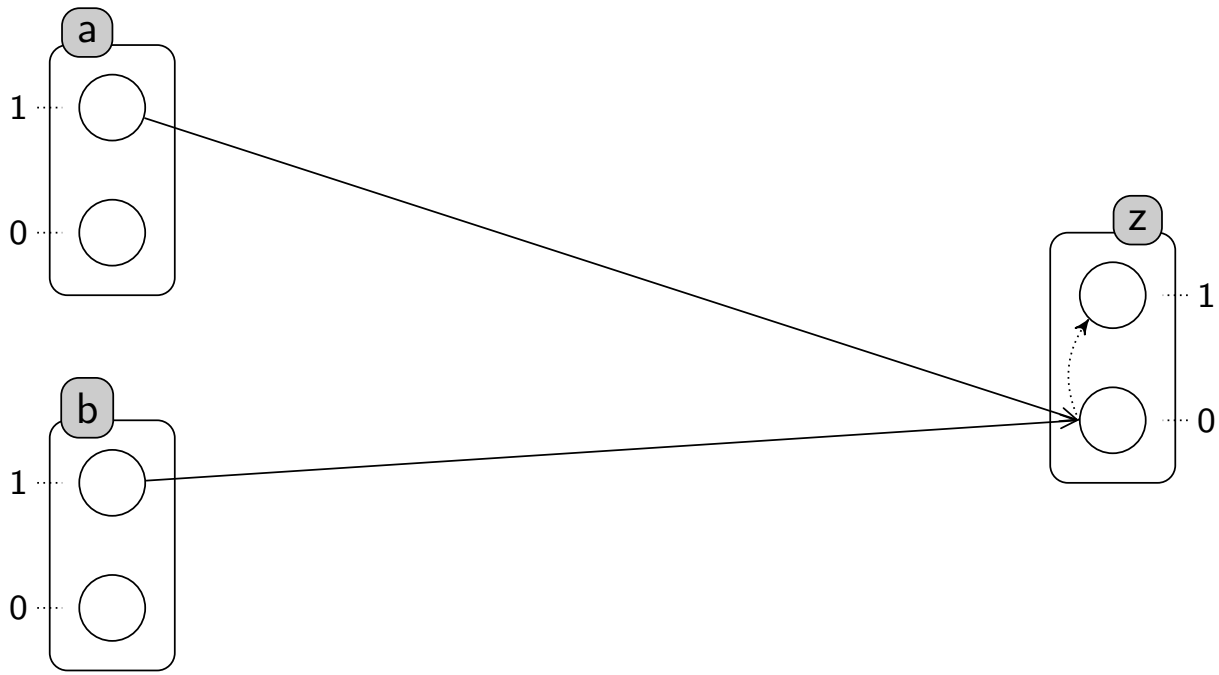


Refining with Cooperation

Allow **cooperation** between two genes

- How to express $(a_1 \wedge b_1) \rightarrow z_0 \Uparrow z_1$?

