Scalable Formal Analysis of Dynamics of Biological Networks

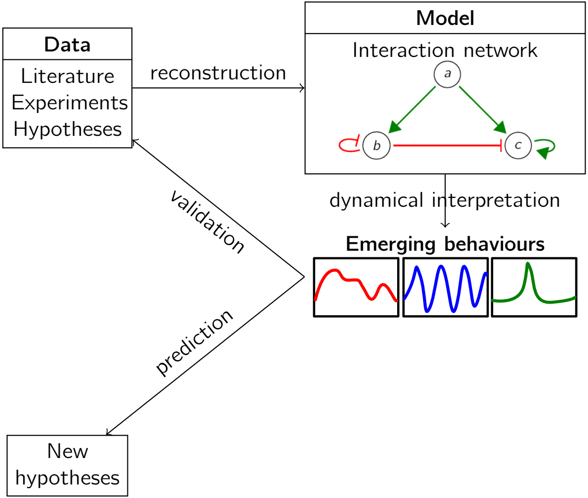
Loïc Paulevé

CNRS / LRI, Université Paris-Sud, France (Bioinfo team)
loic.pauleve@lri.fr
http://loicpauleve.name

May 23, 2014 – Masaryk University, Brno, Czech Republic

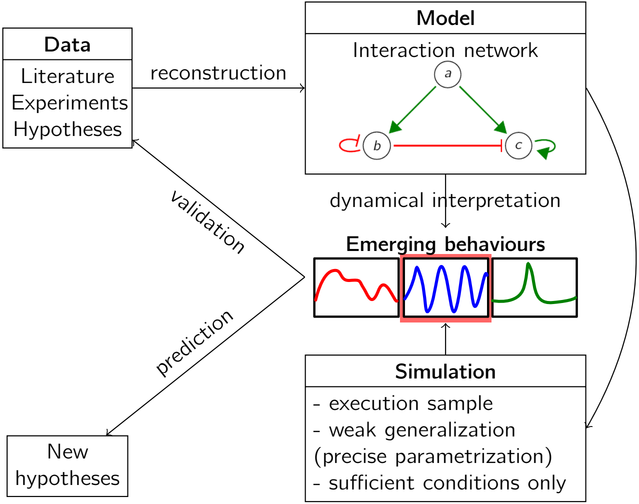
Formal Methods for Systems Biology

Aim: **understand, analyse, control** emerging dynamics.



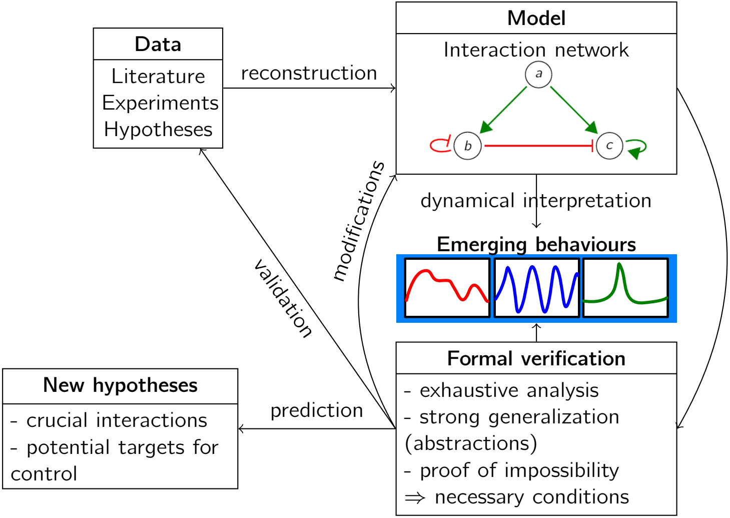
Formal Methods for Systems Biology

Aim: **understand, analyse, control** emerging dynamics.



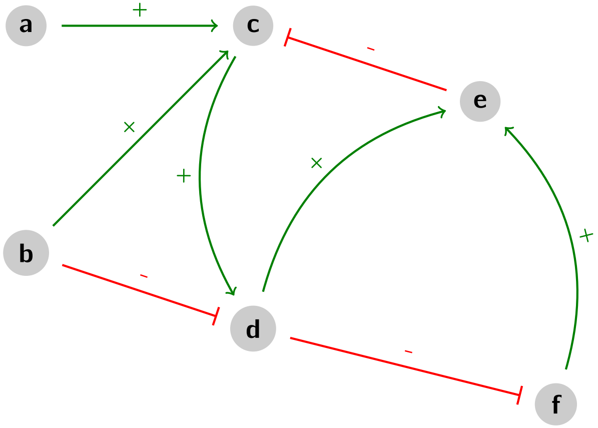
Formal Methods for Systems Biology

Aim: **understand, analyse, control** emerging dynamics.



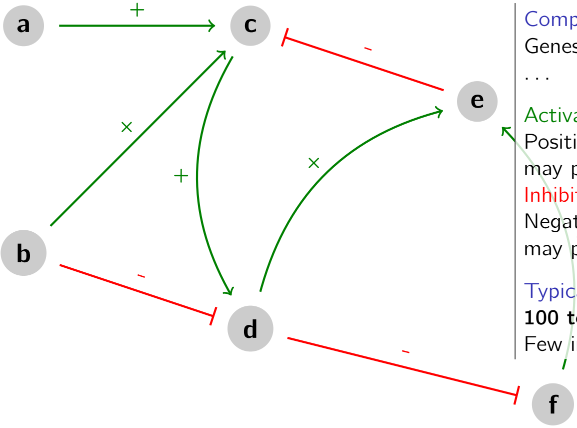
Interaction Networks

E.g., Regulatory or Signalling Networks



Interaction Networks

E.g., Regulatory or Signalling Networks



Components

Genes, proteins, complexes, . . .

Activations

Positive influence (*a* increase may promote *c* increase).

Inhibitions

Negative influence (*b* increase may promote *d* decrease).

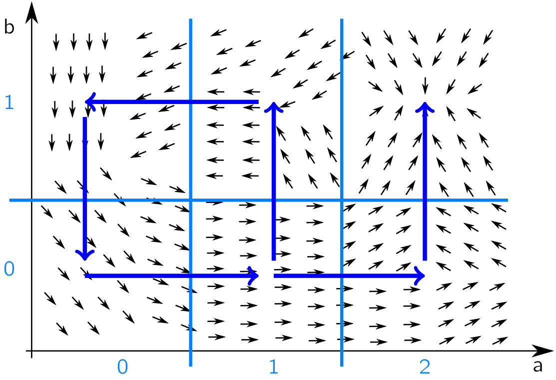
Typical settings

100 to +10,000 components

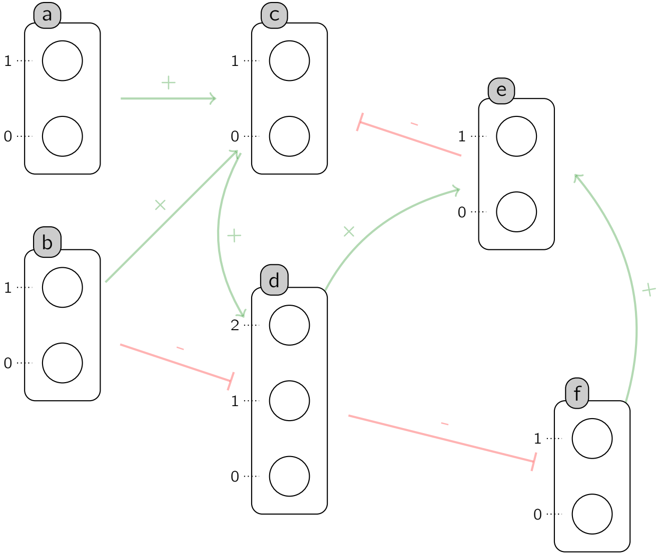
Few information on kinetics

Qualitative models

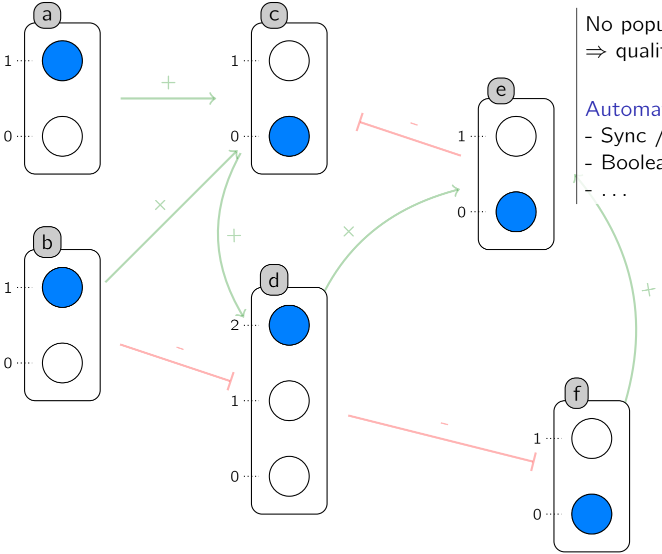
- Assume a quantization of the species population/concentration.
- Have a finite discrete state space (typically 2^n states).
- Non-deterministic dynamics.



Qualitative Models for Interaction Networks



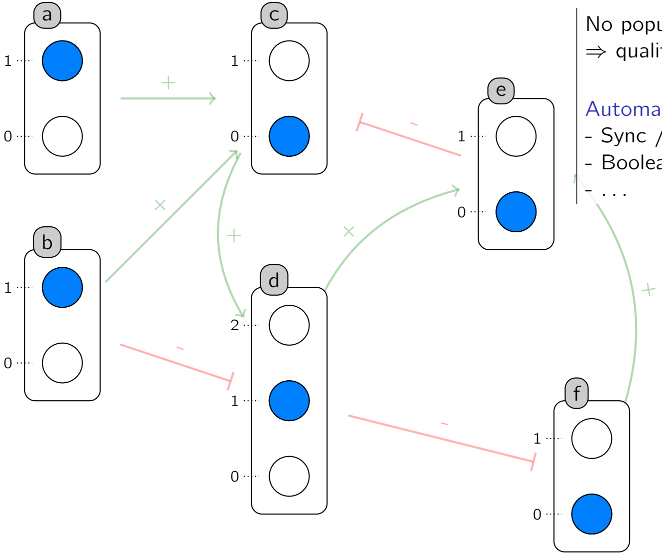
Qualitative Models for Interaction Networks



No population
⇒ qualitative levels

Automata networks
- Sync / async
- Boolean/multi-valued
- . . .

