Causality Abstractions in Non-Deterministic Automata Networks

Loïc Paulevé

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

Joint work with *O. Roux*, *M. Magnin*, *M. Folschette* (IRCCyN), *G. Andrieux* (IRISA)

November 5, 2014 - Nice, France

Causality

(`event A` **and** `event B`) **or** `event C` cause `event D`

Causality

(`event A` **and** `event B`) **or** `event C` cause `event D`

`event A` **or** `event C` necessary for `event D`
`event B` **or** `event C` necessary for `event D`

`event A` **and** `event B` sufficient for `event D`
`event C` sufficient for `event D`

Causality

(`event A` **and** `event B`) **or** `event C` cause `event D`

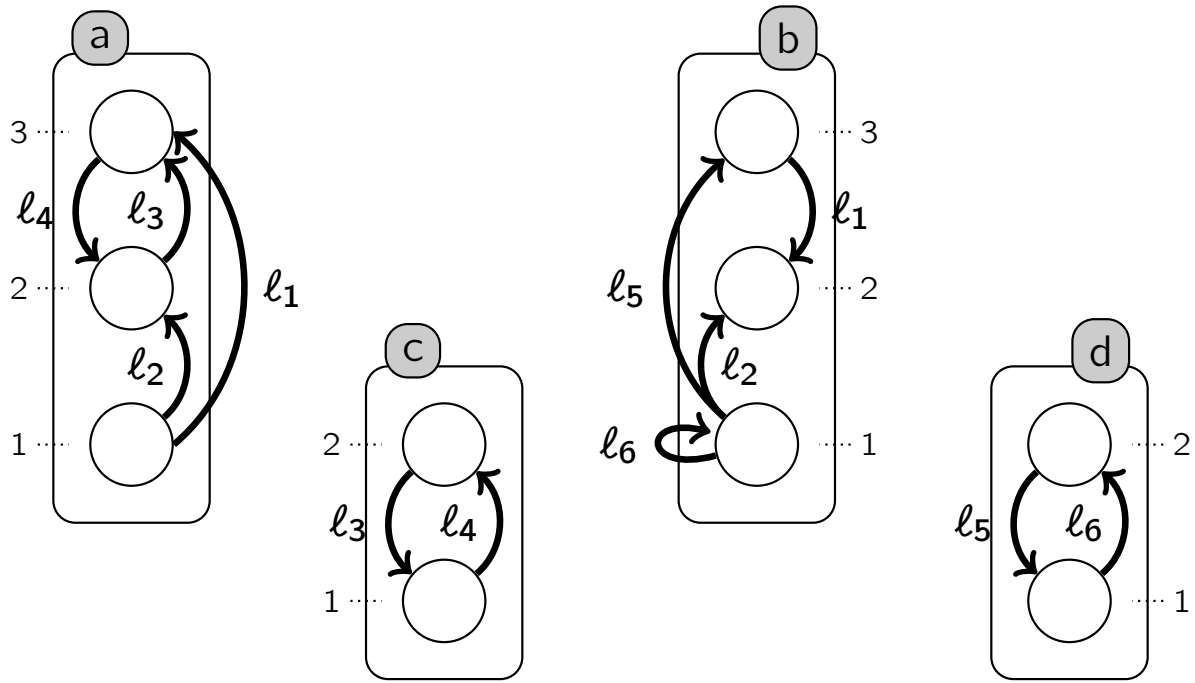
`event A` **or** `event C` necessary for `event D`
`event B` **or** `event C` necessary for `event D`

`event A` **and** `event B` sufficient for `event D`
`event C` sufficient for `event D`

Overview (1)

- Events are automaton state changes
- We focus on *local* causality, i.e. with very limited scope
- $\Rightarrow$ formal reasoning on local causalities to capture global dynamics.

Non-Deterministic Finite Automata Networks

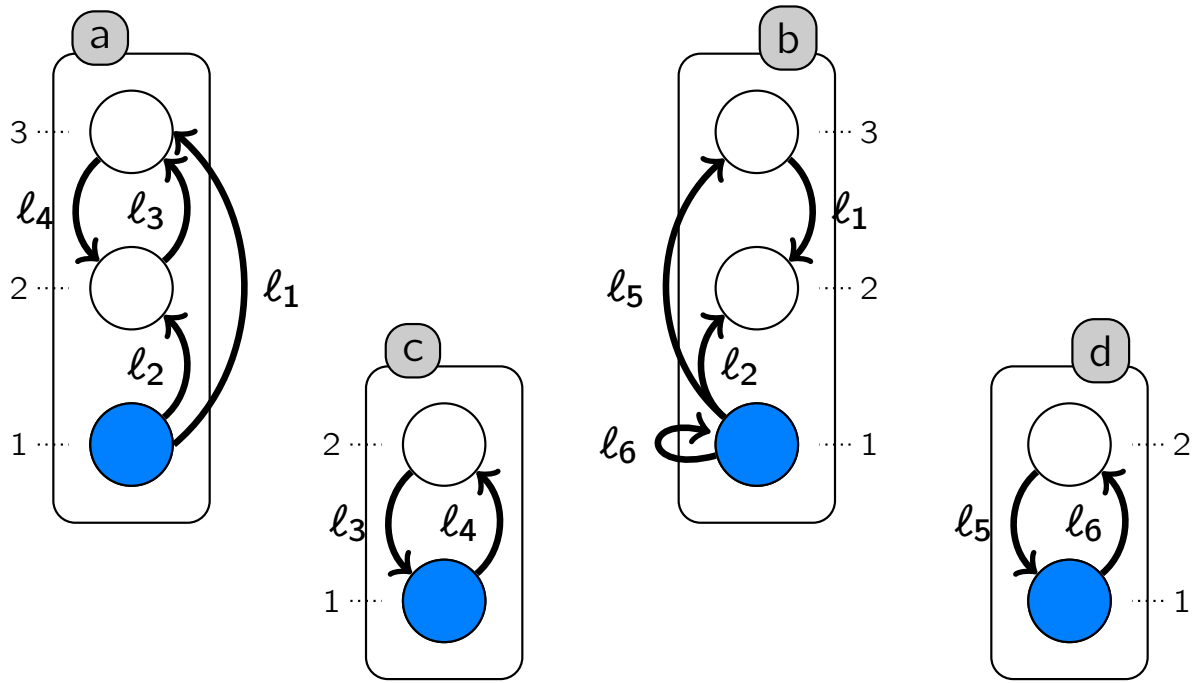


- transition ℓ pre-condition: ${}^\bullet\ell = \{a_i \mid a_i \xrightarrow{\ell} a_j\}$:

$$s \to s' \overset{\Delta}{\Leftrightarrow} \exists \ell \in \mathcal{L} : \forall a_i \in {}^\bullet\ell, s(a) = a_i \wedge \forall a_j \in \ell^\bullet, s'(a) = a_j$$
$$\wedge \forall b \in \Sigma, LS(b) \cap {}^\bullet\ell = \emptyset \Rightarrow s(b) = s'(b).$$

- (or 1-safe Petri nets with mutually exclusive places)

Non-Deterministic Finite Automata Networks

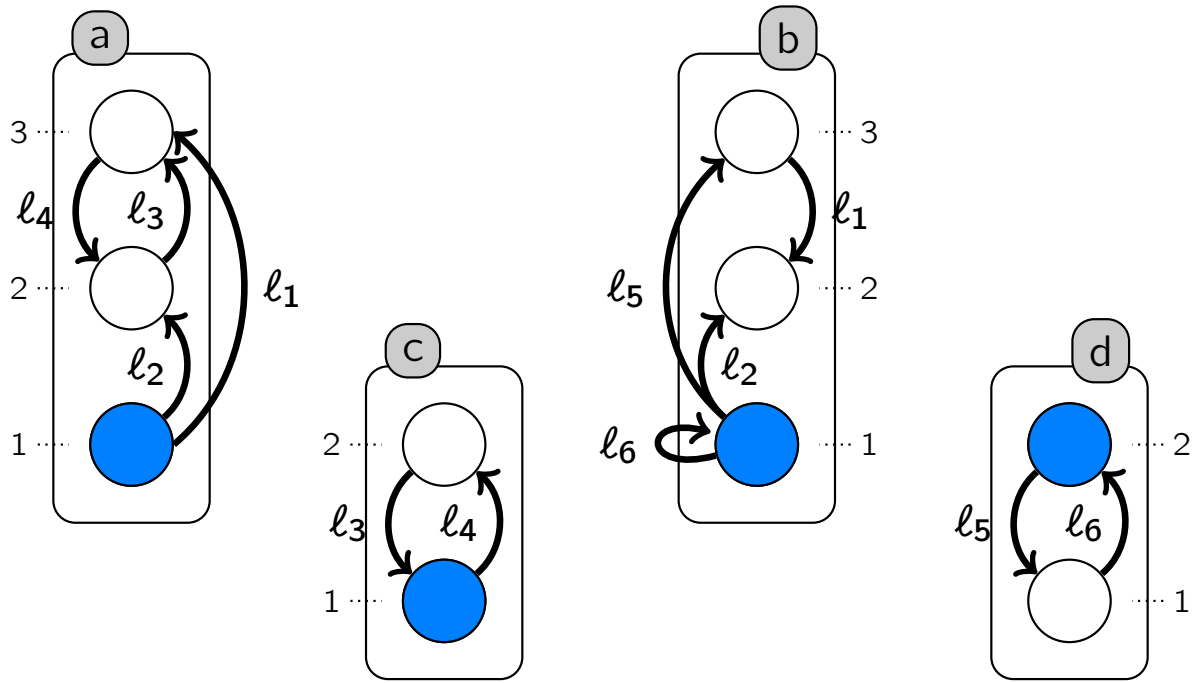


- transition ℓ pre-condition: ${}^\bullet\ell = \{a_i \mid a_i \xrightarrow{\ell} a_j\}$:

$$s \to s' \overset{\Delta}{\Leftrightarrow} \exists \ell \in \mathcal{L} : \forall a_i \in {}^\bullet\ell, s(a) = a_i \wedge \forall a_j \in \ell^\bullet, s'(a) = a_j$$
$$\wedge \forall b \in \Sigma, LS(b) \cap {}^\bullet\ell = \emptyset \Rightarrow s(b) = s'(b).$$

- (or 1-safe Petri nets with mutually exclusive places)

Non-Deterministic Finite Automata Networks

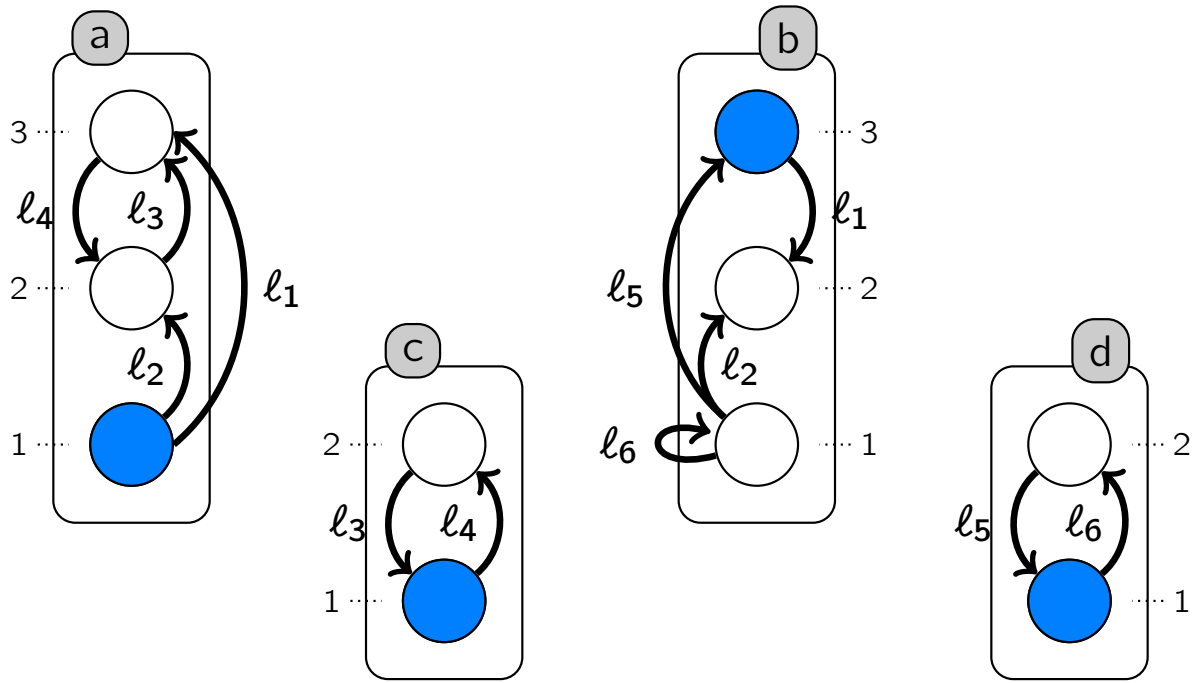


- transition ℓ pre-condition: ${}^\bullet\ell = \{a_i \mid a_i \xrightarrow{\ell} a_j\}$:

$$s \to s' \overset{\Delta}{\Leftrightarrow} \exists \ell \in \mathcal{L} : \forall a_i \in {}^\bullet\ell, s(a) = a_i \wedge \forall a_j \in \ell^\bullet, s'(a) = a_j$$
$$\wedge \forall b \in \Sigma, LS(b) \cap {}^\bullet\ell = \emptyset \Rightarrow s(b) = s'(b).$$

- (or 1-safe Petri nets with mutually exclusive places)