$\rightarrow$ Add a "**cooperative sort**" reflecting the state of a and b

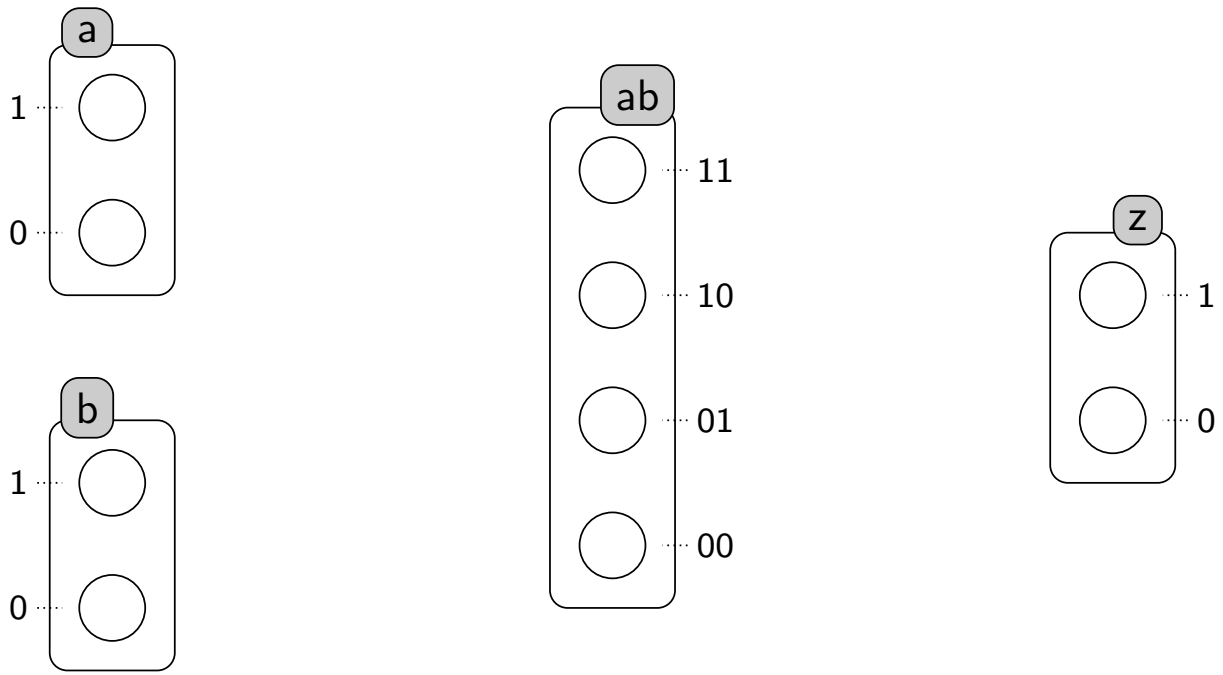


Refining with Cooperation

Allow **cooperation** between two genes

- How to express $(a_1 \wedge b_1) \rightarrow z_0 \Uparrow z_1$?