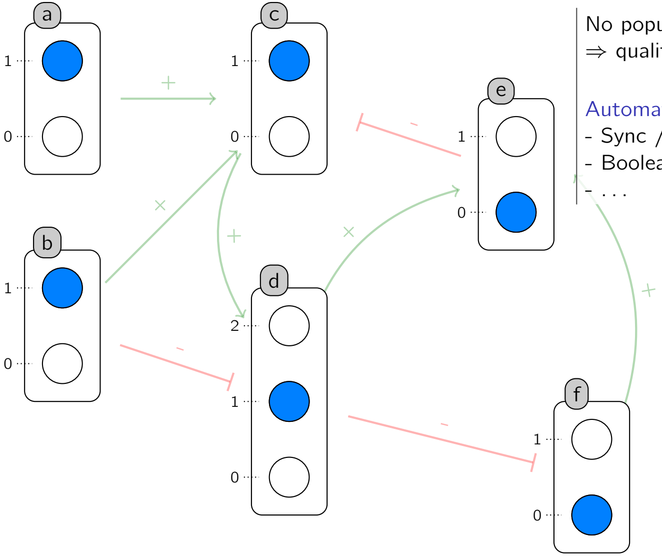
Qualitative Models for Interaction Networks



No population
⇒ qualitative levels

Automata networks
- Sync / async
- Boolean/multi-valued
- . . .

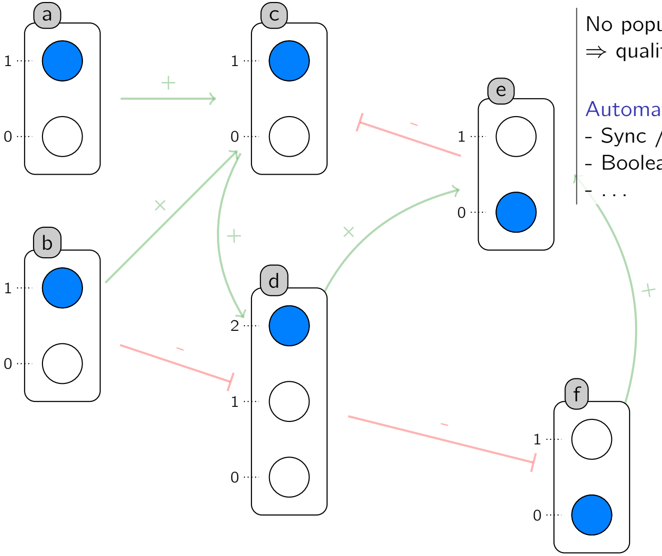
Qualitative Models for Interaction Networks



No population
⇒ qualitative levels

Automata networks
- Sync / async
- Boolean/multi-valued
- . . .

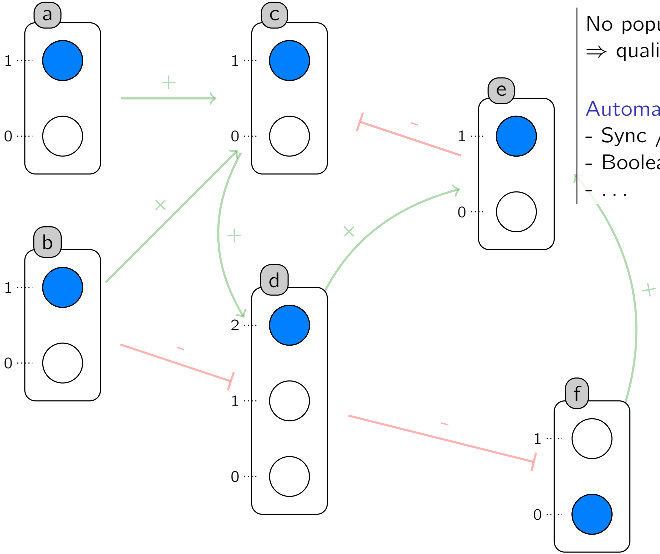
Qualitative Models for Interaction Networks



No population
⇒ qualitative levels

Automata networks
- Sync / async
- Boolean/multi-valued
- . . .

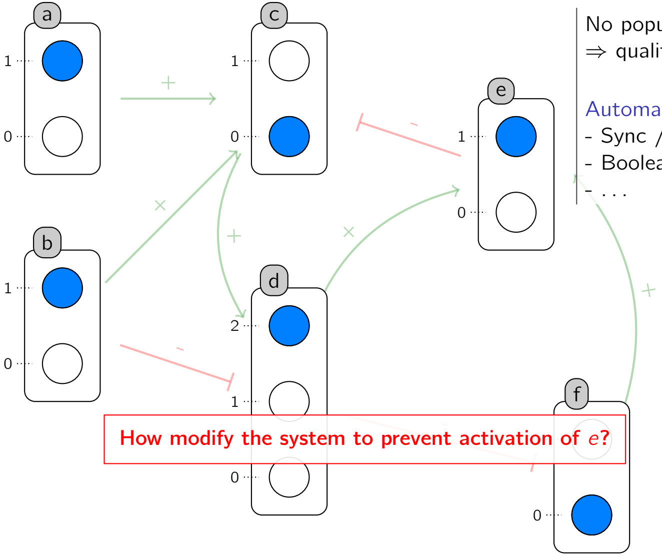
Qualitative Models for Interaction Networks



No population
⇒ qualitative levels

Automata networks
- Sync / async
- Boolean/multi-valued
- . . .

Qualitative Models for Interaction Networks



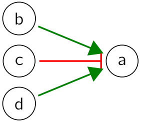
No population
⇒ qualitative levels

Automata networks
- Sync / async
- Boolean/multi-valued
- . . .

Issues with Large Interaction Networks

Modelling issues

- Partially-specified interactions.
- Boolean networks need to be fully specified (deterministic Boolean function f_a).
- Intractable enumeration of all models.

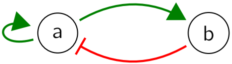


Analysis issues

- Combinatorial explosion of behaviours (e.g. 2^{100} to $2^{10\,000}$ states).
- Large range of initial conditions to consider.
- Difficult to extract comprehensive proofs of (im)possibility.

Failure of classical model-checking techniques,
Need **new formal approaches** to capture dynamics of large networks

Static Analysis based on Interaction Graph



Relationships between the interaction graph and dynamical properties:

- Multi-stationnarity requires a positive circuit (René Thomas conjecture) [Soule in ComPlexUs, 2003] [Richard, Comet in Discrete Appl. Math., 2007].
- Sustained oscillations require a negative circuit (René Thomas conjecture) [Remy, *et al.* in Adv. Appl. Math., 2008] [Richard in Adv. Appl. Math., 2010].
- The maximum number of fixed points can be characterized [Aracena in Bul. of Mathematical Biology, 2008]; [Richard in Discrete Appl. Math., 2009].
- Topological Fixed Points [Paulevé, Richard in CRAS 2010].
- Difference between synchronous/asynchronous update [Noual, Regnault, Sené]
- etc.

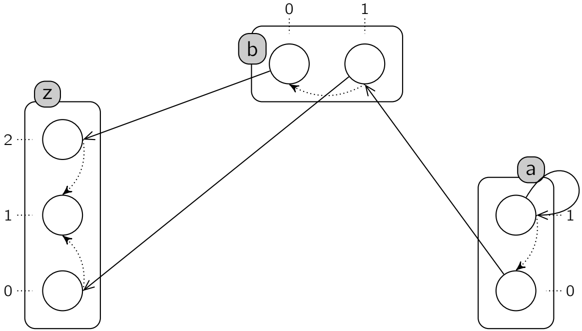
(See [Paulevé, Richard at SASB'11] for a short survey).

Outline

Outline

The Process Hitting Framework

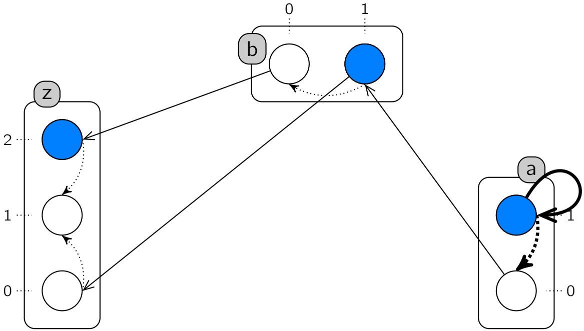
[Paulevé, Magnin, Roux in TCSB 2011]



- Automata: a,b,z; Processes: $a_0, a_1, b_0, b_1, z_0, z_1, z_2$;
- Actions: a_0 *hits* b_1 to make it *bounce* to b_0, . . . ;
- States: $\langle a_1, b_1, z_2\rangle$, $\langle a_0, b_1, z_2\rangle$, $\langle a_0, b_0, z_2\rangle$, . . . ;

The Process Hitting Framework

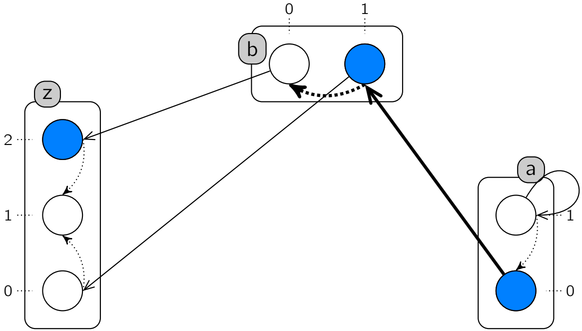
[Paulevé, Magnin, Roux in TCSB 2011]



- Automata: a,b,z; Processes: $a_0, a_1, b_0, b_1, z_0, z_1, z_2$;
- Actions: a_0 *hits* b_1 to make it *bounce* to b_0, . . . ;
- States: $\langle a_1, b_1, z_2\rangle$, $\langle a_0, b_1, z_2\rangle$, $\langle a_0, b_0, z_2\rangle$, . . . ;

The Process Hitting Framework

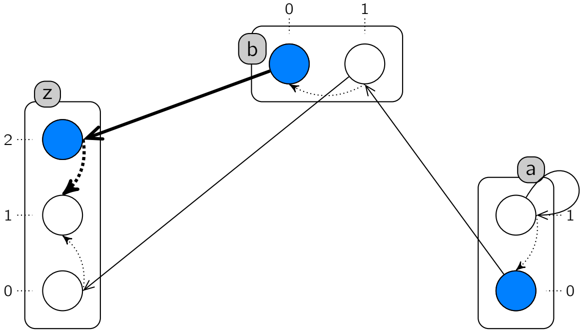
[Paulevé, Magnin, Roux in TCSB 2011]



- Automata: a,b,z; Processes: $a_0, a_1, b_0, b_1, z_0, z_1, z_2$;
- Actions: a_0 *hits* b_1 to make it *bounce* to b_0, . . . ;
- States: $\langle a_1, b_1, z_2\rangle$, $\langle a_0, b_1, z_2\rangle$, $\langle a_0, b_0, z_2\rangle$, . . . ;

The Process Hitting Framework

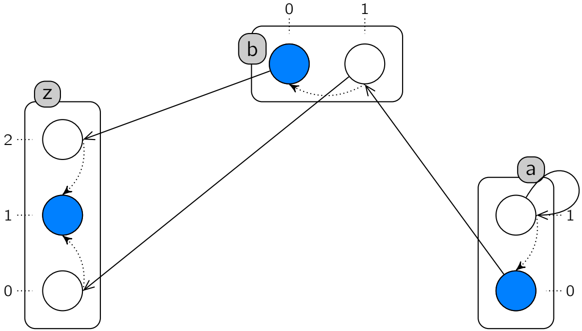
[Paulevé, Magnin, Roux in TCSB 2011]



- Automata: a,b,z; Processes: $a_0, a_1, b_0, b_1, z_0, z_1, z_2$;
- Actions: a_0 *hits* b_1 to make it *bounce* to b_0, . . . ;
- States: $\langle a_1, b_1, z_2\rangle$, $\langle a_0, b_1, z_2\rangle$, $\langle a_0, b_0, z_2\rangle$, . . . ;

The Process Hitting Framework

[Paulevé, Magnin, Roux in TCSB 2011]



- Automata: a,b,z; Processes: $a_0, a_1, b_0, b_1, z_0, z_1, z_2$;
- Actions: a_0 *hits* b_1 to make it *bounce* to b_0, . . . ;
- States: $\langle a_1, b_1, z_2\rangle$, $\langle a_0, b_1, z_2\rangle$, $\langle a_0, b_0, z_2\rangle$, . . . ;

The Process Hitting

Why a new framework?