Non-Deterministic Finite Automata Networks

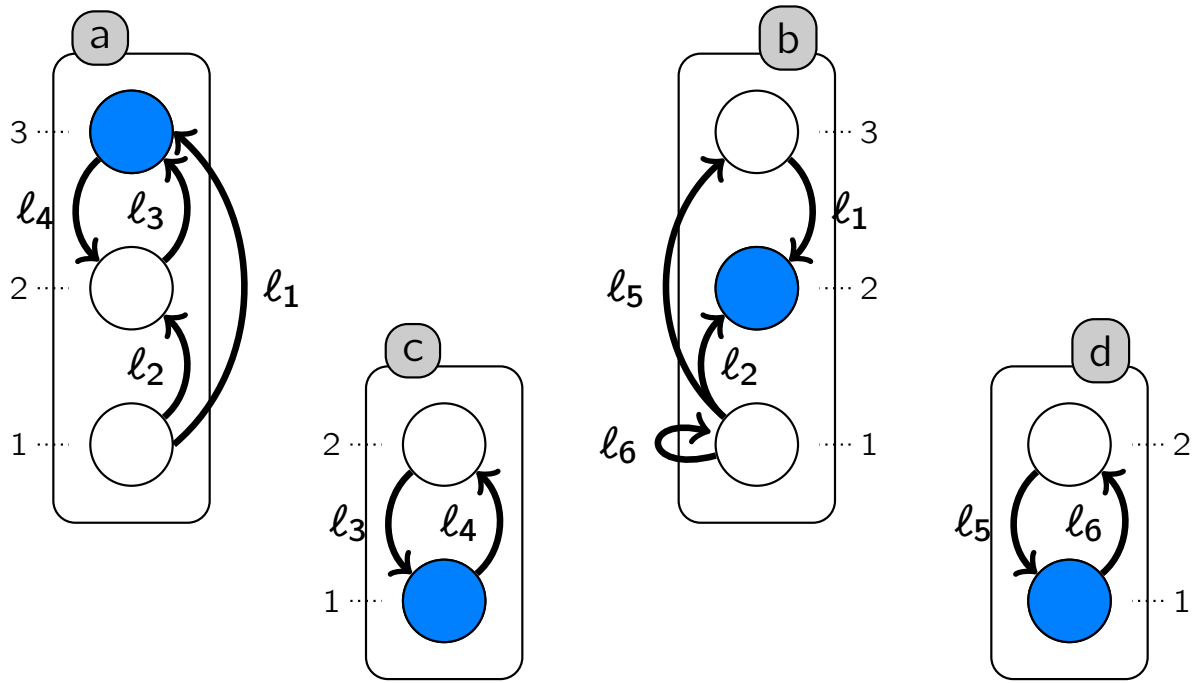


- transition ℓ pre-condition: ${}^\bullet\ell = \{a_i \mid a_i \xrightarrow{\ell} a_j\}$:

$$s \to s' \overset{\Delta}{\Leftrightarrow} \exists \ell \in \mathcal{L} : \forall a_i \in {}^\bullet\ell, s(a) = a_i \wedge \forall a_j \in \ell^\bullet, s'(a) = a_j$$
$$\wedge \forall b \in \Sigma, LS(b) \cap {}^\bullet\ell = \emptyset \Rightarrow s(b) = s'(b).$$

- (or 1-safe Petri nets with mutually exclusive places)

Non-Deterministic Finite Automata Networks

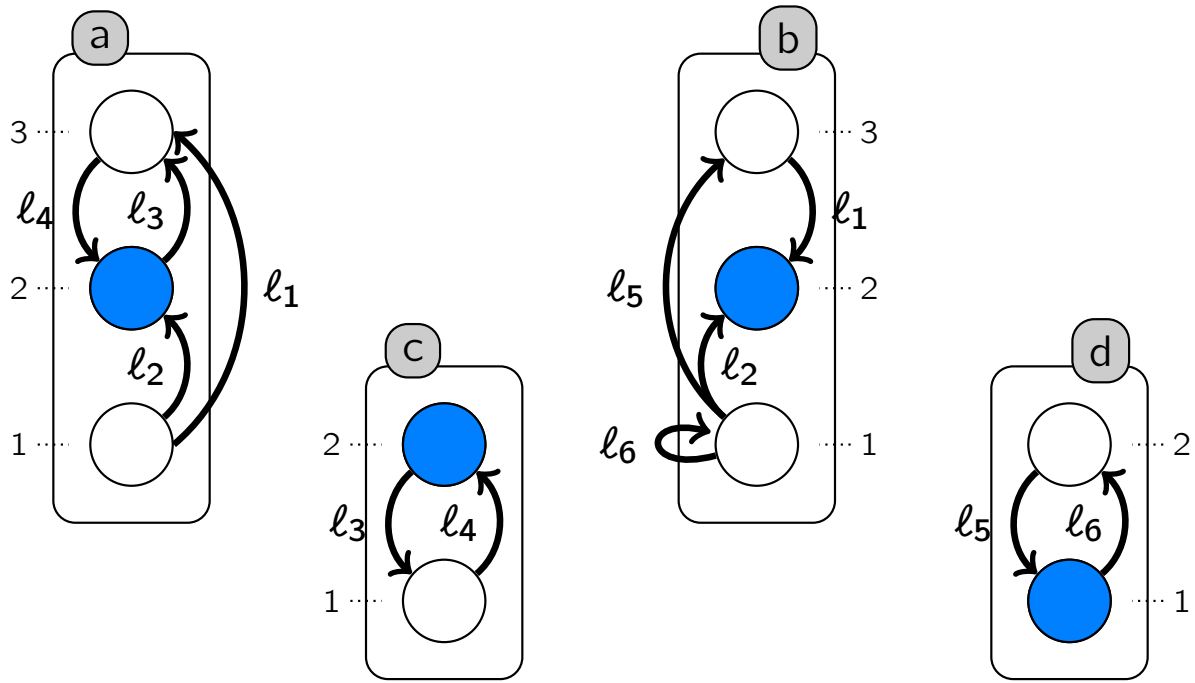


- transition ℓ pre-condition: ${}^\bullet\ell = \{a_i \mid a_i \xrightarrow{\ell} a_j\}$:

$$s \to s' \overset{\Delta}{\Leftrightarrow} \exists \ell \in \mathcal{L} : \forall a_i \in {}^\bullet\ell, s(a) = a_i \wedge \forall a_j \in \ell^\bullet, s'(a) = a_j$$
$$\wedge \forall b \in \Sigma, LS(b) \cap {}^\bullet\ell = \emptyset \Rightarrow s(b) = s'(b).$$

- (or 1-safe Petri nets with mutually exclusive places)

Non-Deterministic Finite Automata Networks



- transition ℓ pre-condition: ${}^\bullet\ell = \{a_i \mid a_i \xrightarrow{\ell} a_j\}$:

$$s \to s' \overset{\Delta}{\Leftrightarrow} \exists \ell \in \mathcal{L} : \forall a_i \in {}^\bullet\ell, s(a) = a_i \wedge \forall a_j \in \ell^\bullet, s'(a) = a_j$$
$$\wedge \forall b \in \Sigma, LS(b) \cap {}^\bullet\ell = \emptyset \Rightarrow s(b) = s'(b).$$

- (or 1-safe Petri nets with mutually exclusive places)

Indeterministic Finite Automata Networks

Comments

- Transition-centered specification
- Can model indeterministic discrete function:
  $f^a(x) = \begin{cases} 1 & \text{if } x[b] \geq 1 \vee x[c] \geq 1 \\ 0 & \text{if } x[b] = 0 \vee x[c] = 0 \end{cases}$
- Can model any discrete network async/sync update.

Specializations (sub-classes)

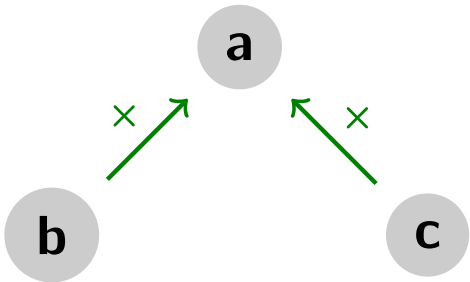
Asynchronous Automata Network

- Only one automaton is updated at each transition
  $(\forall \ell, \#\{a_j \mid a_i \xrightarrow{\ell} a_j, j \neq i\} = 1)$

AAN + binary pre-condition (a.k.a. Process Hitting)

- Asynchronous Automata Network
- + any transition concerns at most two automata
  $(\forall \ell, \#^\bullet \ell \leq 2)$.

Interaction Networks with Automata Networks

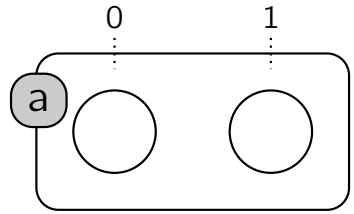


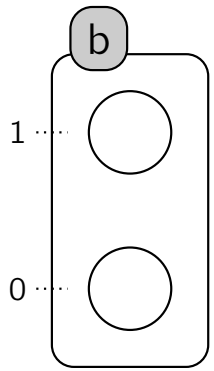
1. $f^a(x) = x[b] \wedge x[c]$

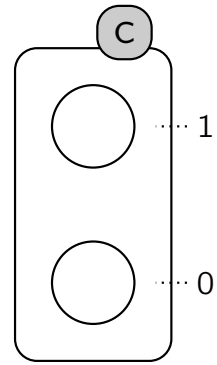
transitions:

$a_0 \to a_1$: $b_1 \wedge c_1$

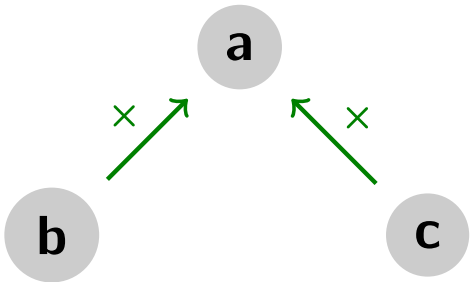
$a_1 \to a_0$: $b_0 \vee c_0$







Interaction Networks with Automata Networks

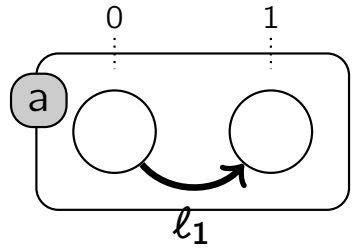


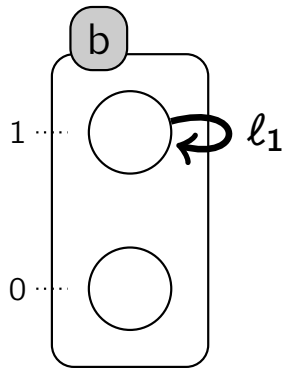
1. $f^a(x) = x[b] \wedge x[c]$

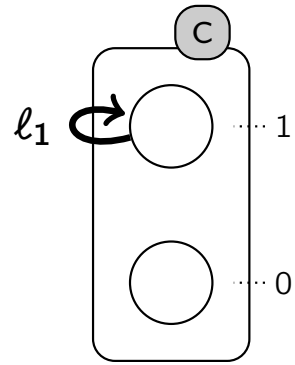
transitions:

$a_0 \to a_1$: $b_1 \wedge c_1$

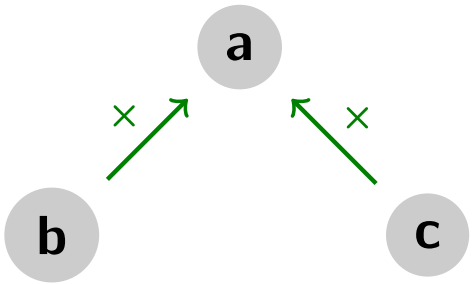
$a_1 \to a_0$: $b_0 \vee c_0$







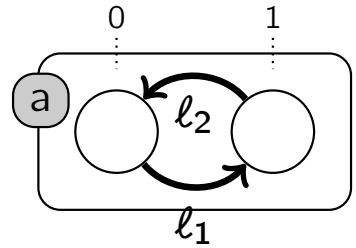
Interaction Networks with Automata Networks

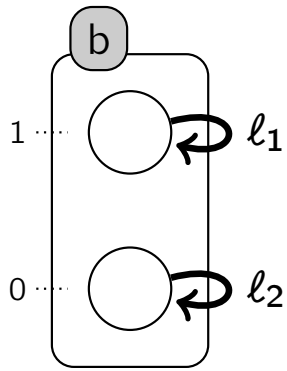


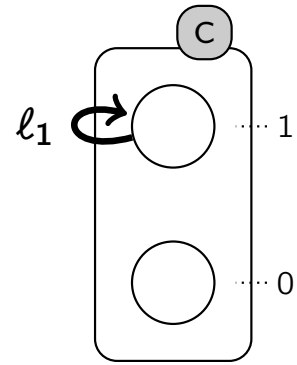
1. $f^a(x) = x[b] \wedge x[c]$
transitions:

$a_0 \to a_1$: $b_1 \wedge c_1$

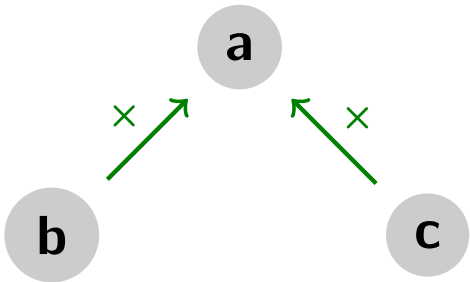
$a_1 \to a_0$: $b_0 \vee c_0$







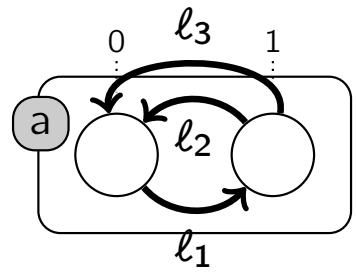
Interaction Networks with Automata Networks