$\rightarrow$ Add a "**cooperative sort**" reflecting the state of a and b

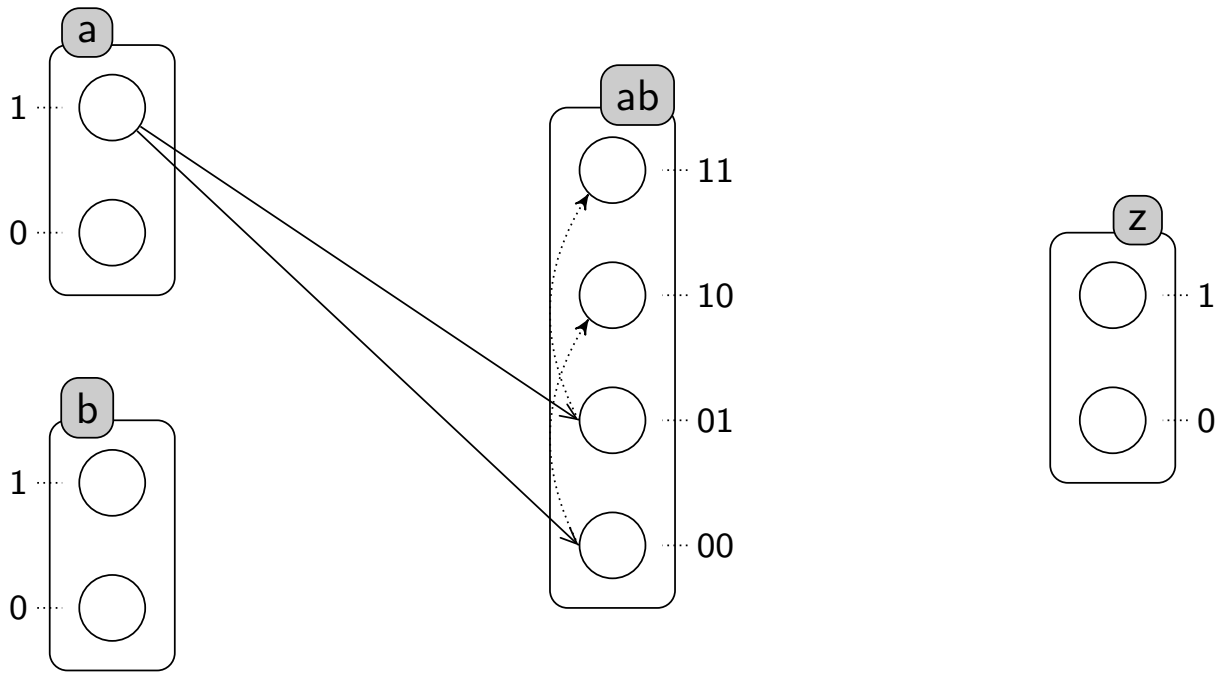


Refining with Cooperation

Allow **cooperation** between two genes

- How to express $(a_1 \wedge b_1) \rightarrow z_0 \Rsh z_1$?

$\rightarrow$ Add a "**cooperative sort**" reflecting the state of a and b

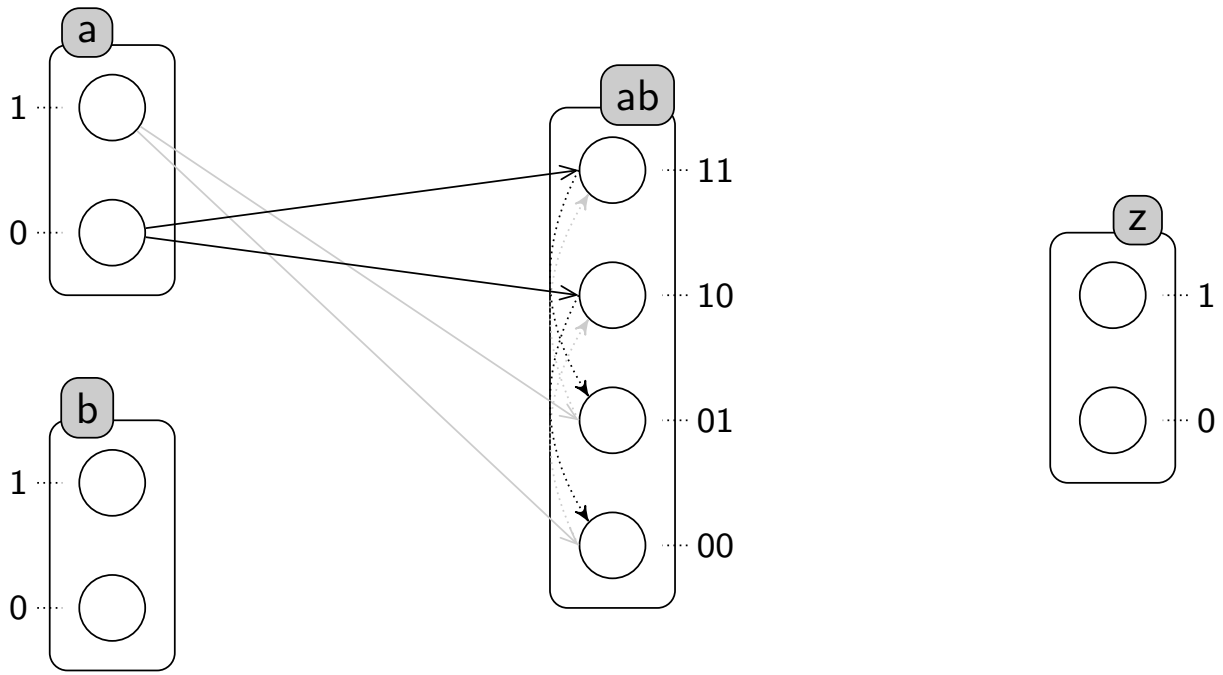


Refining with Cooperation

Allow **cooperation** between two genes

- How to express $(a_1 \wedge b_1) \rightarrow z_0 \nearrow z_1$?