Features of the Process Hitting

- Simple formalism; but enough to model networks dynamics.
- Special class of Asynchronous Automata Networks (or Petri Nets).
- A transition is triggered by only one process (biological or logical).

Advantages for Modelling

- Atomic description of transitions.
- Allows to model networks with partial knowledge on cooperations
  $\Rightarrow$ encodes non-deterministic Boolean functions; e.g.:
  $$f_a(x) = \begin{cases} 1 & \text{if } x_b = 1 \vee x_c = 1 \\ 0 & \text{if } x_b = 0 \vee x_c = 0 \end{cases}$$

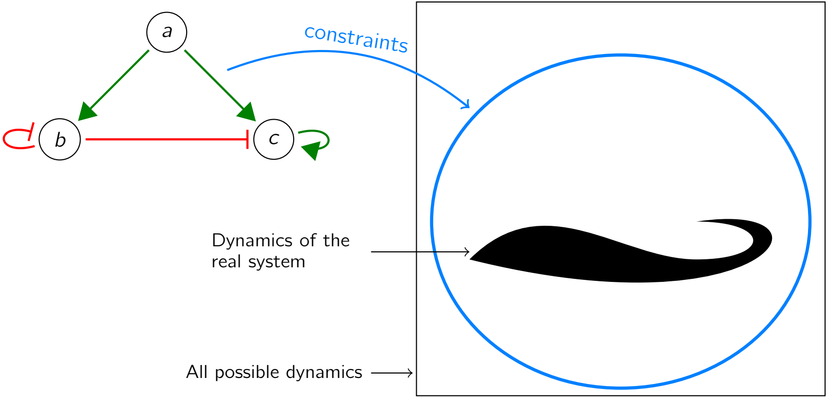
Advantages for Analysis

- Easy fixed point derivation (not shown in this talk).
- Very efficient causality analysis;
- allows highly scalable reachability analysis.

Limitations

- Synchrone update is complex to encode (but possible);
- Over-approximation approach: focus mainly on necessary conditions
  (but work in progress for the counterpart).

Generalised Dynamics of Interaction Networks

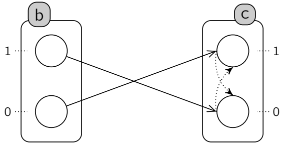


Dynamics over-approximation

- A component can not increase if none effective activator is present.
- A component can not decrease if none effective inhibitor is present.

Modelling Regulation with the Process Hitting

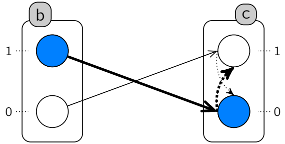
Boolean case:



- Independant regulations, automatic encoding of interaction graphs.
- Without knowledge of cooperation between regulators.
  $\Rightarrow$ most permissive dynamics of the network.

Modelling Regulation with the Process Hitting

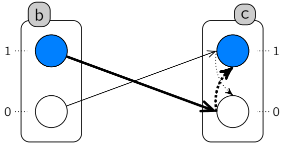
Boolean case:



- Independant regulations, automatic encoding of interaction graphs.
- Without knowledge of cooperation between regulators.
  ⇒ most permissive dynamics of the network.

Modelling Regulation with the Process Hitting

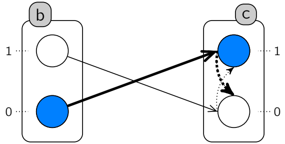
Boolean case:



- Independant regulations, automatic encoding of interaction graphs.
- Without knowledge of cooperation between regulators.
  $\Rightarrow$ most permissive dynamics of the network.

Modelling Regulation with the Process Hitting

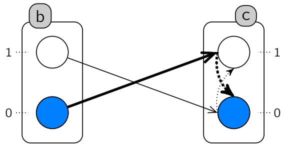
Boolean case:



- Independant regulations, automatic encoding of interaction graphs.
- Without knowledge of cooperation between regulators.
  ⇒ most permissive dynamics of the network.

Modelling Regulation with the Process Hitting

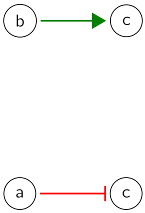
Boolean case:

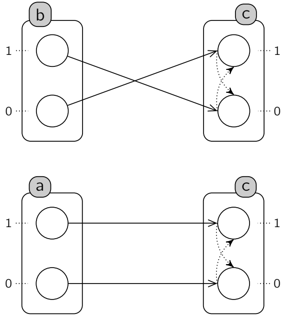


- Independant regulations, automatic encoding of interaction graphs.
- Without knowledge of cooperation between regulators.
  ⇒ most permissive dynamics of the network.

Modelling Regulation with the Process Hitting

Boolean case:

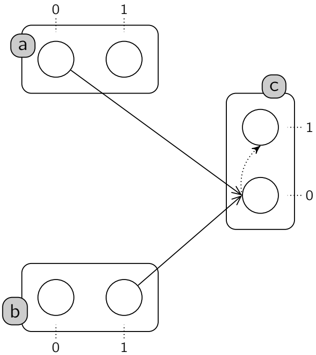




- Independant regulations, automatic encoding of interaction graphs.
- Without knowledge of cooperation between regulators.
  ⇒ most permissive dynamics of the network.

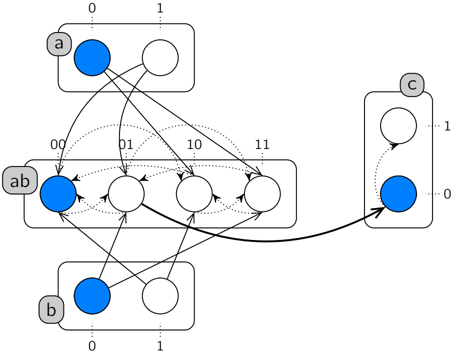
Refining with Cooperation

- Constraint: $c_0 \upharpoonright c_1$ when a_0 and b_1 are present.
- Introduction of a cooperative automata reflecting the state of a and b.



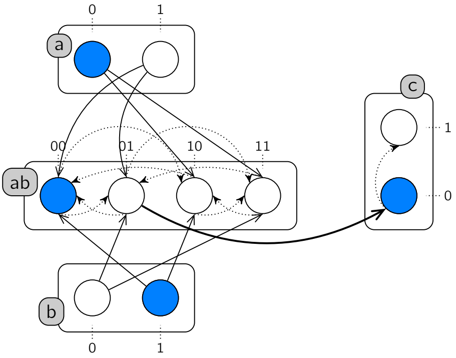
Refining with Cooperation

- Constraint: $c_0 \upharpoonright\!\!\rightarrow c_1$ when a_0 and b_1 are present.
- Introduction of a cooperative automata reflecting the state of a and b.



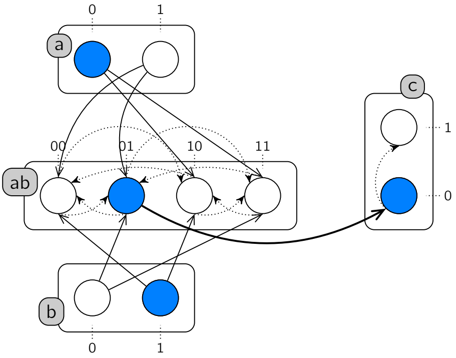
Refining with Cooperation

- Constraint: $c_0 \rightarrow c_1$ when a_0 and b_1 are present.
- Introduction of a cooperative automata reflecting the state of a and b.



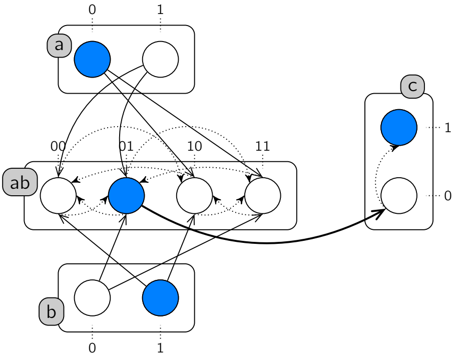
Refining with Cooperation

- Constraint: $c_0 \rightarrow c_1$ when a_0 and b_1 are present.
- Introduction of a cooperative automata reflecting the state of *a* and *b*.



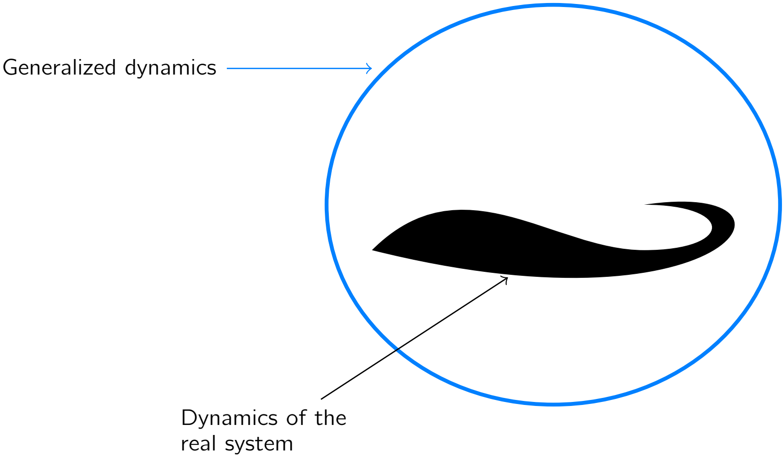
Refining with Cooperation

- Constraint: $c_0 \upharpoonright c_1$ when a_0 and b_1 are present.
- Introduction of a cooperative automata reflecting the state of *a* and *b*.