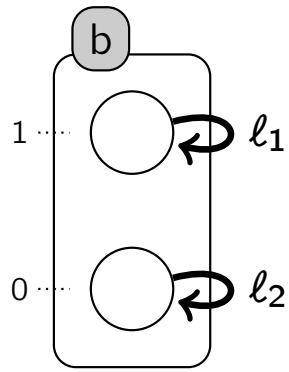


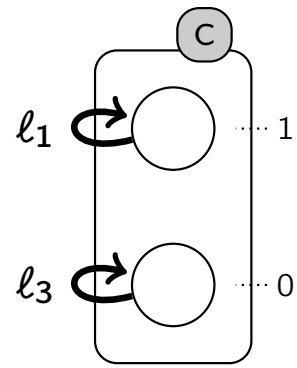
1. $f^a(x) = x[b] \wedge x[c]$
transitions:

$a_0 \to a_1$: $b_1 \wedge c_1$

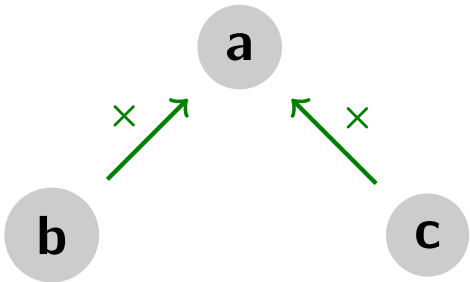
$a_1 \to a_0$: $b_0 \vee c_0$







Interaction Networks with Automata Networks



1. $f^a(x) = x[b] \wedge x[c]$
transitions:

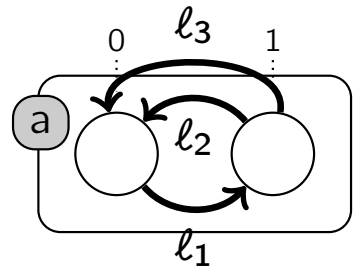
$a_0 \to a_1$: $b_1 \wedge c_1$

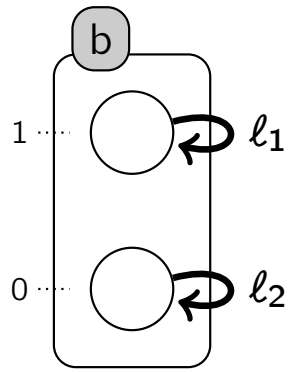
$a_1 \to a_0$: $b_0 \vee c_0$

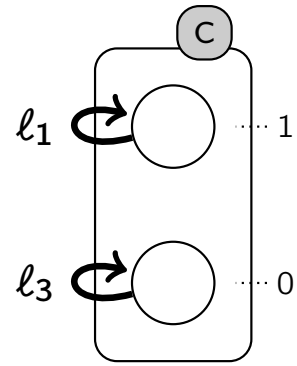
2. Non-deterministic f^a
transitions:

$a_0 \to a_1$: $b_1 \vee c_1$

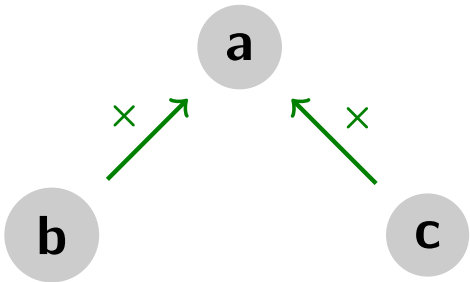
$a_1 \to a_0$: $b_0 \vee c_0$







Interaction Networks with Automata Networks



1. $f^a(x) = x[b] \wedge x[c]$ transitions:

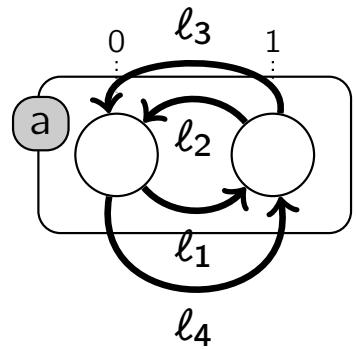
$a_0 \to a_1$: $b_1 \wedge c_1$

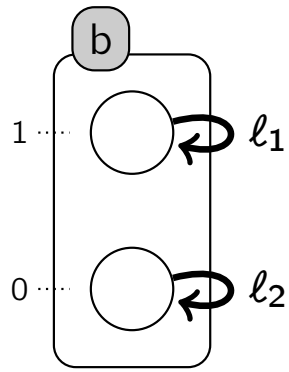
$a_1 \to a_0$: $b_0 \vee c_0$

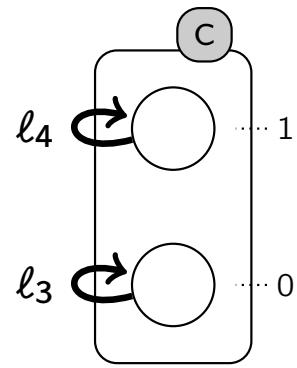
2. Non-deterministic f^a transitions:

$a_0 \to a_1$: $b_1 \vee c_1$

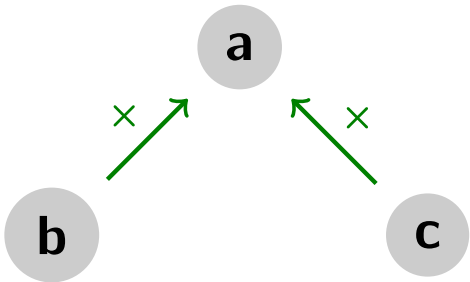
$a_1 \to a_0$: $b_0 \vee c_0$







Interaction Networks with Automata Networks



1. $f^a(x) = x[b] \wedge x[c]$ transitions:

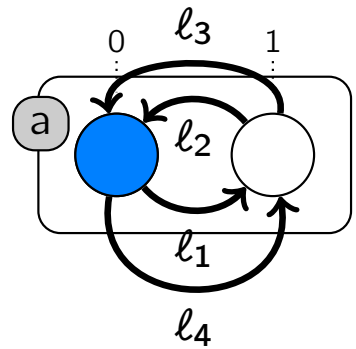
$a_0 \to a_1$: $b_1 \wedge c_1$

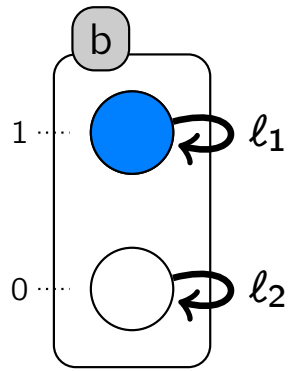
$a_1 \to a_0$: $b_0 \vee c_0$

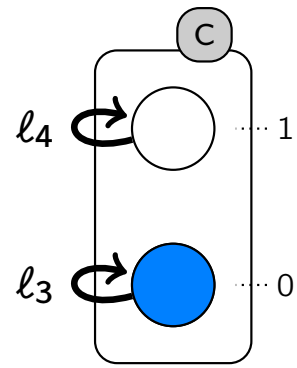
2. Non-deterministic f^a transitions:

$a_0 \to a_1$: $b_1 \vee c_1$

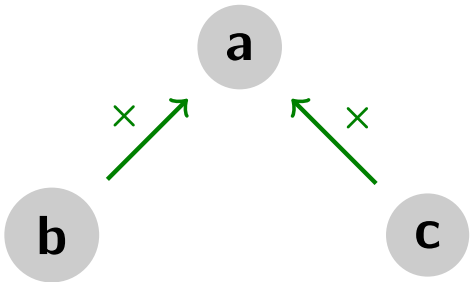
$a_1 \to a_0$: $b_0 \vee c_0$







Interaction Networks with Automata Networks



1. $f^a(x) = x[b] \wedge x[c]$ transitions:

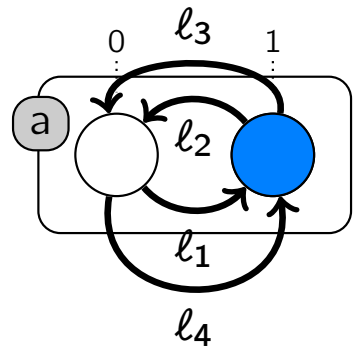
$a_0 \to a_1$: $b_1 \wedge c_1$

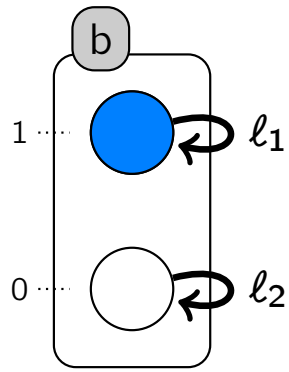
$a_1 \to a_0$: $b_0 \vee c_0$

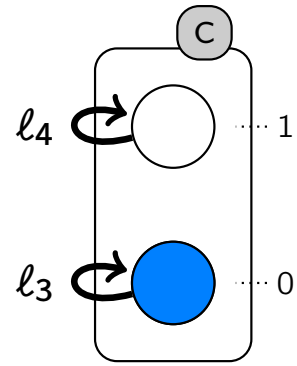
2. Non-deterministic f^a transitions:

$a_0 \to a_1$: $b_1 \vee c_1$

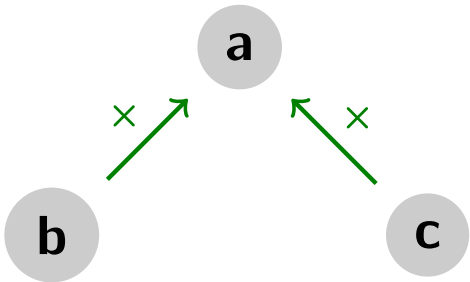
$a_1 \to a_0$: $b_0 \vee c_0$







Interaction Networks with Automata Networks



1. $f^a(x) = x[b] \wedge x[c]$ transitions:

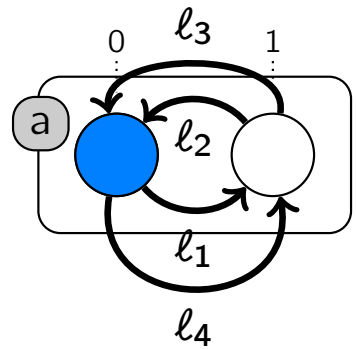
$a_0 \to a_1$: $b_1 \wedge c_1$

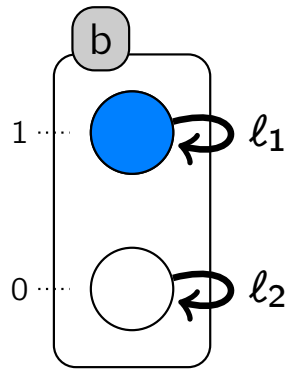
$a_1 \to a_0$: $b_0 \vee c_0$

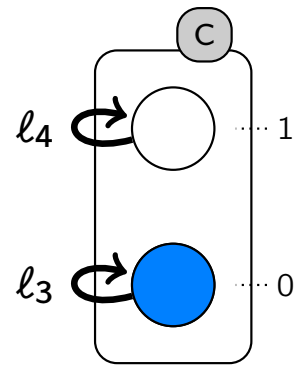
2. Non-deterministic f^a transitions:

$a_0 \to a_1$: $b_1 \vee c_1$

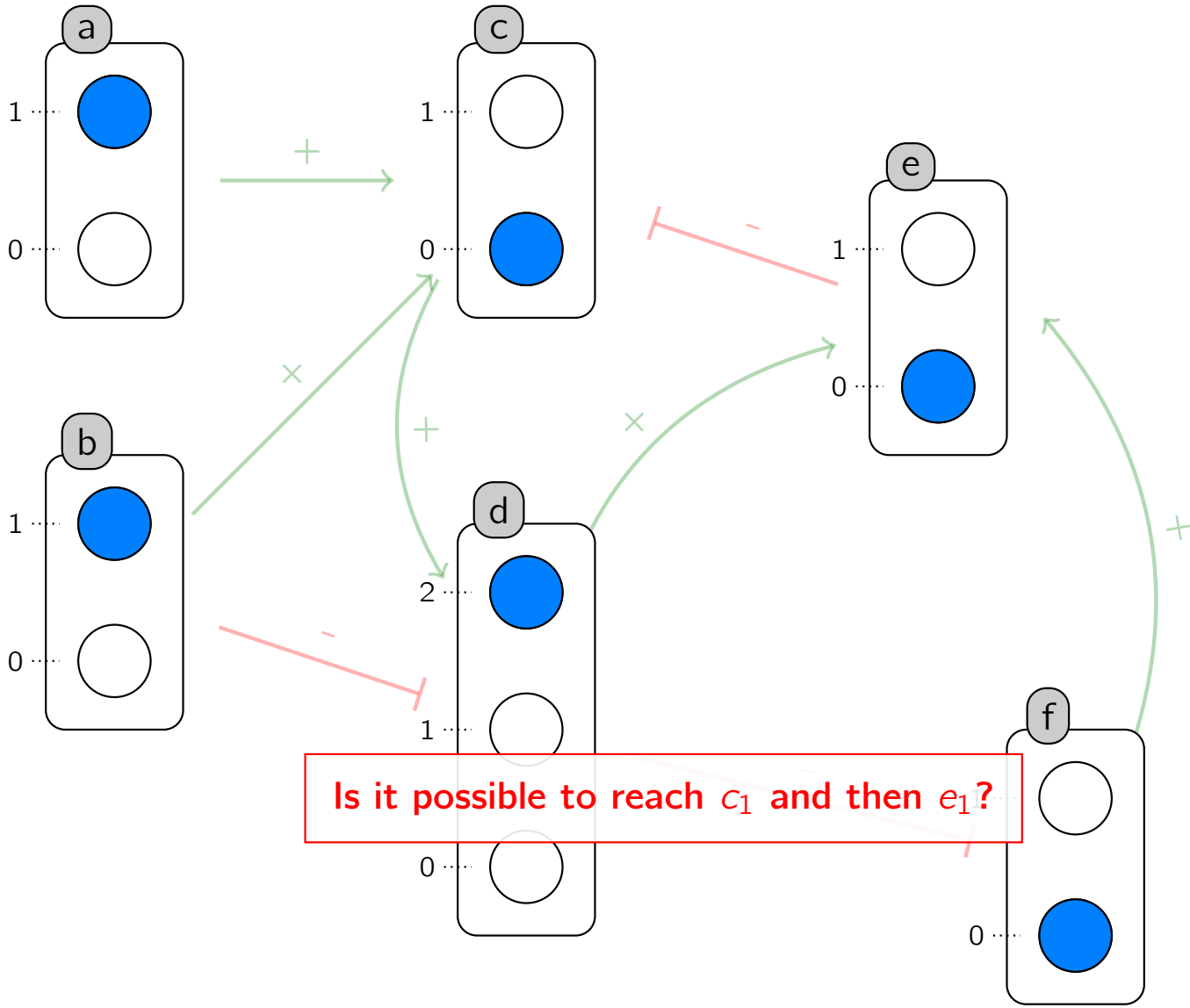
$a_1 \to a_0$: $b_0 \vee c_0$



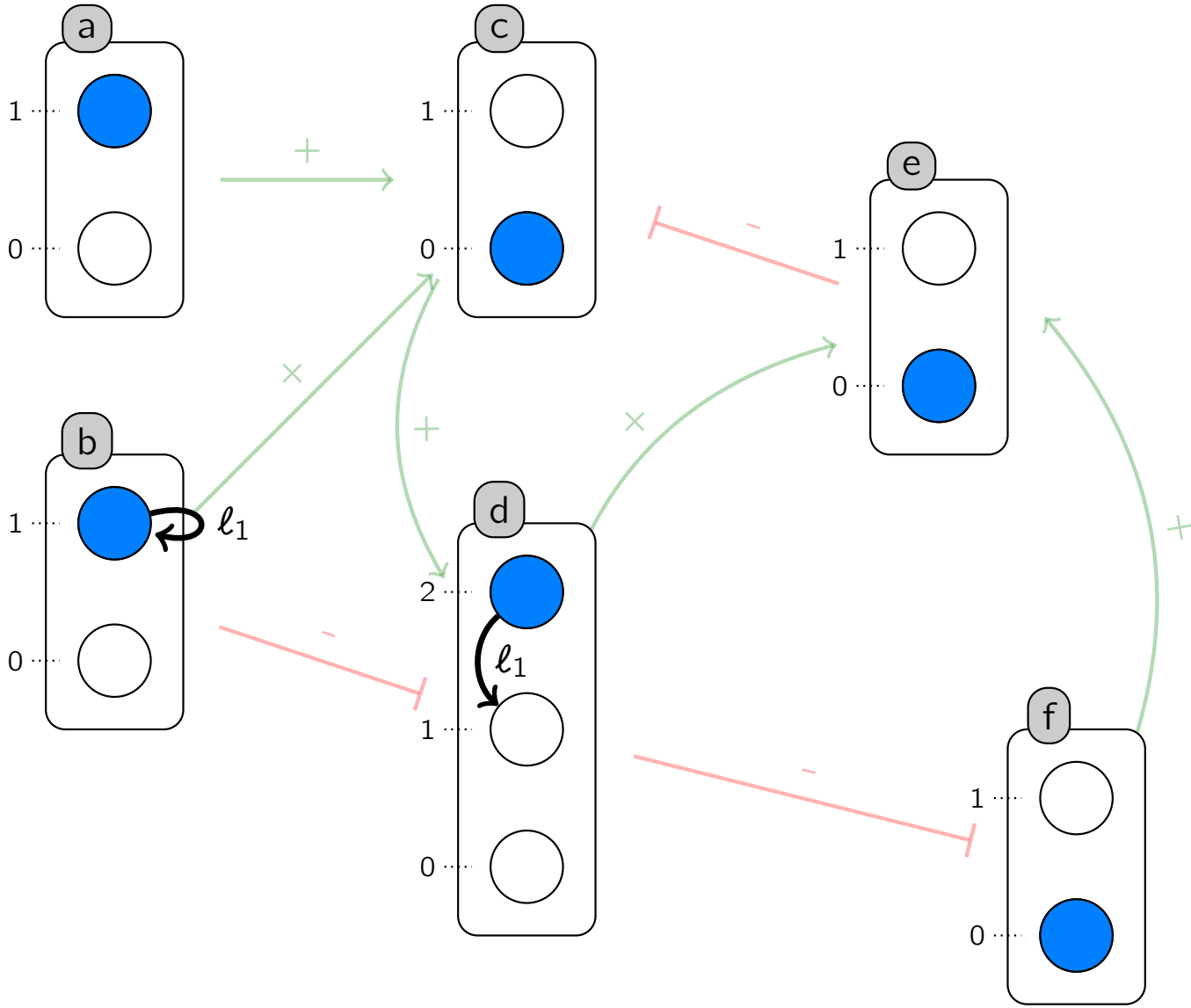




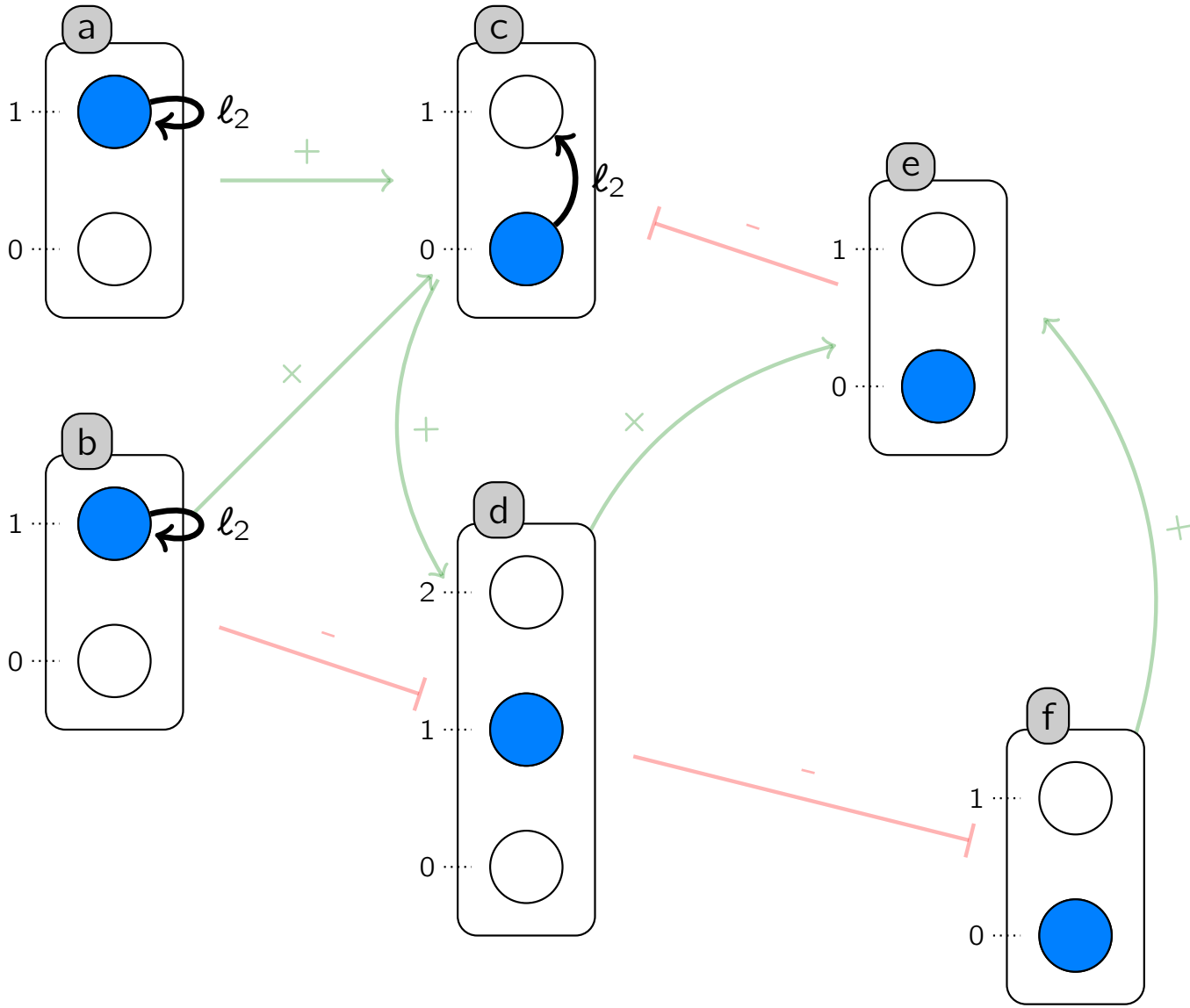
Reachability



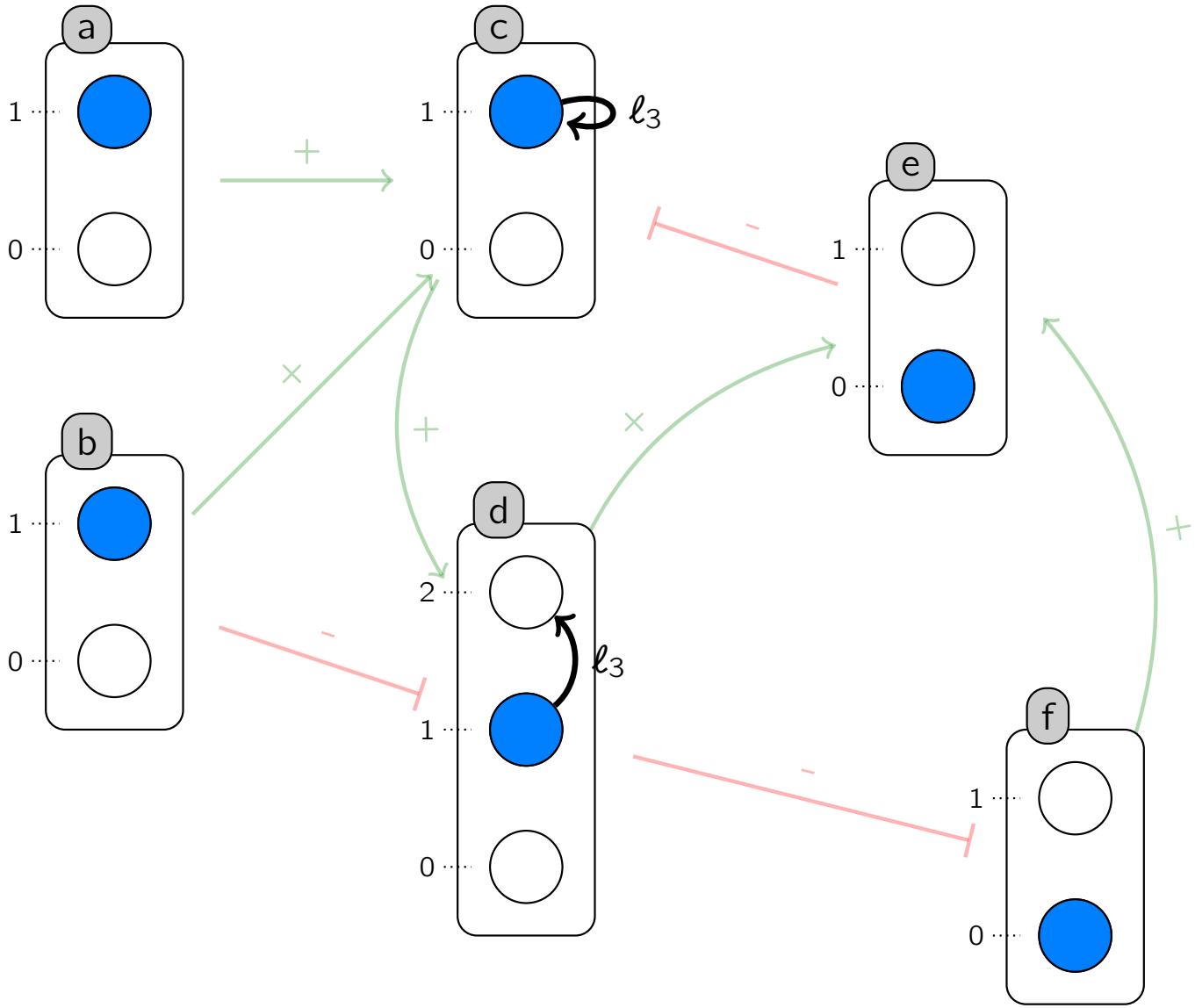
Reachability



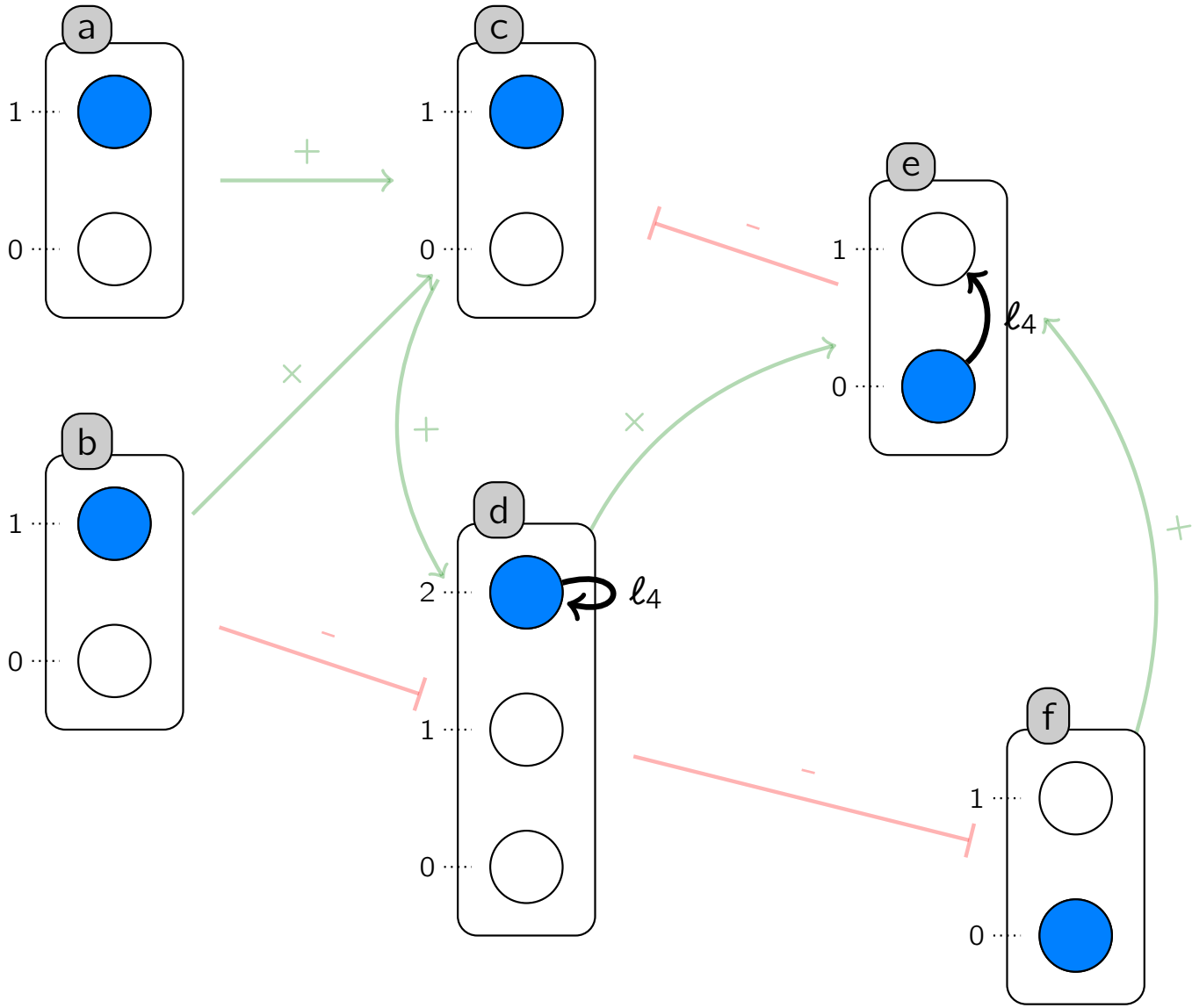
Reachability



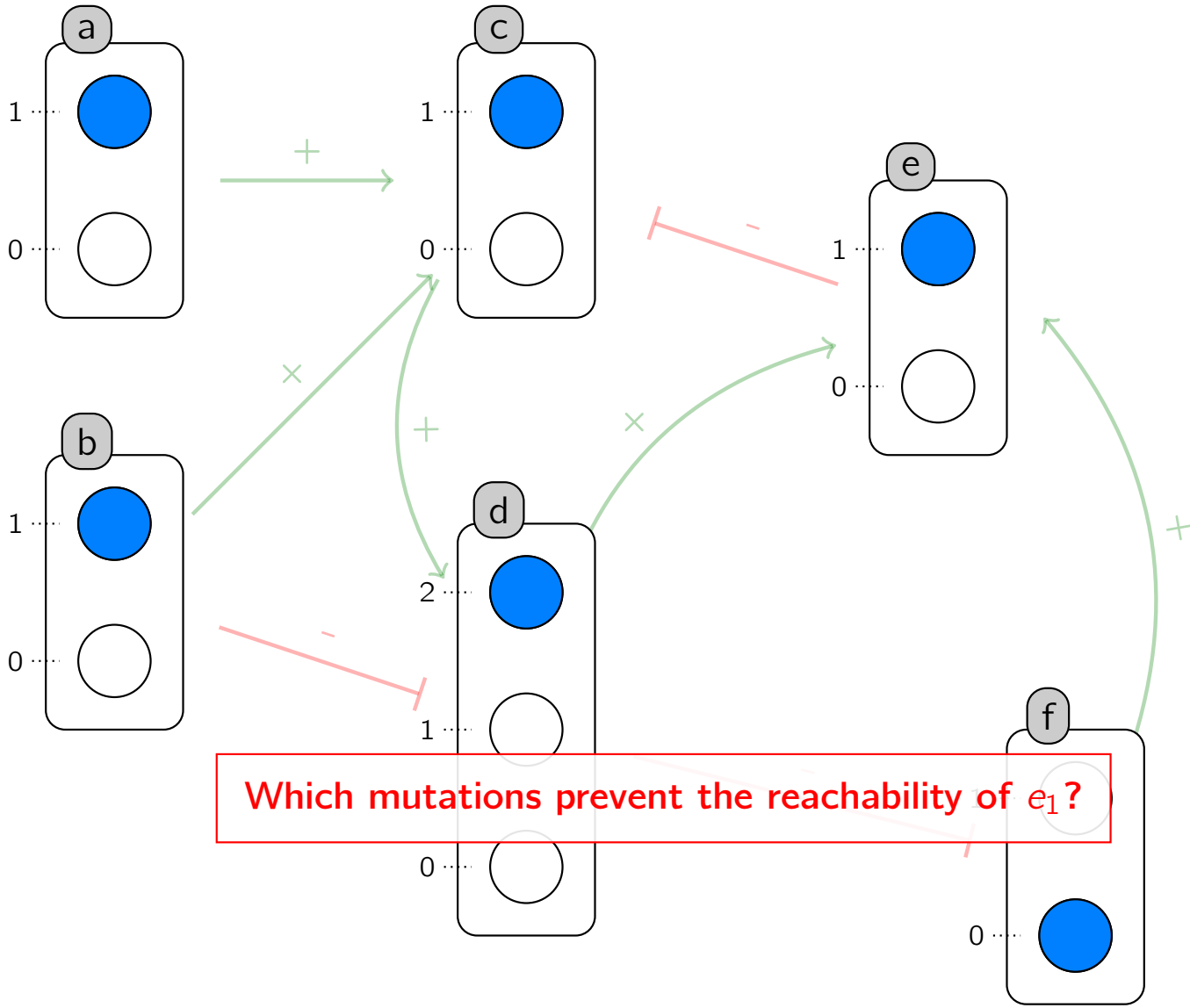
Reachability



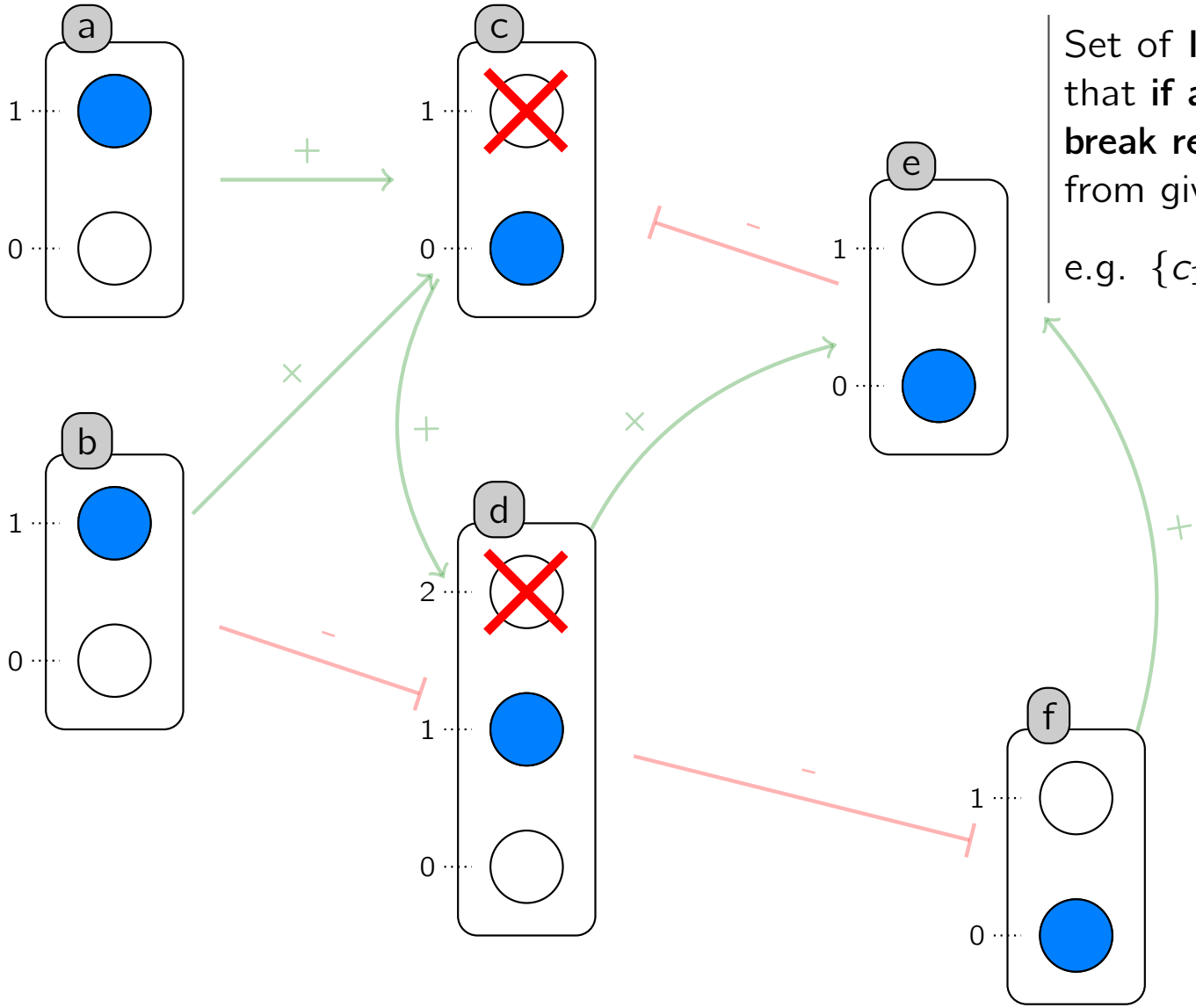
Reachability



Reachability



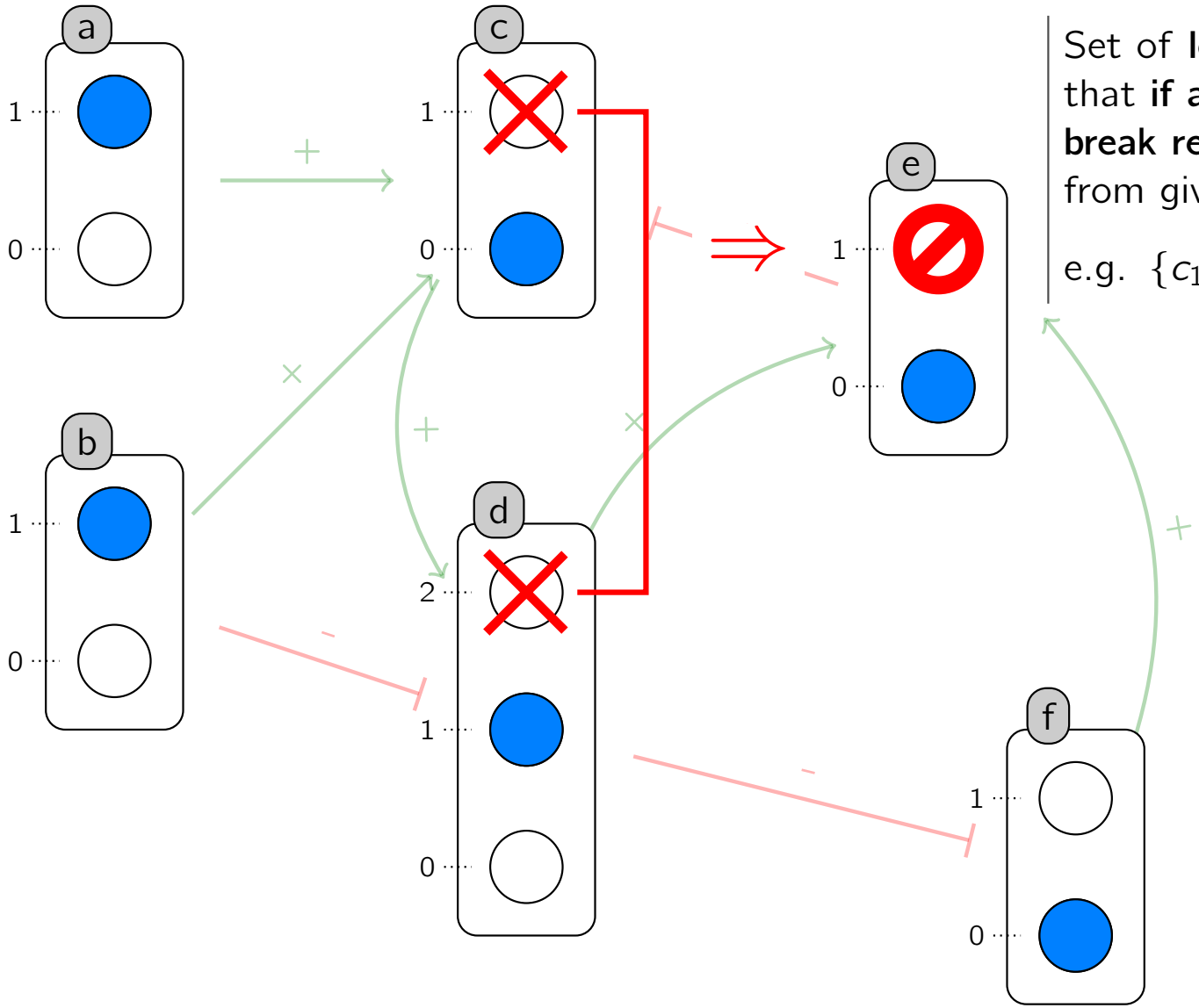
Cut Sets for Reachability



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

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

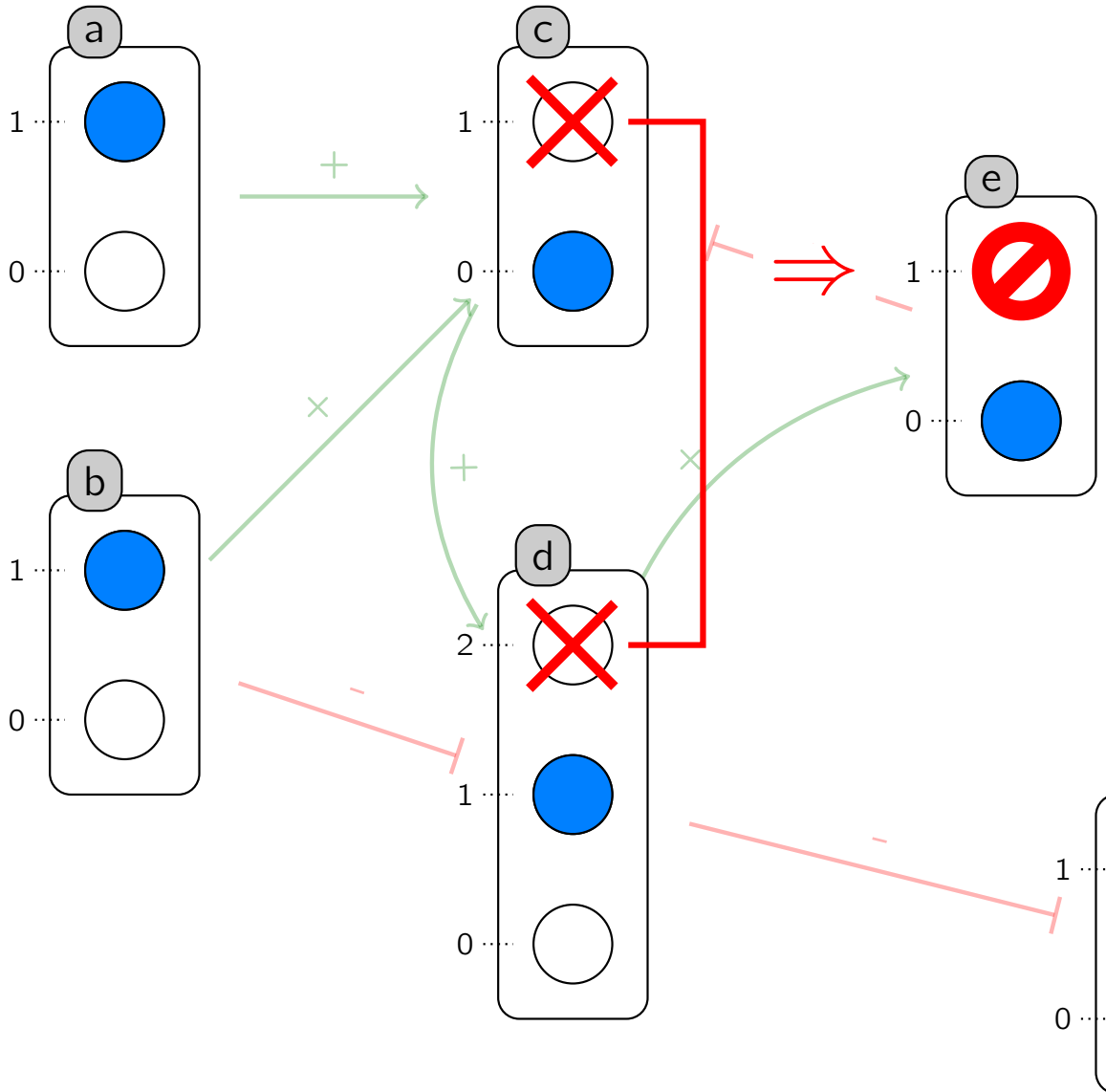
Cut Sets for Reachability