$\rightarrow$ Add a "**cooperative sort**" reflecting the state of a and b

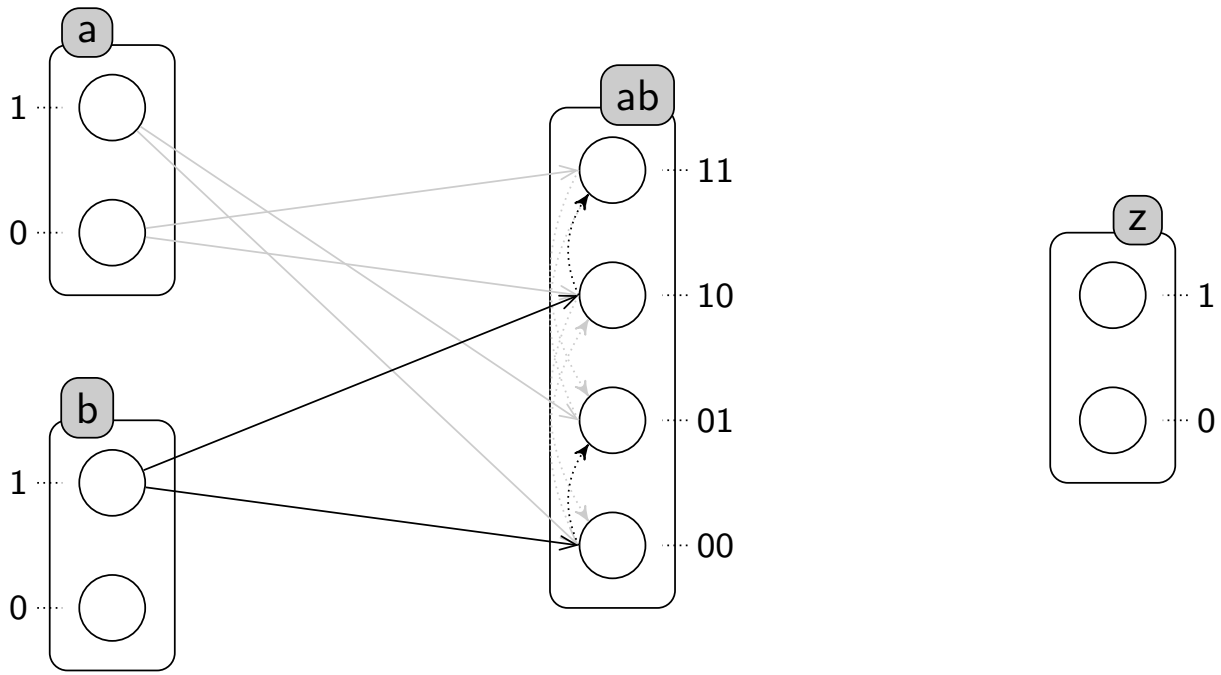


Refining with Cooperation

Allow **cooperation** between two genes

- How to express $(a_1 \wedge b_1) \rightarrow z_0 \Rsh z_1$?