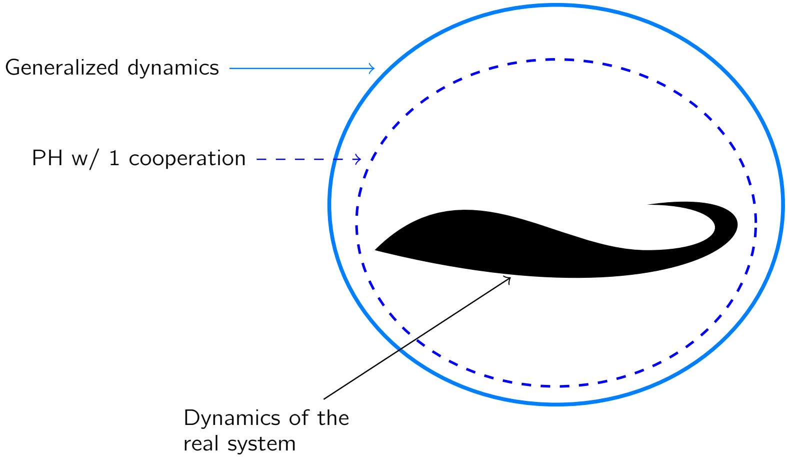


⇒ The Process Hitting can model *any* interaction network with partial knowledge on the cooperations (over-approximation of dynamics).

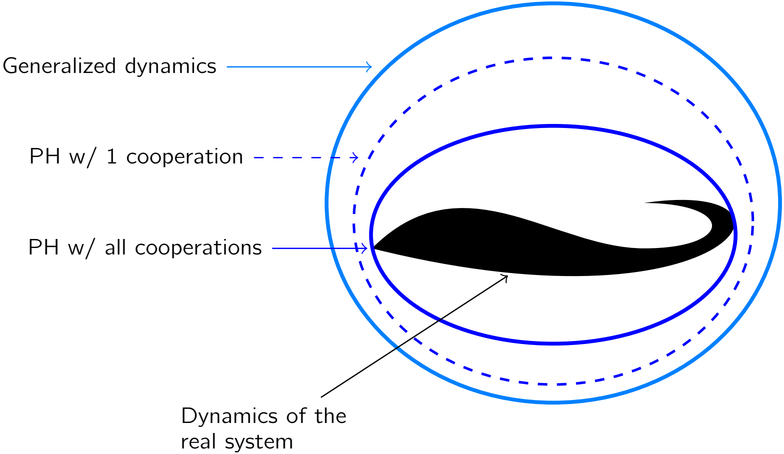
Abstraction Relationships



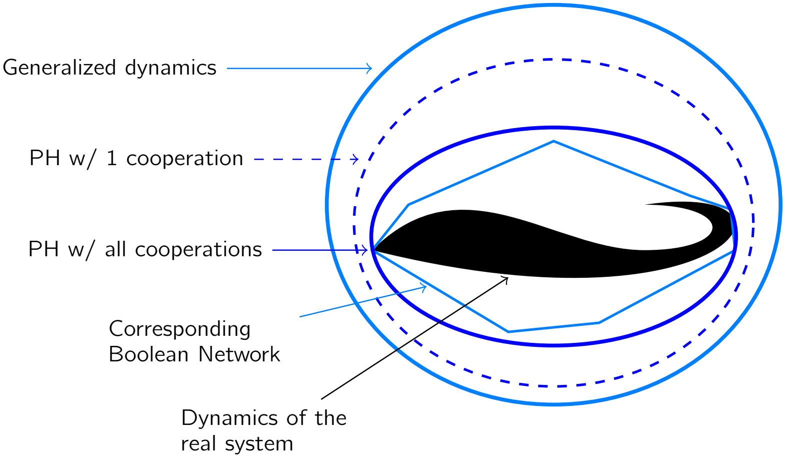
Abstraction Relationships



Abstraction Relationships

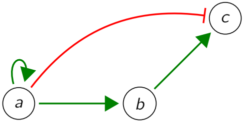


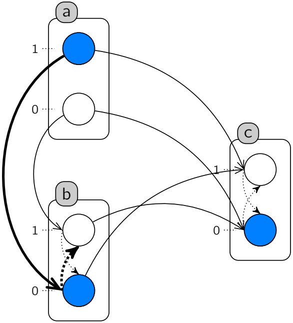
Abstraction Relationships



Toy example

Incoherent feed-forward loop

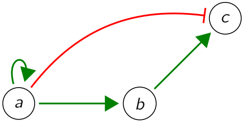


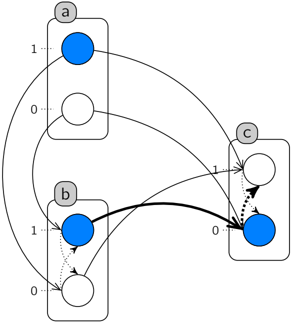


$$\langle a_1, b_0, c_0\rangle$$
$$\downarrow$$
$$\langle a_1, b_1, c_0\rangle \longleftrightarrow \langle a_1, b_1, \mathbf{c_1}\rangle$$

Toy example

Incoherent feed-forward loop

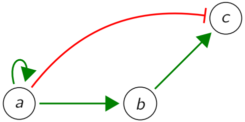


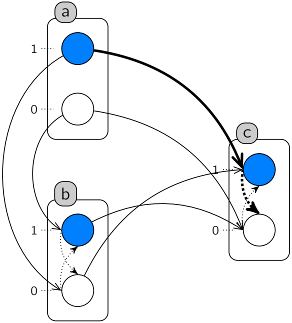


$$\begin{array}{ccc} \langle a_1, b_0, c_0\rangle & & \\ \downarrow & & \\ \langle a_1, b_1, c_0\rangle & \longleftrightarrow & \langle a_1, b_1, \mathbf{c_1}\rangle \end{array}$$

Toy example

Incoherent feed-forward loop

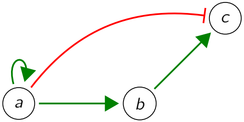


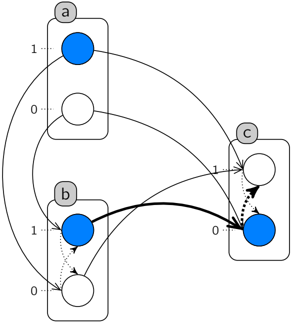


$$\begin{tikzcd}
\langle a_1, b_0, c_0 \rangle \arrow[d] & \\
\langle a_1, b_1, c_0 \rangle \arrow[r, leftrightarrow] & \langle a_1, b_1, \mathbf{c_1} \rangle
\end{tikzcd}$$

Toy example

Incoherent feed-forward loop

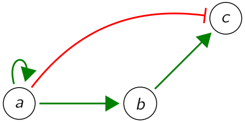


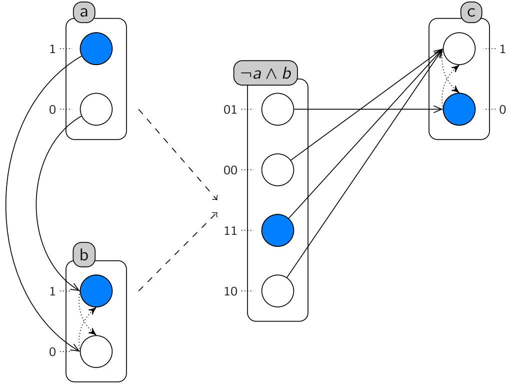


$$\langle a_1, b_0, c_0\rangle$$
$$\downarrow$$
$$\langle a_1, b_1, c_0\rangle \longleftrightarrow \langle a_1, b_1, \mathbf{c_1}\rangle$$

Toy example

Incoherent feed-forward loop

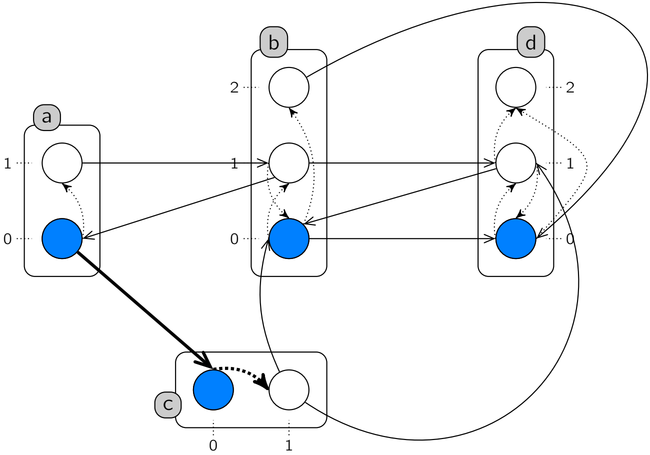




$$
\begin{tikzcd}
\langle a_1, b_0, ab_{00}, c_0\rangle \arrow[r] \arrow[d] & \langle a_1, b_0, ab_{10}, c_0\rangle \arrow[d] \\
\langle a_1, b_1, ab_{00}, c_0\rangle \arrow[r] \arrow[d] & \langle a_1, b_1, ab_{10}, c_0\rangle \arrow[d] \\
\langle a_1, b_1, ab_{01}, c_0\rangle \arrow[r] \arrow[d] & \color{blue}{\langle a_1, b_1, ab_{11}, c_0\rangle} \\
\langle a_1, b_1, ab_{01}, \mathbf{c_1}\rangle \arrow[d] & \\
\langle a_1, b_1, ab_{11}, \mathbf{c_1}\rangle \arrow[uur, bend right] &
\end{tikzcd}
$$

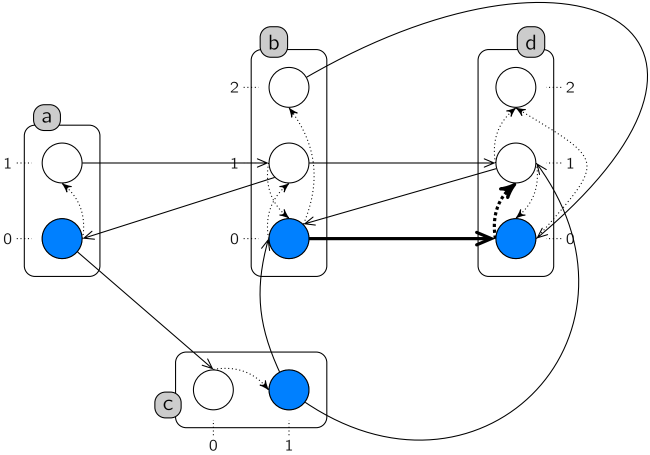
Outline

Looking for Scenarios



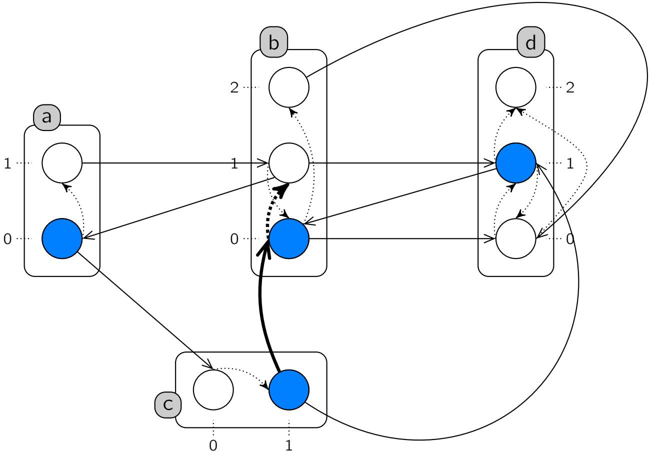
$\mathbf{a_0 \rightarrow c_0 \upharpoonright c_1} :: b_0 \rightarrow d_0 \upharpoonright d_1 :: c_1 \rightarrow b_0 \upharpoonright b_1 :: b_1 \rightarrow d_1 \upharpoonright d_2$