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

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

Cut Sets for Reachability



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

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

Applications

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

Motivation

Scalable analysis of dynamics of automata networks

Motivation

Scalable analysis of dynamics of automata networks

Standard Model-Checking..

- is not scalable: `PSPACE-complete`.
- urban legend: *but it is easy with symbolic model-checking* (same complexity).
- Provide no comprehensive proof of the result.

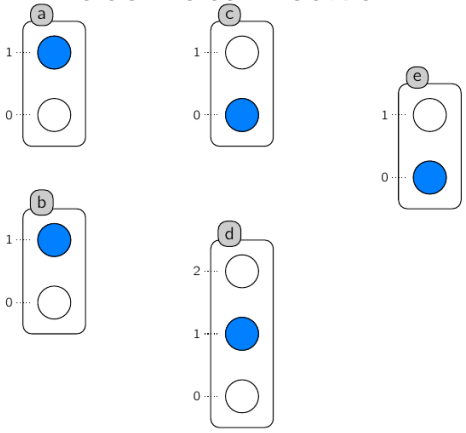
What is hard? branching (due to concurrency).

Contributions

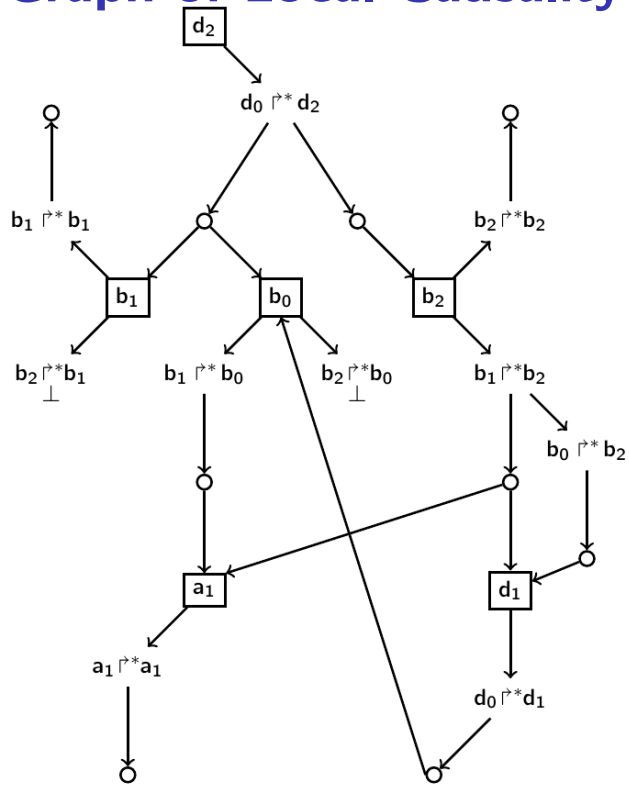
- Compact representation of causality
  (exploits local causality + concurrency).
- Highly scalable reachability/cut-sets analysis (but incomplete)
- Comprehensive proofs.

Overview (2)

Automata network



Graph of Local Causality



Over-approximation of reachability
Under-approximation of reachability
Under-approximation of cut sets

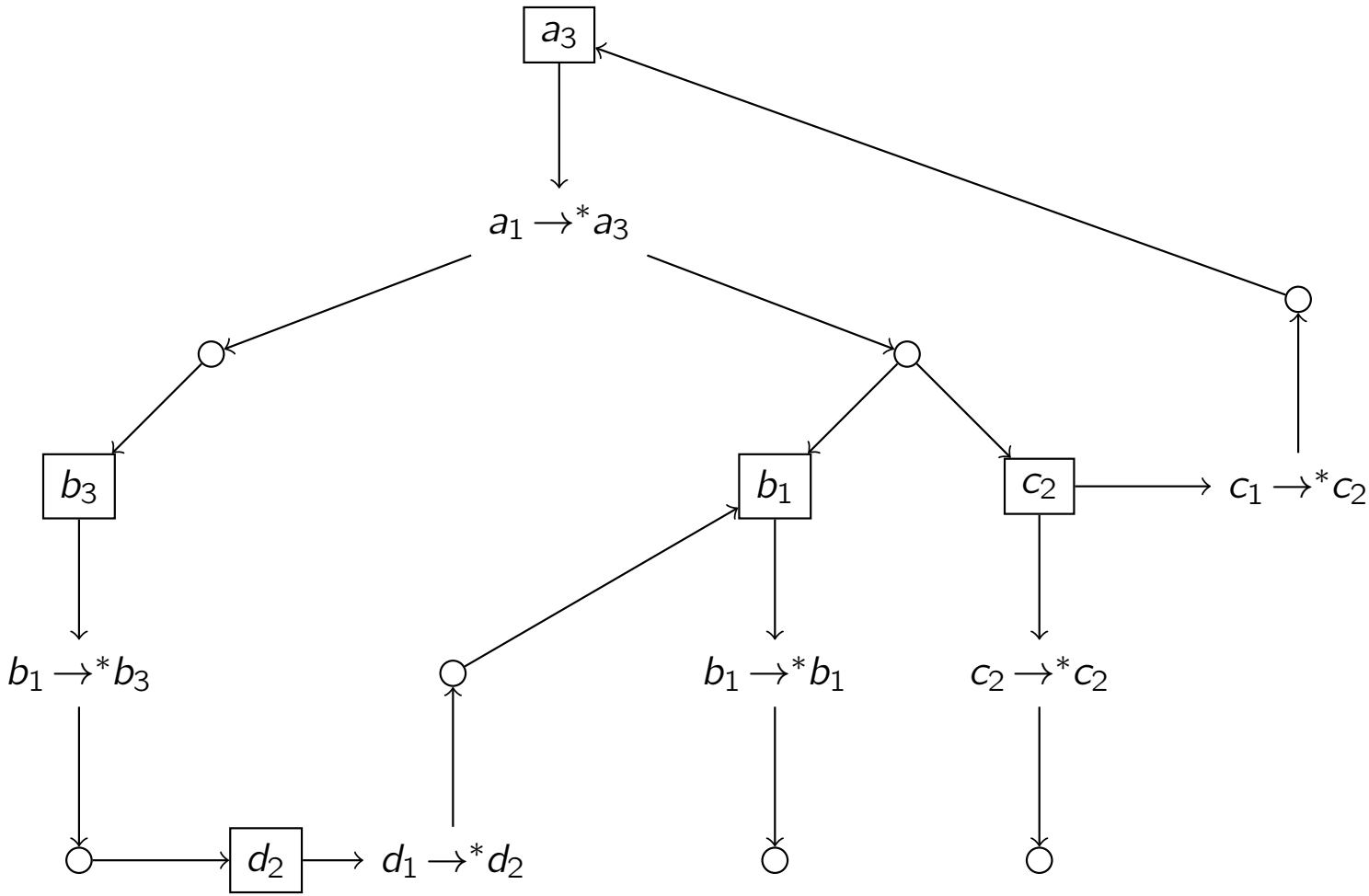
[Paulevé et al. in Math. Struct. in Comp. Sci. 2012]
[Paulevé et al. at CAV 2013]

Outline

Outline

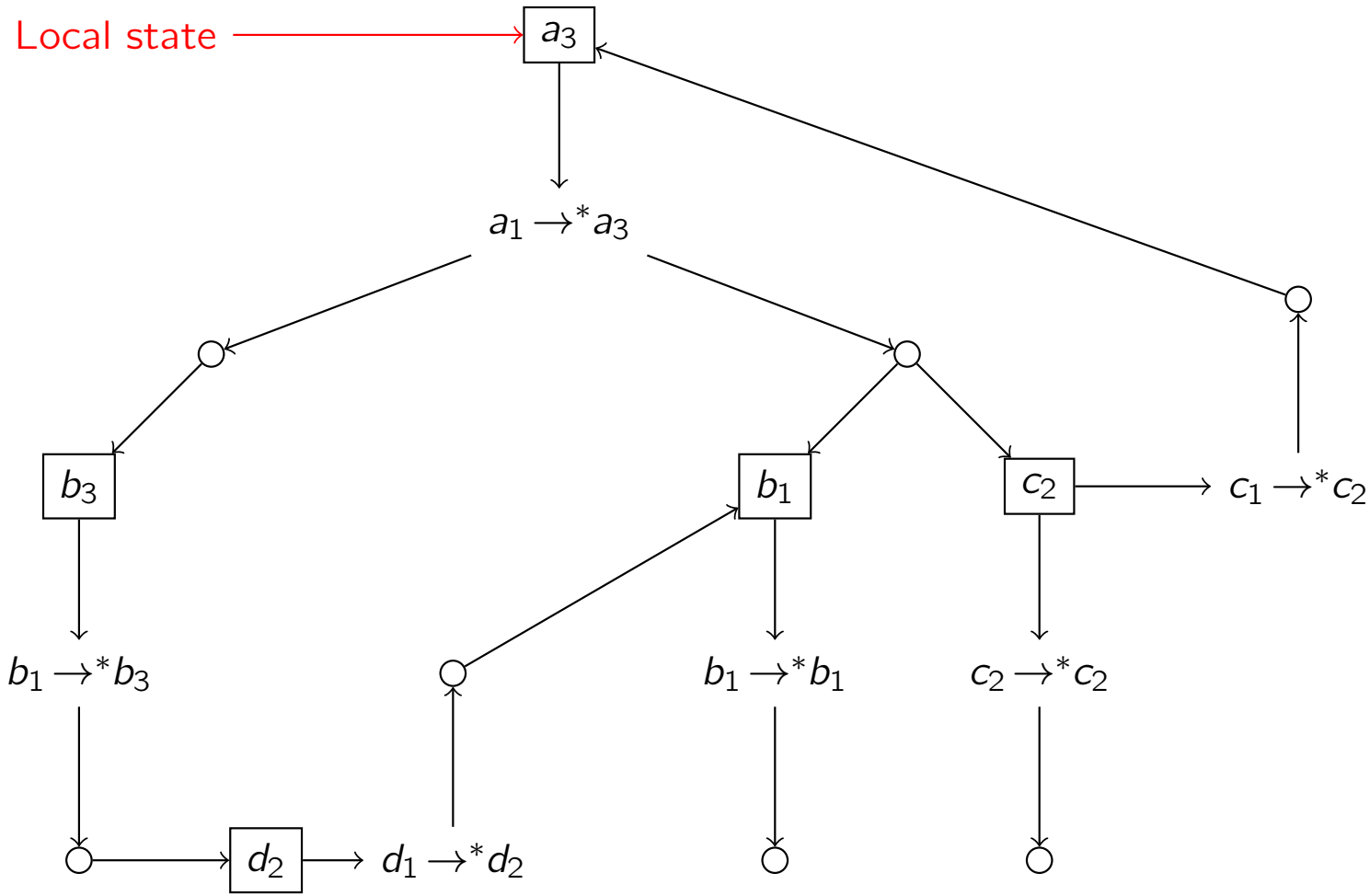
Local Causality Graph

- Causality of a_3.
- Initial context $\varsigma = \{a \mapsto \{1\}; b \mapsto \{1\}; c \mapsto \{1, 2\}; d \mapsto \{2\}\}$.