$\rightarrow$ Add a "**cooperative sort**" reflecting the state of a and b

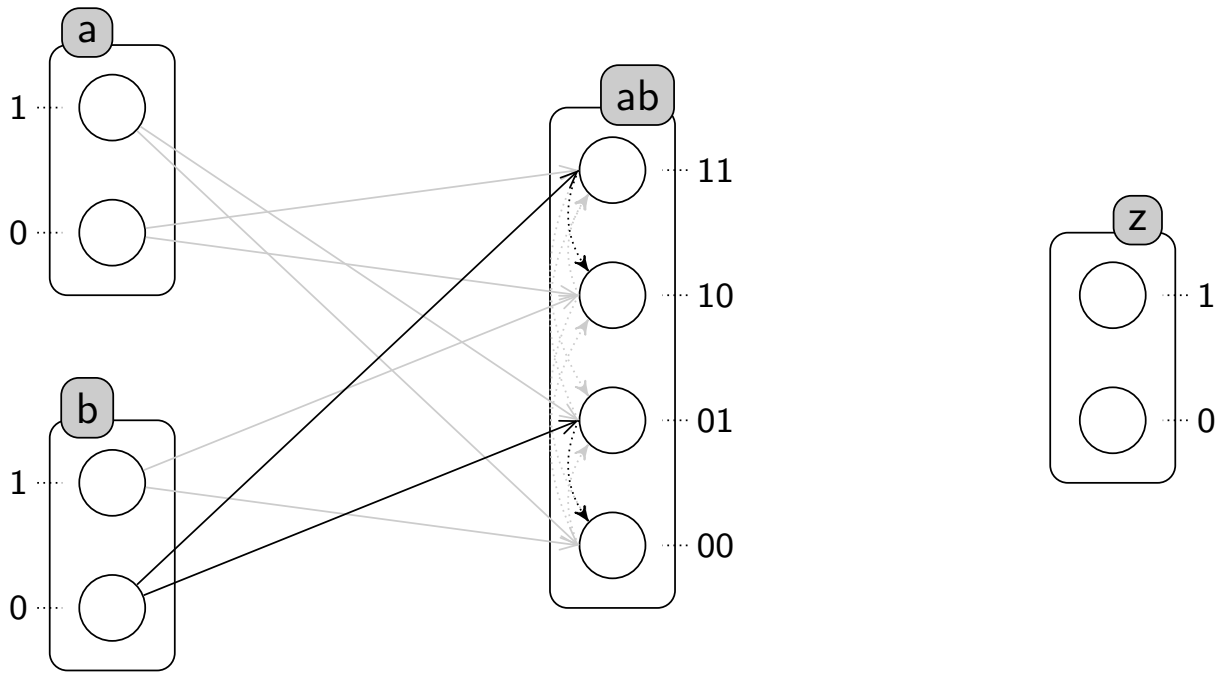


Refining with Cooperation

Allow **cooperation** between two genes

- How to express $(a_1 \wedge b_1) \rightarrow z_0 \Uparrow z_1$?