Looking for Scenarios



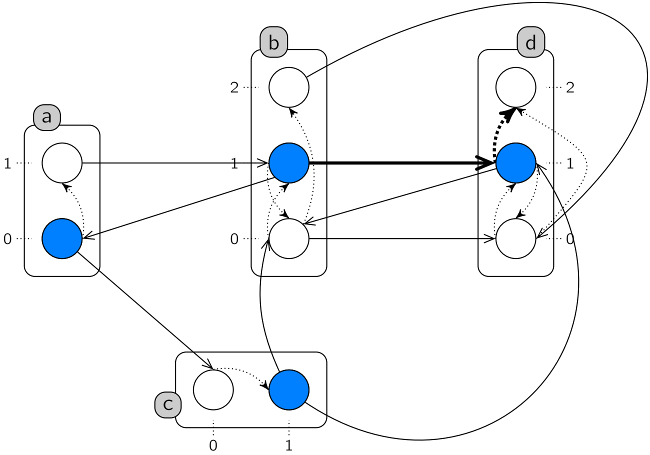
$a_0 \rightarrow c_0 \upharpoonright c_1 :: \mathbf{b_0 \rightarrow d_0 \upharpoonright d_1} :: c_1 \rightarrow b_0 \upharpoonright b_1 :: b_1 \rightarrow d_1 \upharpoonright d_2$

Looking for Scenarios



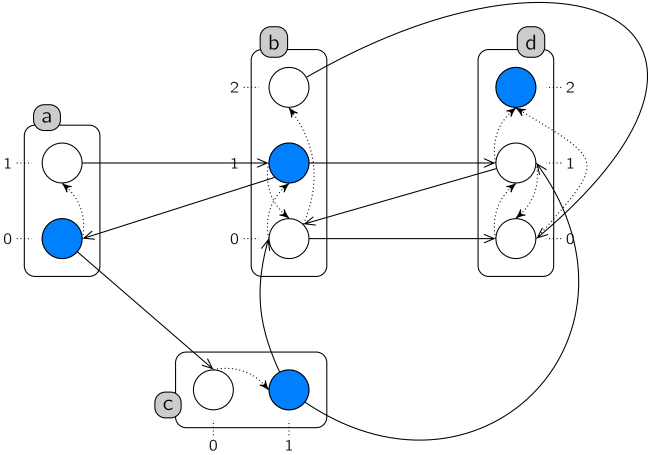
$a_0 \rightarrow c_0 \upharpoonright c_1 :: b_0 \rightarrow d_0 \upharpoonright d_1 :: \mathbf{c_1 \rightarrow b_0 \upharpoonright b_1} :: b_1 \rightarrow d_1 \upharpoonright d_2$

Looking for Scenarios



$a_0 \rightarrow c_0 \upharpoonright c_1 :: b_0 \rightarrow d_0 \upharpoonright d_1 :: c_1 \rightarrow b_0 \upharpoonright b_1 :: \mathbf{b_1 \rightarrow d_1 \upharpoonright d_2}$

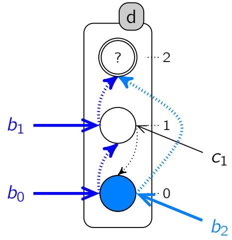
Looking for Scenarios



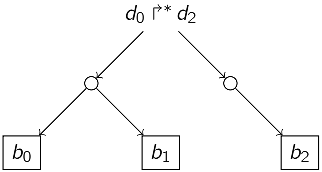
$$a_0 \rightarrow c_0 \upharpoonright c_1 :: b_0 \rightarrow d_0 \upharpoonright d_1 :: c_1 \rightarrow b_0 \upharpoonright b_1 :: b_1 \rightarrow d_1 \upharpoonright d_2$$

Local Causality

Minimal solutions

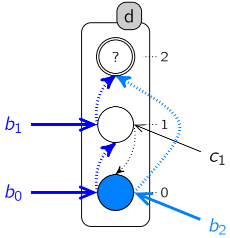


$\textbf{sol}(d_0 \Rsh^* d_2) = \{\{b_0, b_1\}, \{b_2\}\}.$

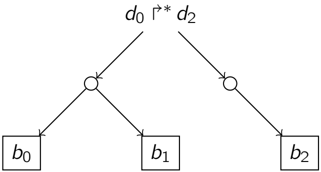


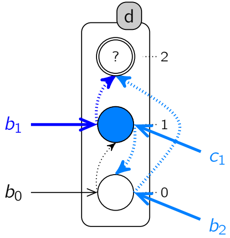
Local Causality

Minimal solutions

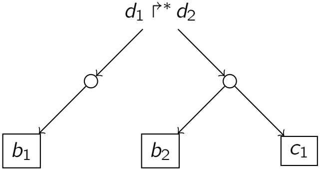


$\mathbf{sol}(d_0 \Rsh^* d_2) = \{\{b_0, b_1\}, \{b_2\}\}.$



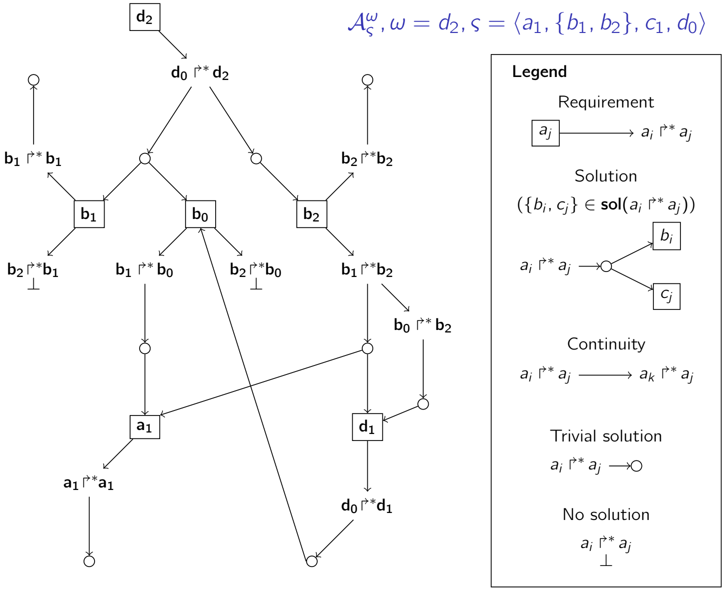


$\mathbf{sol}(d_1 \Rsh^* d_2) = \{\{b_1\}, \{b_2, c_1\}\}.$



Graph of Local Causality

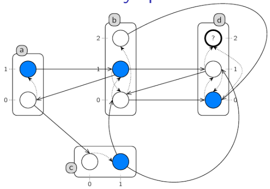
$\mathcal{A}^{\omega}_{\varsigma}, \omega = d_2, \varsigma = \langle a_1, \{b_1, b_2\}, c_1, d_0 \rangle$



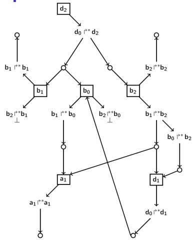
Efficient Reachability Analysis

Abstract interpretation of Process Hitting dynamics

Process Hitting
+ reachability question



Graph of Local Causality



Necessary/sufficient conditions

Yes / No / Maybe

Reach a_i, then b_j, etc.

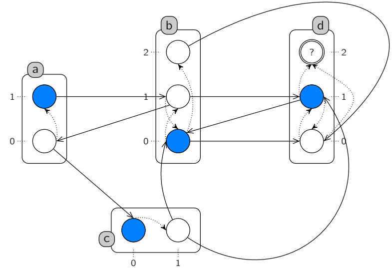
- Over- and under-approximations of local rechability properties.
- Low complexity: poly(nb. automata) $\times$ exp(nb of procs in one automaton)

$\Longrightarrow$ efficient with a small number of processes per automaton, while a very large number of automata can be handled.

[**Mathematical Structures in Computer Science** (2012); workshop **SASB**'10]

Over-approximation of Reachability

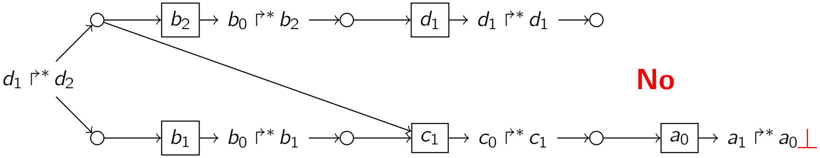
Example



Necessary condition for reaching d_2:

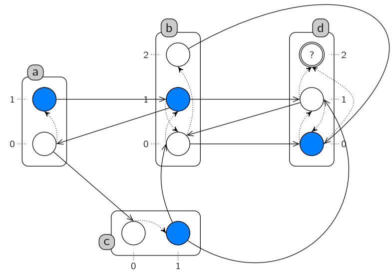
There exists a traversal of the GLC s.t.:

- objective → follow at least one solution;
- process → follow all objectives;
- no cycle.



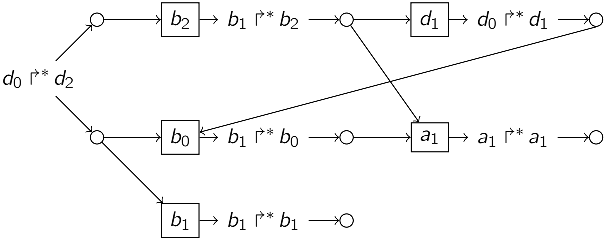
No

Over-approximation of Reachability
Example



Necessary condition for reaching d_2:
There exists a traversal of the GLC s.t.:

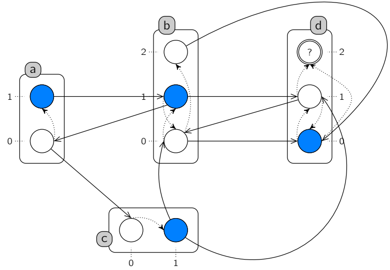
- objective $\rightarrow$ follow at least one solution;
- process $\rightarrow$ follow all objectives;
- no cycle.



Inconc

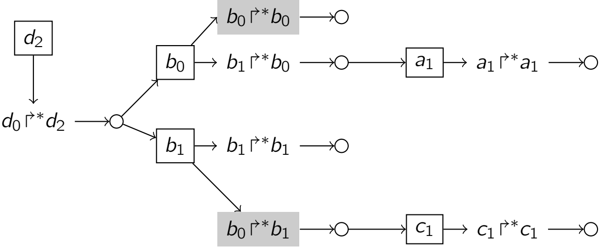
Under-approximation of Reachability

Example



Sufficient condition for reaching d_2:

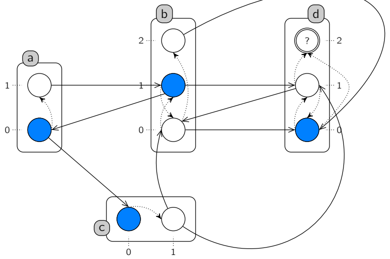
- GLC' has no cycle;
- each objective has at least one solution.