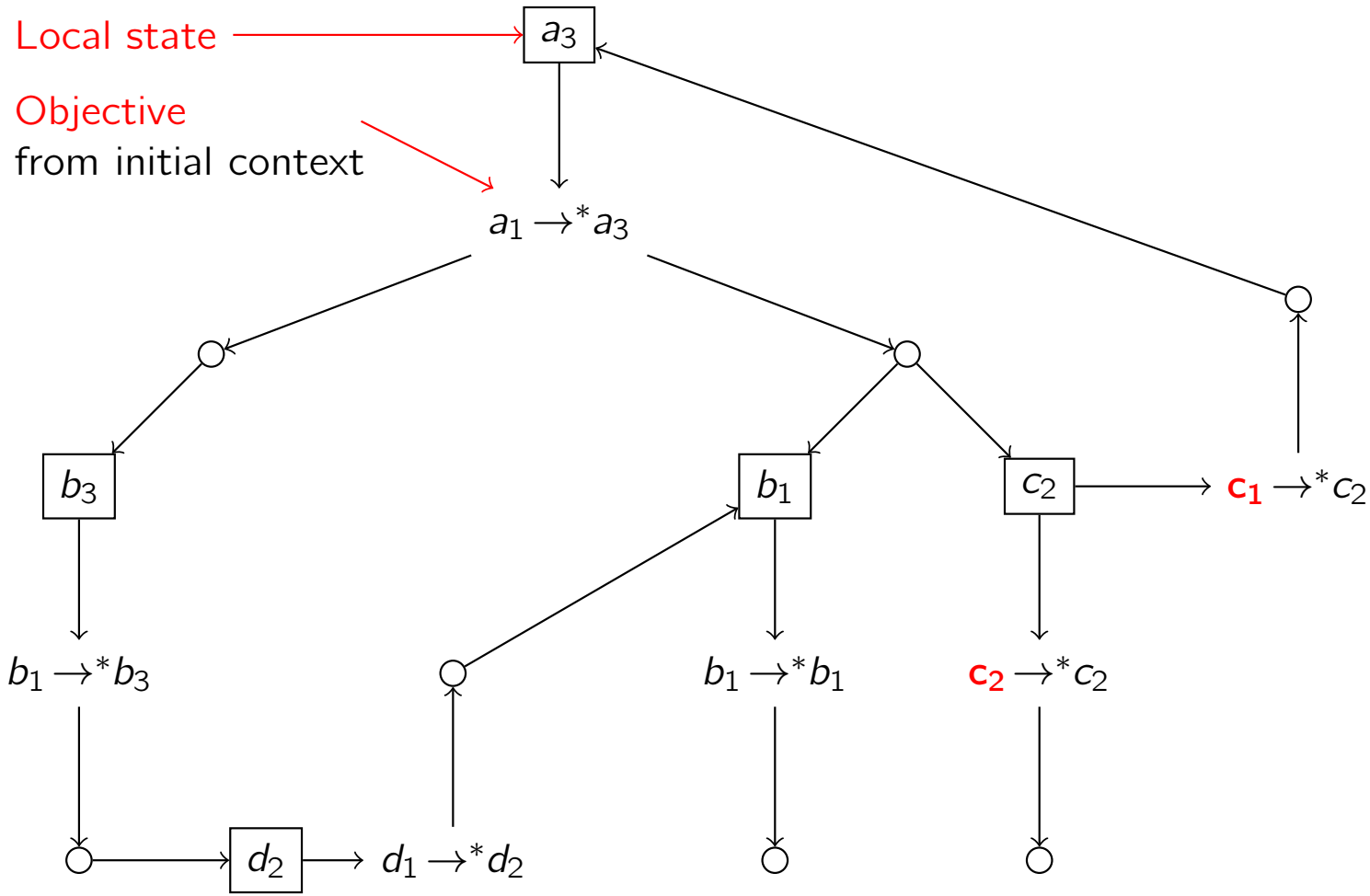
Local Causality Graph

- Causality of a_3.
- Initial context $\varsigma = \{a \mapsto \{1\}; b \mapsto \{1\}; c \mapsto \{1,2\}; d \mapsto \{2\}\}$.



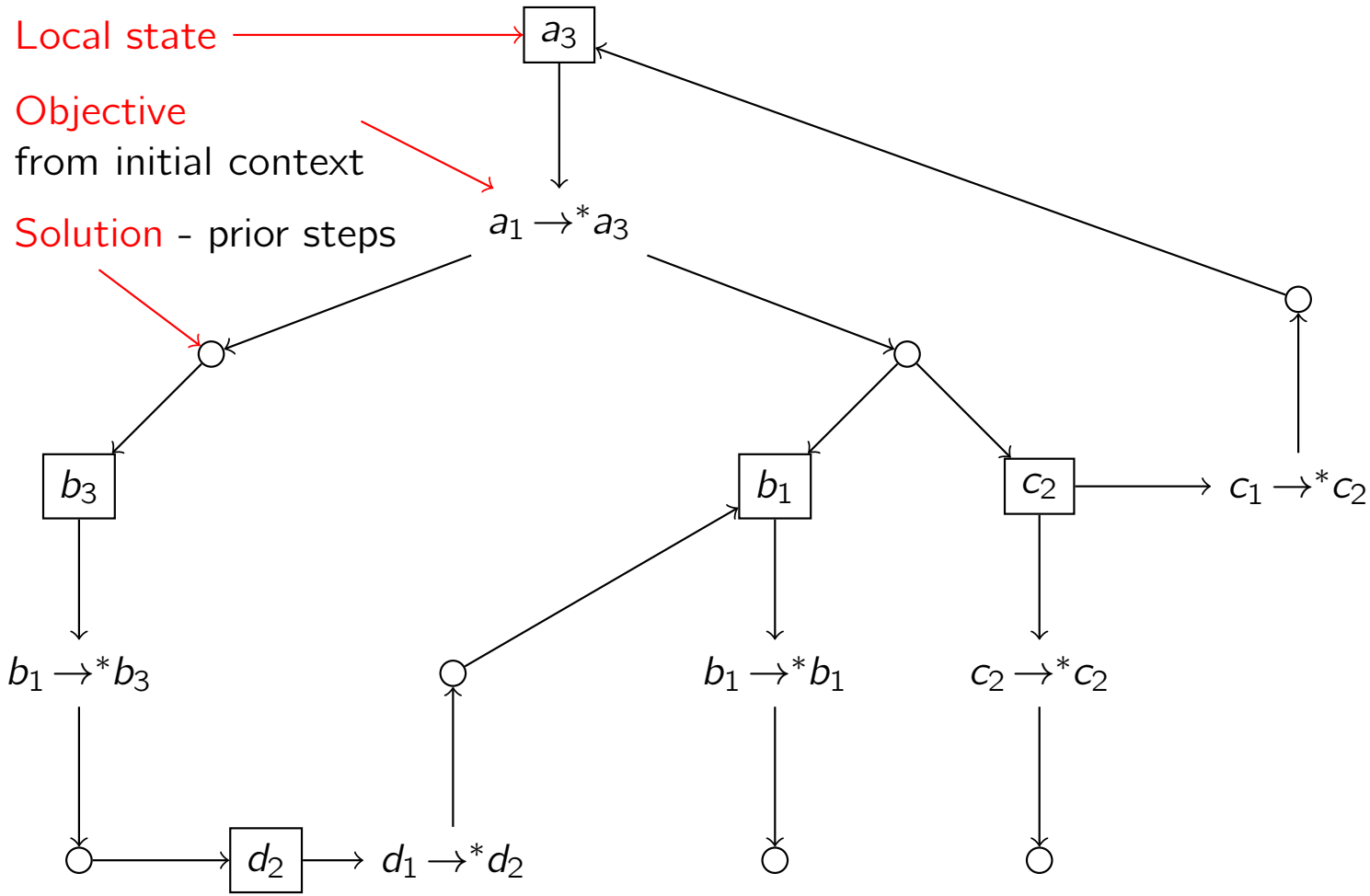
Local Causality Graph

- Causality of a_3.
- Initial context $\varsigma = \{a \mapsto \{1\}; b \mapsto \{1\}; c \mapsto \{1, 2\}; d \mapsto \{2\}\}$.



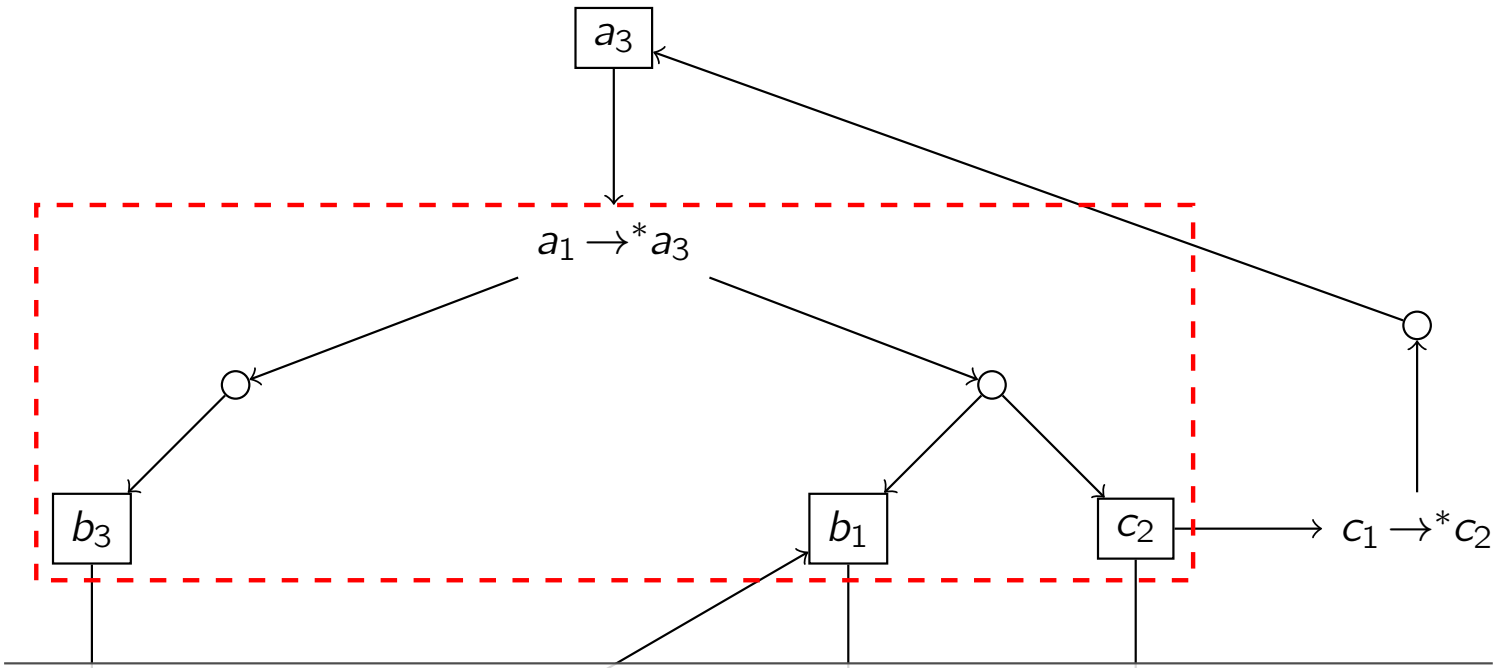
Local Causality Graph

- Causality of a_3.
- Initial context $\varsigma = \{a \mapsto \{1\}; b \mapsto \{1\}; c \mapsto \{1, 2\}; d \mapsto \{2\}\}$.



Local Causality Graph

- Causality of a_3.
- Initial context $\varsigma = \{a \mapsto \{1\}; b \mapsto \{1\}; c \mapsto \{1,2\}; d \mapsto \{2\}\}$.