$\rightarrow$ Add a "**cooperative sort**" reflecting the state of a and b

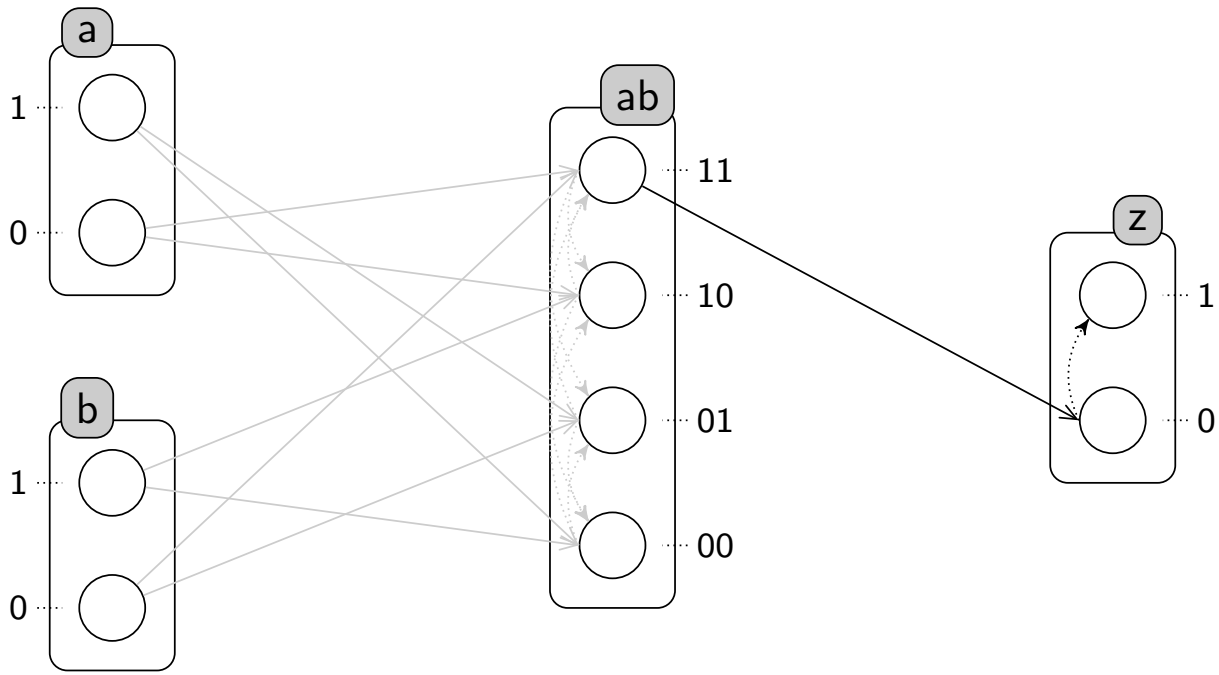


Refining with Cooperation

Allow **cooperation** between two genes

- How to express $(a_1 \wedge b_1) \rightarrow z_0 \Uparrow z_1$?

$\rightarrow$ Add a "**cooperative sort**" reflecting the state of a and b

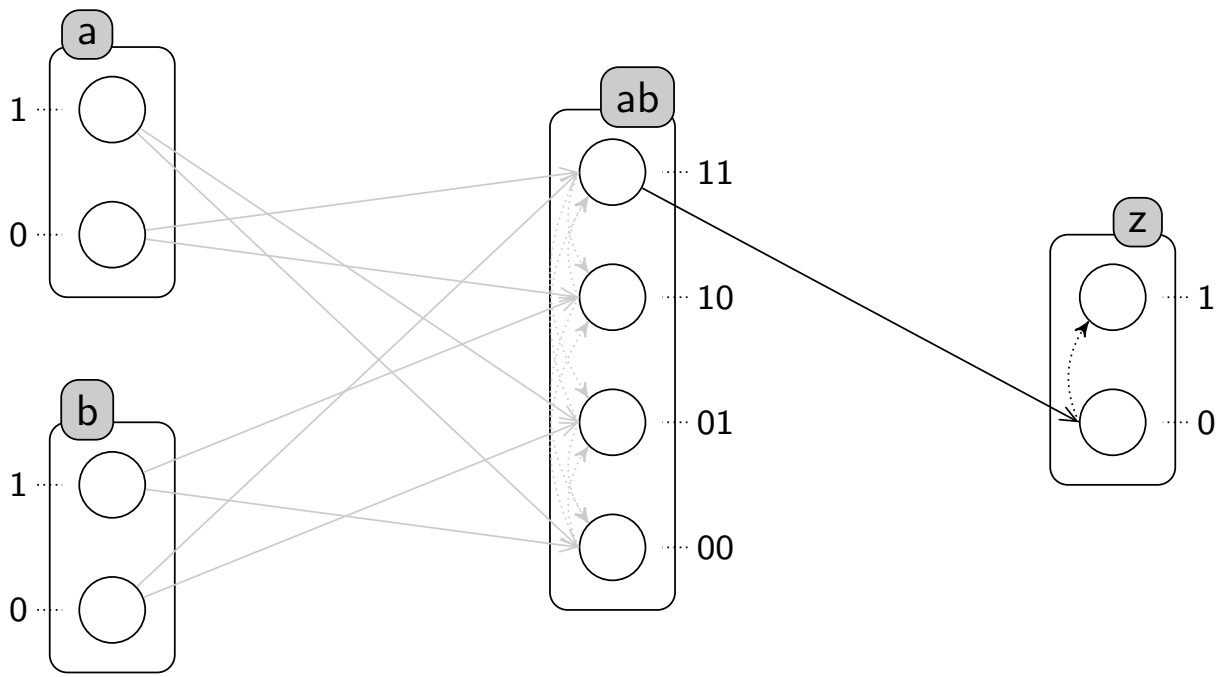


Refining with Cooperation

Allow **cooperation** between two genes

- How to express $(a_1 \wedge b_1) \rightarrow z_0 \uparrow z_1$?