Yes

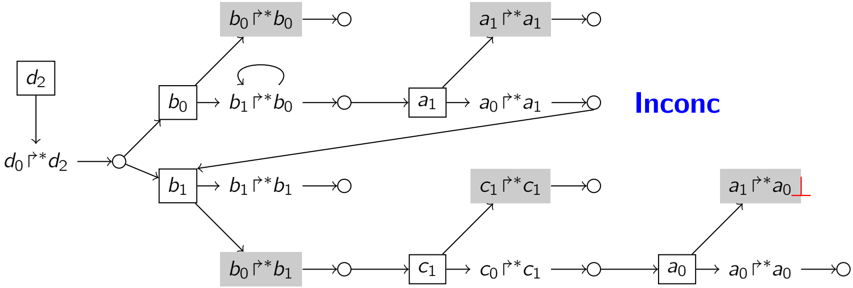
Under-approximation of Reachability

Example



Sufficient condition for reaching d_2:

- GLC' has no cycle;
- each objective has at least one solution.

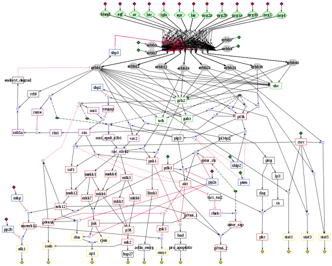


Inconc

Applications

- Signalling networks.
- Wide-range of biological/arbitrary reachability analysis.
- Always conclusive.

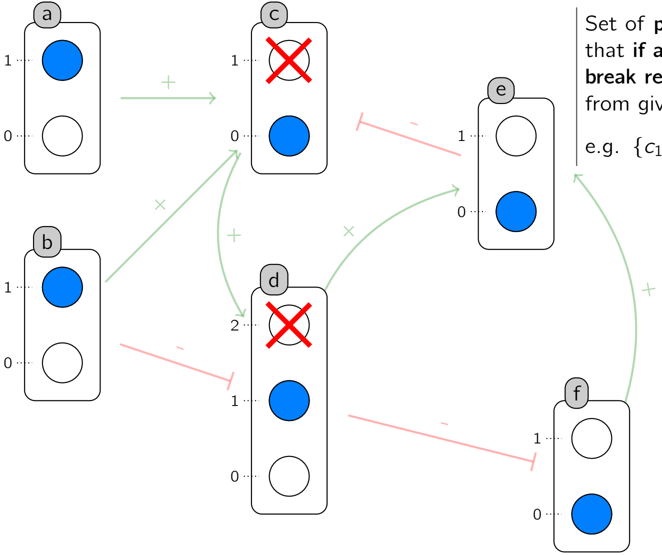
Model	Biocham[1]	libDDD[2]	PINT[3]
EGFR 20	[3s-KO]	[1s-150s]	0.007s
TCR 40	[1s-KO]	[0.6s-KO]	0.004s
TCR 94	KO	KO	0.030s
EGFR 104	KO	KO	0.050s



[1] http://contraintes.inria.fr/biocham (using NuSMV2)
[2] http://move.lip6.fr/software/DDD
[3] http://loicpauleve.name/pint

Cut Sets for Reachability

[Paulevé et al. at CAV'13]

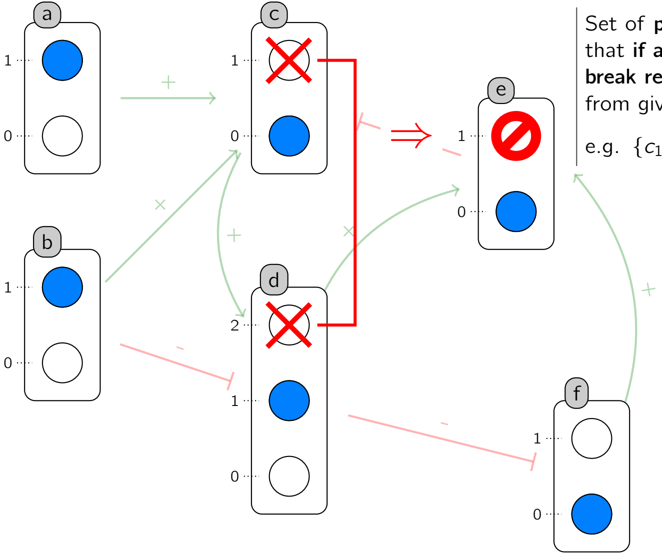


Set of **processes** that **if all disabled** **break reachability** from given initial states

e.g. $\{c_1, d_2\}$

Cut Sets for Reachability

[Paulevé et al. at CAV'13]

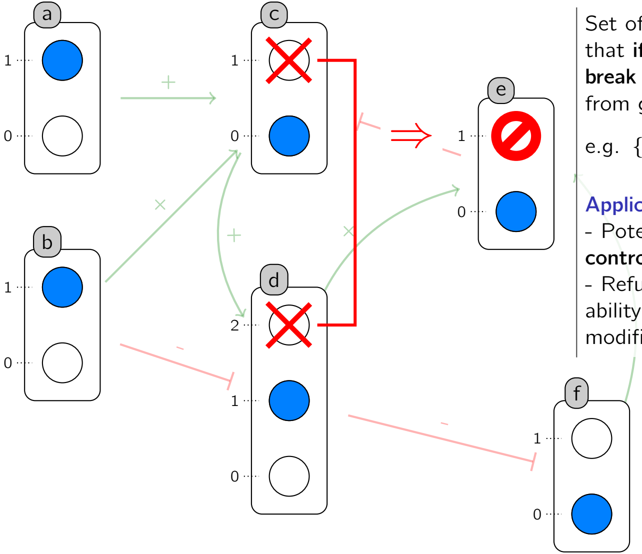


Set of **processes** that **if all disabled break reachability** from given initial states

e.g. $\{c_1, d_2\}$

Cut Sets for Reachability

[Paulevé et al. at CAV'13]



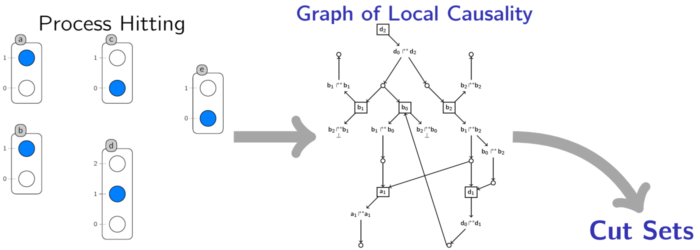
Set of **processes** that **if all disabled break reachability** from given initial states

e.g. $\{c_1, d_2\}$

Applications

- Potential **control targets**
- Refute model if reachability still occurs in the modified (real) system

Cut Sets for Reachability



Algorithm

- Graph flooding algorithm.
- Computes all cut sets at once: no enumeration of candidates.
- Very efficient with large networks.

Returned cut sets

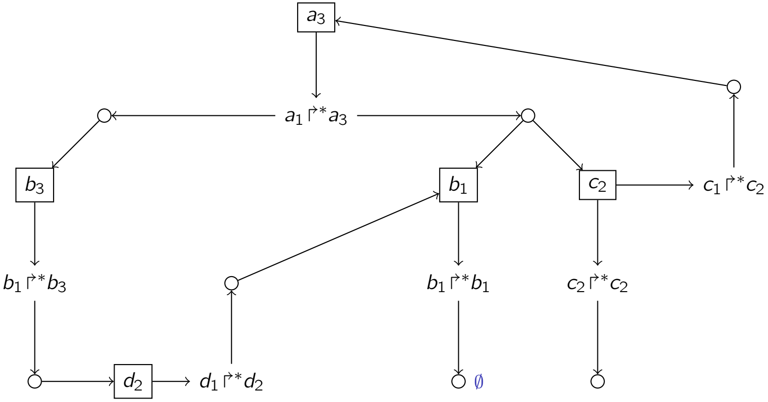
- All valid (break the concerned reachability).
- Some may be missed, some may be non-minimal.

Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

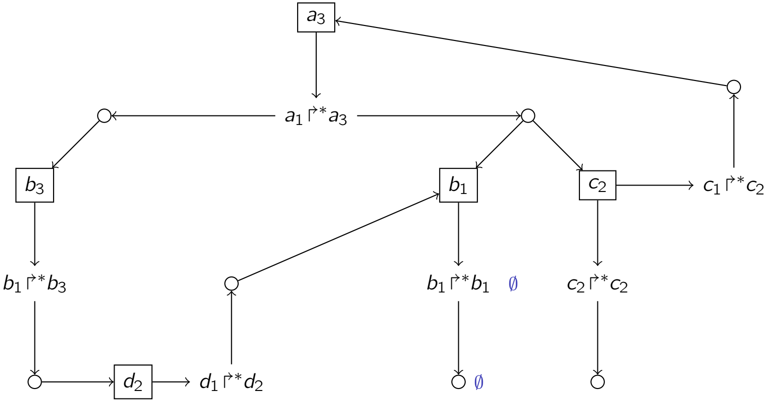


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

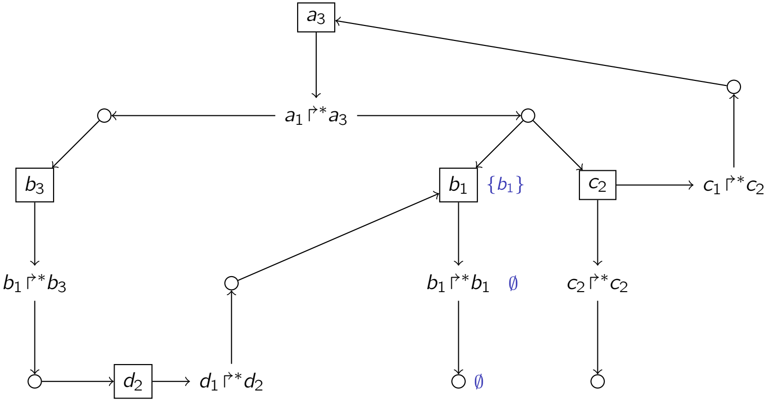


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

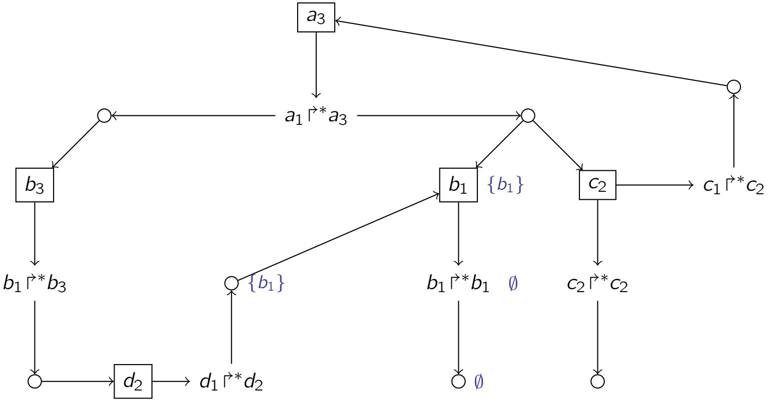


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

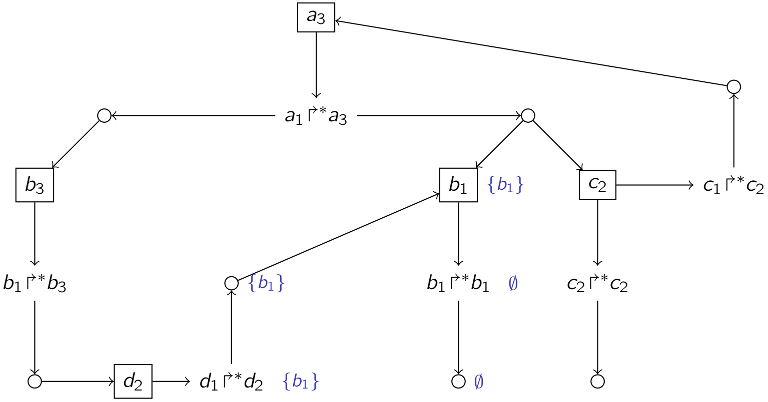


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

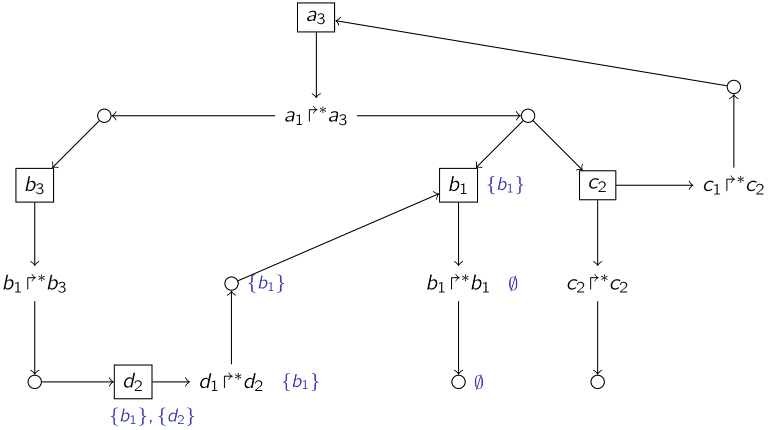


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

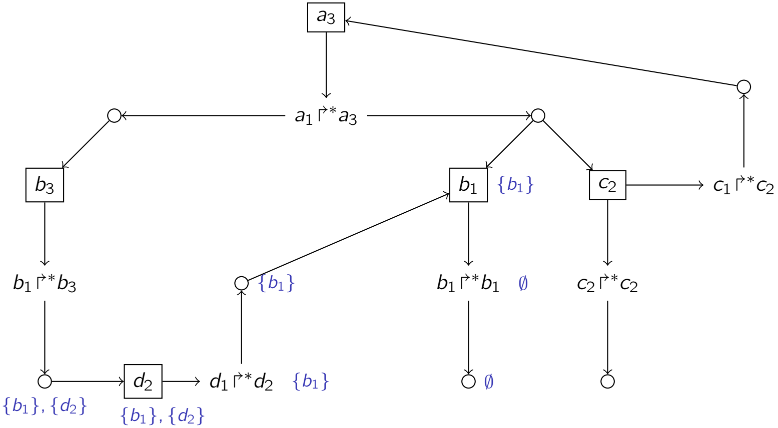


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

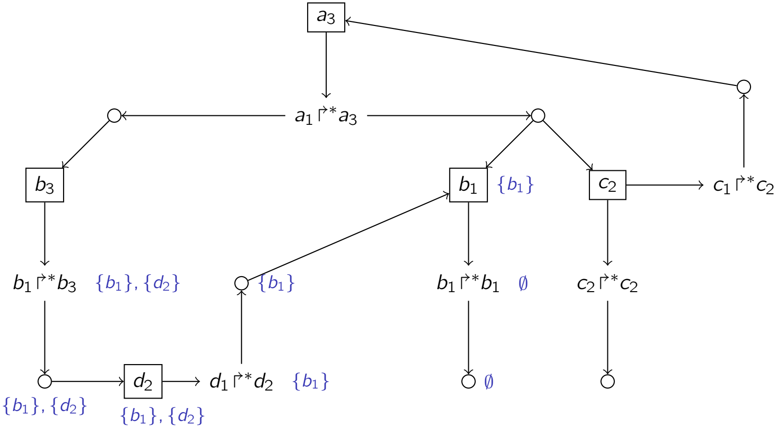


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

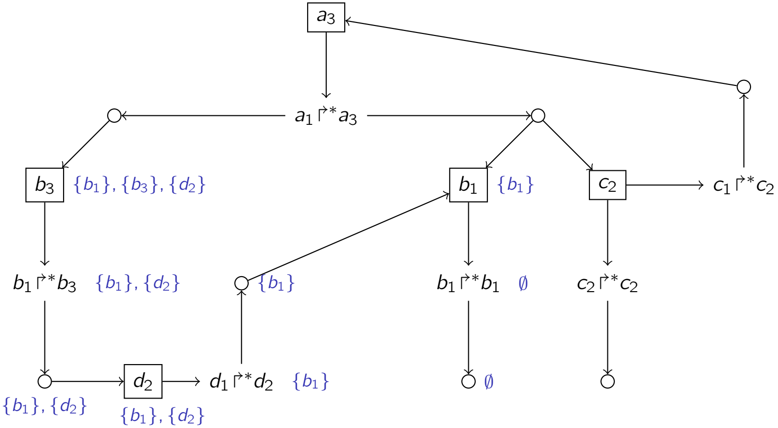


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

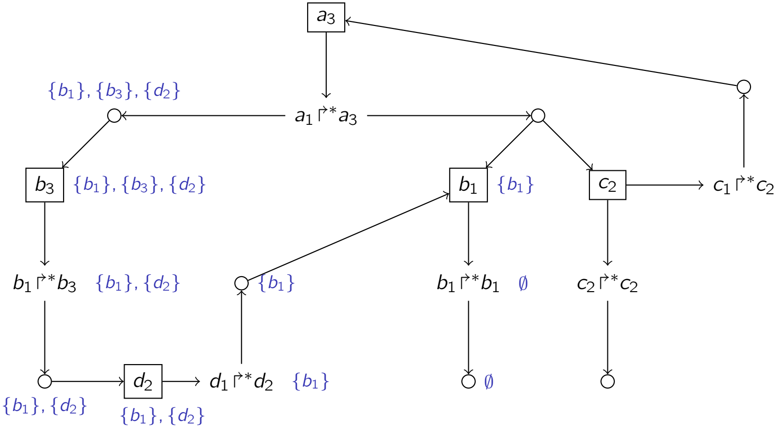


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

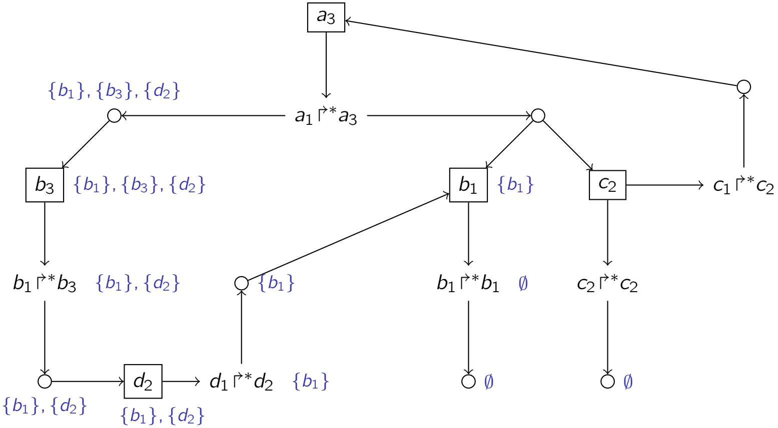


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

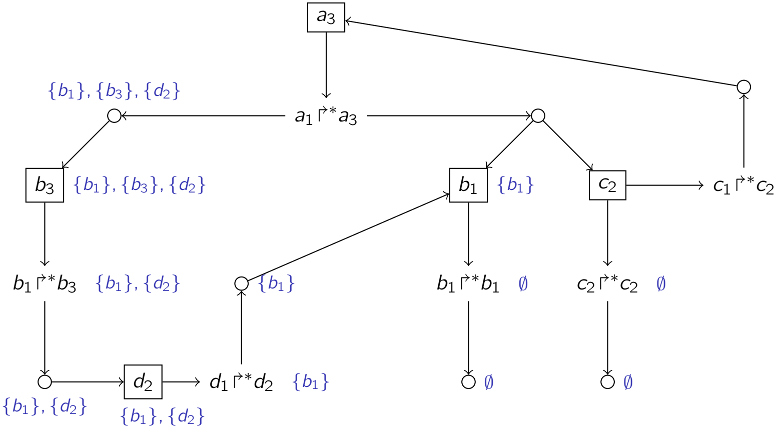


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

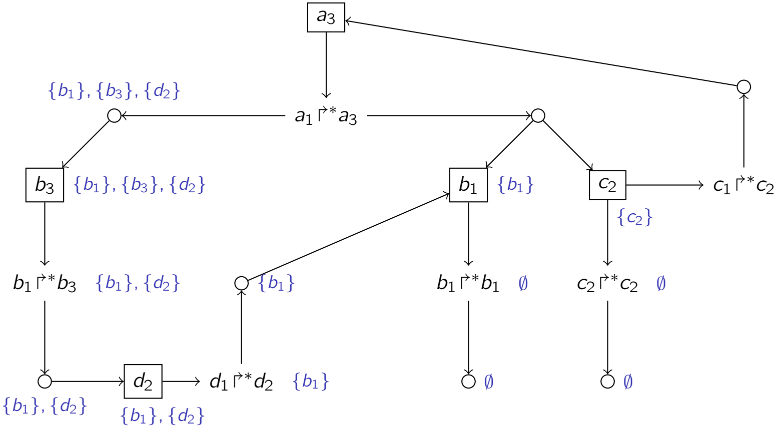


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

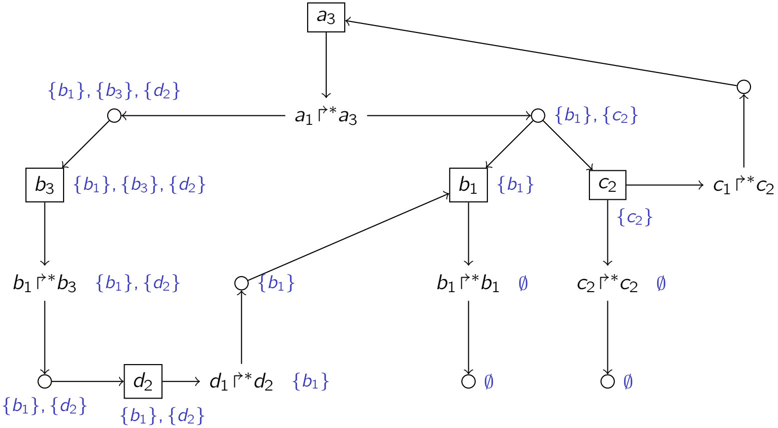


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

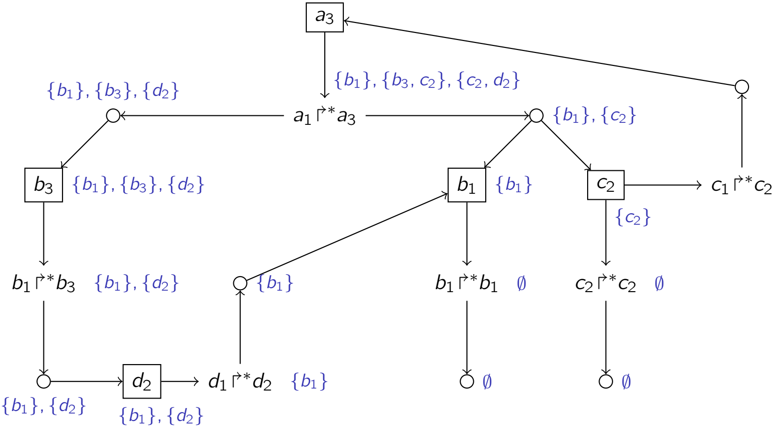


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

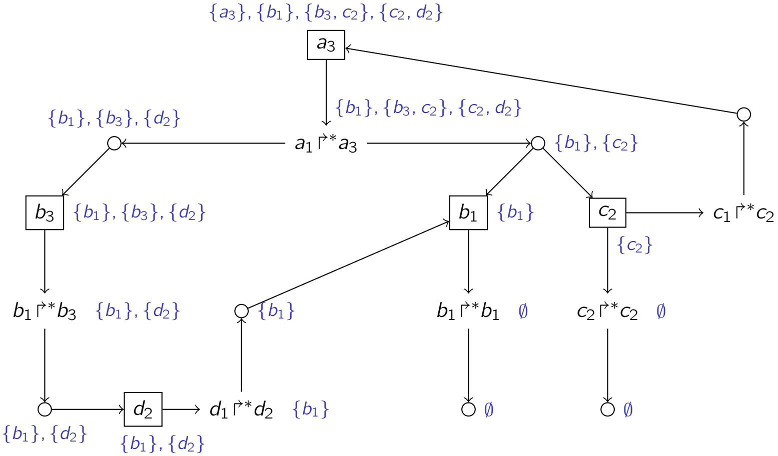


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

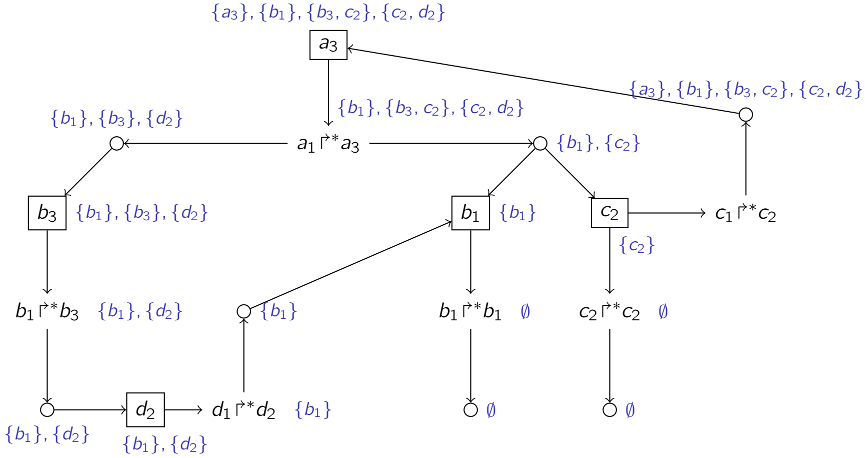


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.

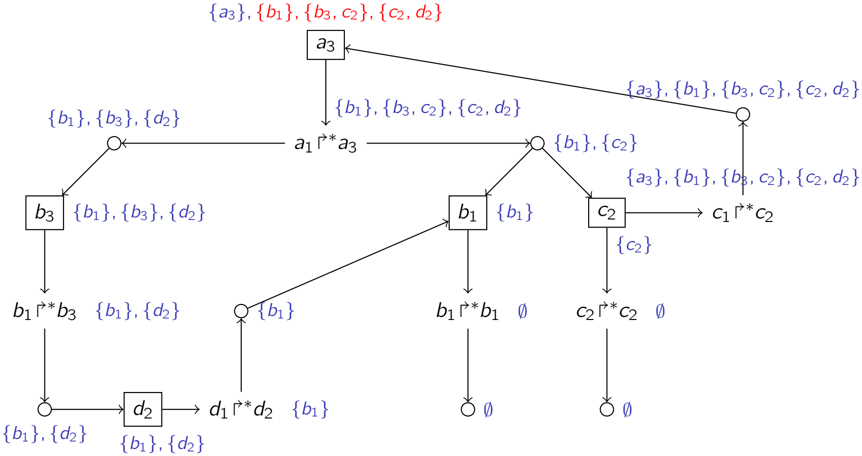


Cut Sets Under-approximation

Example

Sketch

- Follow the topological order of GLC.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.



Formal analysis of the whole PID

Pathway Interaction Database

- Inductions, inhibitions, transcriptional regulation, complex formations, ...
- More than 9000 interacting components.
- Large environment (3000 entry-points).

Graph of Local Causality

- From Process Hitting model (Boolean interpretation).
- (Independent) reachability of active SNAIL, active p15INK4b.
- 20 000 nodes, including 5600 processes (biological or cooperative).

Cut *N*-sets computed

N	Exec. time	$SNAIL_1$	$p15INK4b_1$
1	0.9s	1	1
2	1.6s	+6	+6
3	5.4s	+0	+92
4	39s	+30	+60
5	8.3m	+90	+80
6	2.6h	+930	+208

Outline

Introducing Time and Probabilities

Motivations

- Quantifying probability of reachability properties.
- Quantifying time to reach a given state/attractor.

Related work

- Formal frameworks: hybrid automata, continuous-time Markov chains, etc.
- Tools: model-checkers (PRISM, UPPAAL); numerous simulations techniques.

Continuous-time Markov Chains (CTMCs)

- Each transition receives a rate (speed).
- Rates control the probability of taking transitions
  $P(s \to s') = \frac{\text{rate}(s \to s')}{\sum_{s''} \text{rate}(s \to s'')}$
- Rates control the duration of transition
  $dt(s \to s') \sim exp(\sum_{s''} \text{rate}(s \to s''))$