$\rightarrow$ Add a "**cooperative sort**" reflecting the state of a and b



$\rightarrow$ Introduces a temporal shift (over-approximation)

Using Process Hitting for Interaction Graphs Study

The Interaction Graph is the historical discrete model
Adapted to and widespread in biological studies

Several tools to study Interaction Graphs
But for some results, the State Graph is needed

→ **Combinatorial explosion**

Process Hitting to study large Biological Regulatory Networks

→ Translation from Interaction Graphs + Refining

→ **Efficient static analysis**

Ongoing Work

Several student projects:

- Optimization of fixed points research
  - → OOP-oriented research (Java)
  - → CSP solver-oriented research (Choco) [choco.emn.fr]
- Study of the circadian clock
  - → use of the Process Hitting
- Graphical interface for the Process Hitting
  - → generalize its use by biologists
  - → show/use the static analysis results

Ongoing Work

Some developments:

- Framework evolutions
  - → Priorities between actions
  - → Neutralizing Edges (weak bisimulation)
- Working with Interaction Graphs
  - → Translation to Process Hitting
  - → Joint Actions (weak bisimulation)

Suggested personal subject:

- Using solvers to gain knowledge on Interaction Graphs
- Translation, then refining and constraints
  - → Information on the parametrization

Conclusion

Study of systems

→ **Combinatorial explosion**

Process Hitting

→ Simple modeling for **large systems**

→ Applications to Biological Regulatory Networks

→ Efficient static analysis

→ Further development (priorities, temporal data)

Conclusion

Study of systems

→ **Combinatorial explosion**

Process Hitting

→ Simple modeling for **large systems**

→ Applications to Biological Regulatory Networks

→ Efficient static analysis

→ Further development (priorities, temporal data)

Thank you

Bibliography

[Paulevé11] PhD thesis: *Modélisation, Simulation et Vérification des Grands Réseaux de Régulation Biologique*, October 2011, Nantes, France

[PMR10-TSE] Loïc Paulevé, Morgan Magnin, and Olivier Roux. *Tuning Temporal Features within the Stochastic π-Calculus*. IEEE Transactions on Software Engineering, 37(6):858-871, 2011.

[PRM10-TCSB] Loïc Paulevé, Morgan Magnin, and Olivier Roux. *Refining dynamics of gene regulatory networks in a stochastic π-calculus framework*. In Corrado Priami, Ralph-Johan Back, Ion Petre, and Erik de Vink, editors, Transactions on Computational Systems Biology XIII, volume 6575 of Lecture Notes in Computer Science, 171-191. Springer Berlin/Heidelberg, 2011.

[PMR12-MSCS] Loïc Paulevé, Morgan Magnin, and Olivier Roux. *Static analysis of biological regulatory networks dynamics using abstract interpretation*. Mathematical Structures in Computer Science, in press, 2012.

[PR10-CRAS] Loïc Paulevé and Adrien Richard. *Topological Fixed Points in Boolean Networks*. Comptes Rendus de l'Académie des Sciences - Series I - Mathematics, 348(15-16):825 - 828, 2010.

[RCB08] Adrien Richard, Jean-Paul Comet, and Gilles Bernot. *R. Thomas' logical method*, Apr. 2008. Invited at Tutorials on modelling methods and tools: Modelling a genetic switch and Metabolic Networks, Spring School on Modelling Complex Biological Systems in the Context of Genomics.