Suited for population-counting models, but issues with qualitative models!

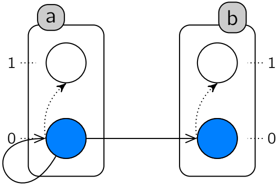
CTMCs for Qualitative Models

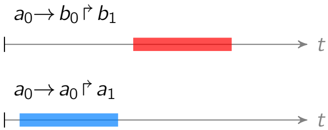
Issue

- Transition in qualitative models hides multiple reactions
- ⇒ some transitions may exhibit very low duration variance.
- But the rate entirely controls the variance (exponential distribution).

Proposed solution: Rate + Stochasticity absorption factor [Paulevé et al. IEEE TSE 11]

- Probability and duration can be independently tunned.
- Duration follows an Erlang distribution (non-Markovian setting).
- Allows to encode any confidence interval for the duration.
- Can still be converted to a regular CTMC at the end.





⇒ b_1 is reached at a very low probability.

Ongoing-work: priorities and time-scales

Motivations

- Rates are difficult to estimate.
- Focus on time-scales (qualitative) rather than precise durations.

Approach

- Process Hitting with Priorities.
- Some actions are always taken first, when possible.
- Adapt previous abstract interpretations.

First results, research directions

- Scalable reachability analysis (under-approximation) [Folschette et al. at CS2Bio'13].
- Take into account priorities for cut sets.

Summary

The Process Hitting framework

- Particular class of Asynchronous Automata Networks.
- Suited for modelling large interaction networks.
- Allows incomplete knowledge of cooperations (contrary to classical Boolean/multi-valued networks).

Formal analysis of dynamics

- Addressed in this talk: reachability and cut sets.
- Scalable thanks to abstract interpretation (potentially inconclusive).
- Graph of Local Causality provides comprehensive proofs.
- Over-approximations apply to any Automata Netwoks.

Link with other formalisms

- Any Boolean network can be encoded in Process Hitting.
- Inference of Boolean networks from Process Hitting [Folschette et al. at CMSB'12].
- Automatic encoding of interaction databases in progress.

Pint Software

http://loicpauleve.name/pint

Pint

- Textual language for Process Hitting
- Command line utilities for analysis.

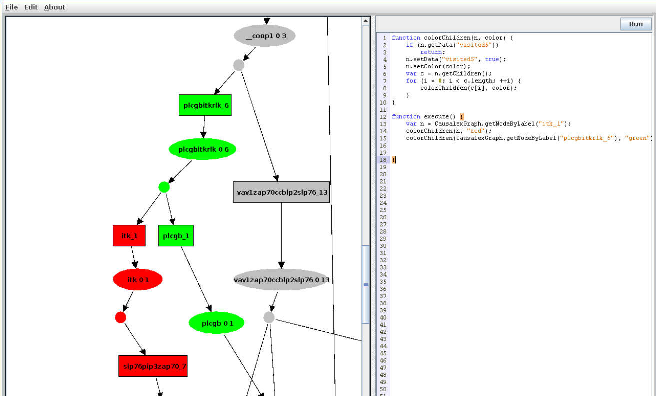
Main features

- Reachability analysis.
- Cut set analysis.
- Listing of fixed points (steady states).
- Non-markovian simulator for stochasticity absorption.
- Importation from various formats (CellNetAnalyser, SIF, ginML (partial), etc.)
- Exportation to various formats (PRISM, Biocham, Boolean networks, etc.)

Graphical interface in progress. . .

CausalEx Software

Coming soon. . .



Graphical interface for exploring Graphs of Local Causality

- Navigation
- Interactive scripting (javascript)
- Algorithm visualization

ACK: Fabienne Hirwa and Jean-Christophe Souplet from the software development team/LRI

Acknowledgement

IRCCyN, Nantes MeForBio

- Olivier Roux
- Morgan Magnin
- Carito Guziolowski
- Maxime Folschette
- Courtney Chancellor

IRISA, Rennes Dyliss

- Geoffroy Andrieux
- Nathalie Theret (INSERM)
- Anne Siegel

ETH Zürich BISON

- Heinz Koeppl

ANR BioTempo.