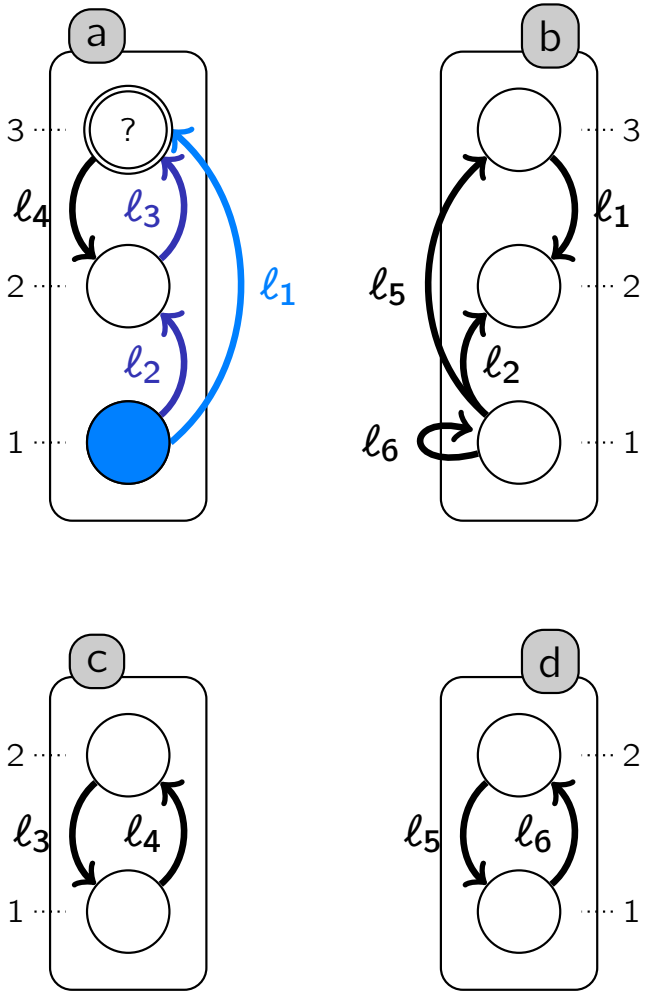


Objective completeness criteria

Objective is impossible from any state if at least one local state of each solution is disabled.

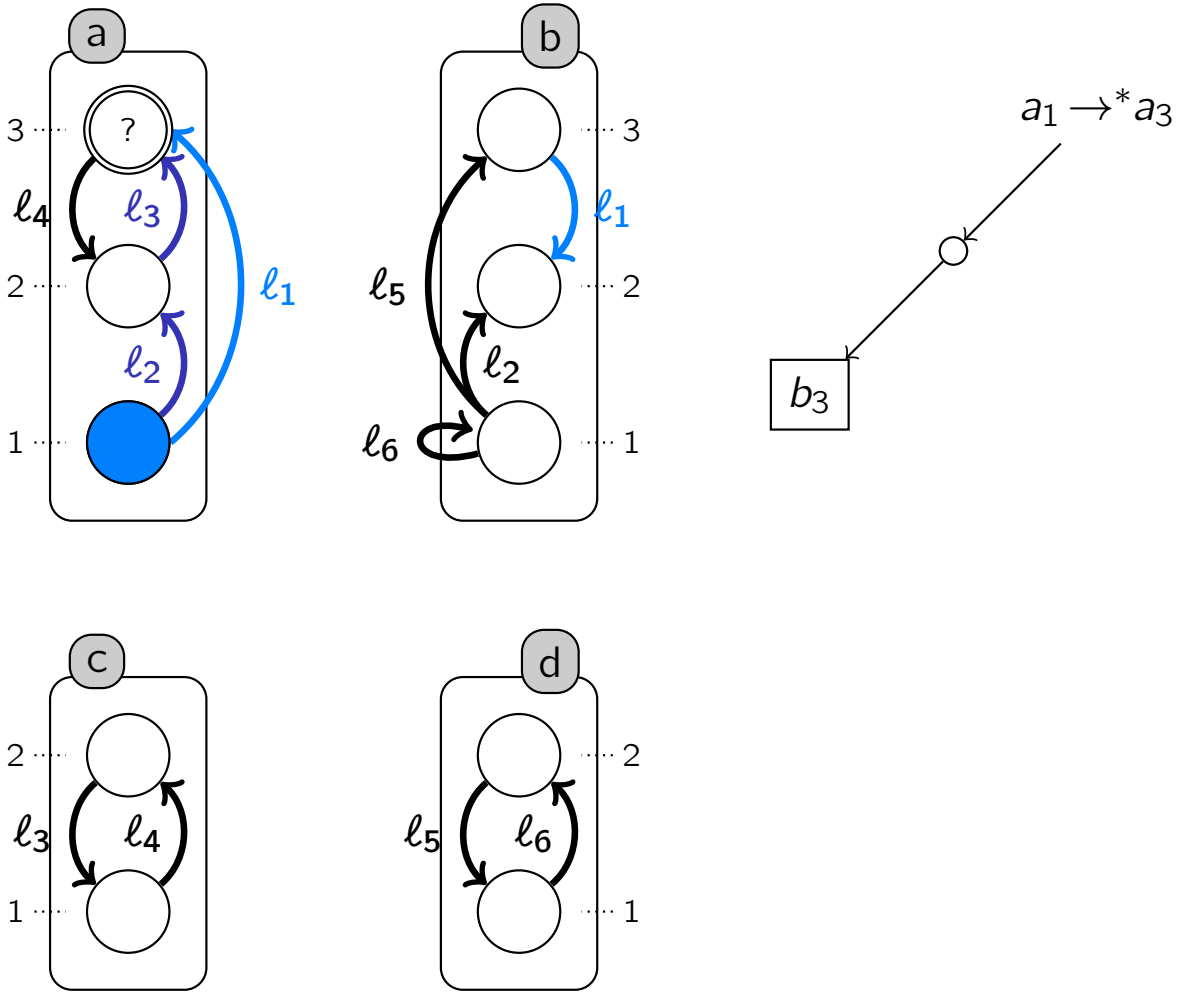
E.g. $a_1 \to^* a_3$ is impossible in $\mathcal{M} \ominus \{b_3, b_1\}$ and in $\mathcal{M} \ominus \{b_3, c_2\}$

Computing LCG for Automata Networks

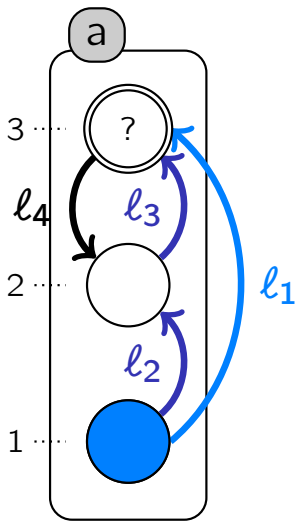


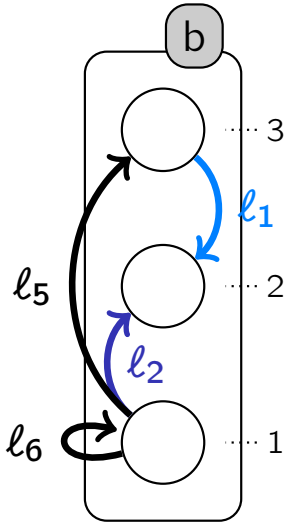
$a_1 \rightarrow^* a_3$

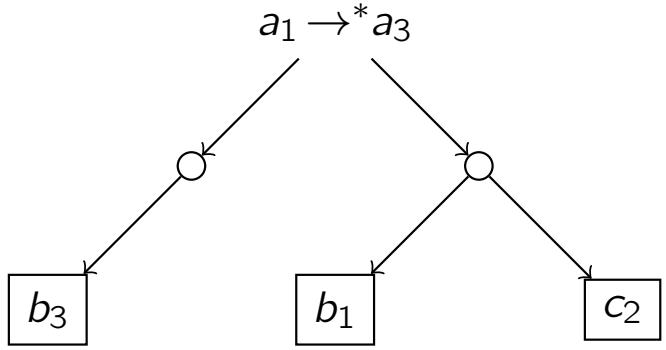
Computing LCG for Automata Networks



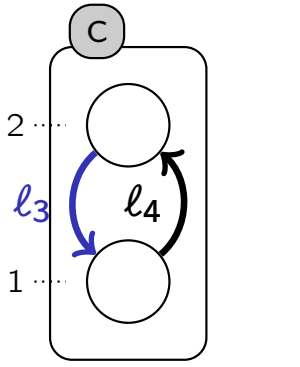
Computing LCG for Automata Networks

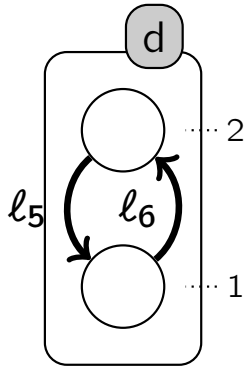




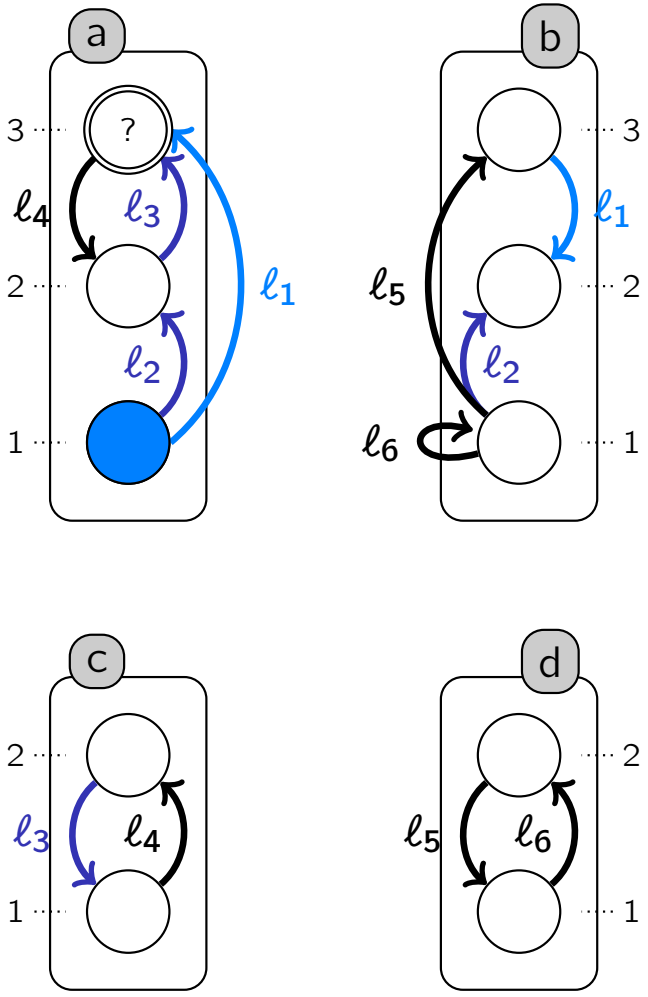


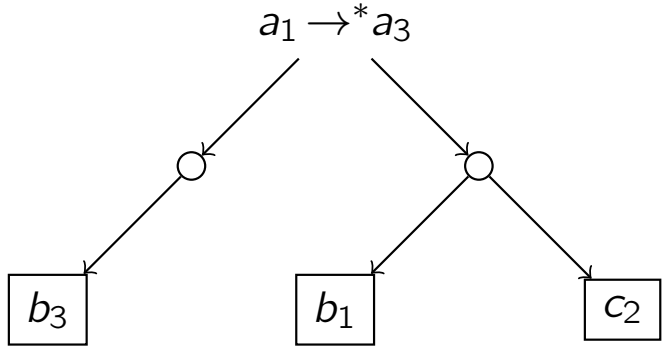
(ignore order, count, synchronism)





Computing LCG for Automata Networks





(ignore order, count, synchronism)

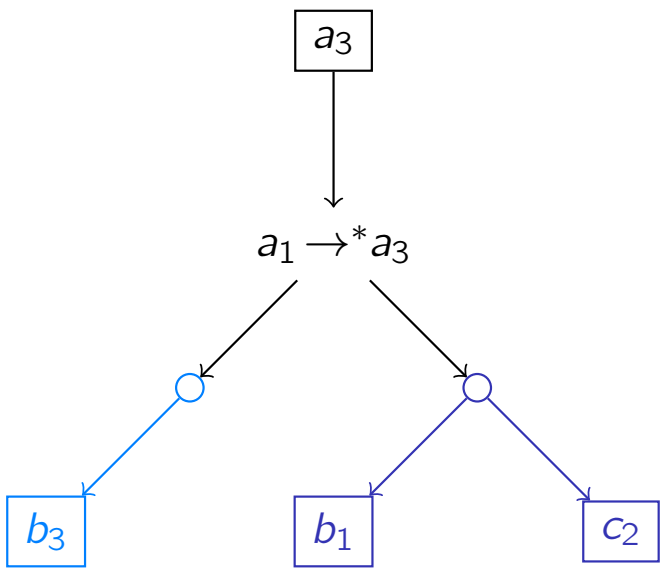
Complexity of LCG (construction + size of LCG)

- polynomial in the total number of local states;
- exponential in the number of local states within one automaton.

Abstract Interpretation with LCG

- ω: successive local reachability property: e.g. $a_i :: b_j :: \cdots z_l$.
- ς: initial context: e.g. $\varsigma = \{a \mapsto \{1\}; b \mapsto \{1\}; c \mapsto \{1,2\}; d \mapsto \{2\}\}$.

$$\gamma_\varsigma(\omega) = \{\delta \in \textbf{traces} \mid \delta \text{ concretizes } \omega \wedge \delta \text{ starts in } \varsigma\}$$



$\Longrightarrow$

$$\gamma_\varsigma(a_3) = \gamma_\varsigma(b_3 :: a_3) \cup \gamma_\varsigma(b_1 :: c_2 :: a_3) \cup \gamma_\varsigma(c_2 :: b_1 :: a_3)$$

Reachability Analysis using LCG

Given

- ω: successive local reachability property: e.g. $a_i :: b_j :: \cdots z_l$.
- ς: initial context: e.g. $\varsigma = \{a \mapsto \{1\}; b \mapsto \{1\}; c \mapsto \{1,2\}; d \mapsto \{2\}\}$.

decide if $\gamma_\varsigma(\omega) \neq \emptyset$.

Results

- Necessary condition for general case.
- Stronger necessary conditions for Asynchronous ANs.
- Sufficient condition for Asynchronous ANs.

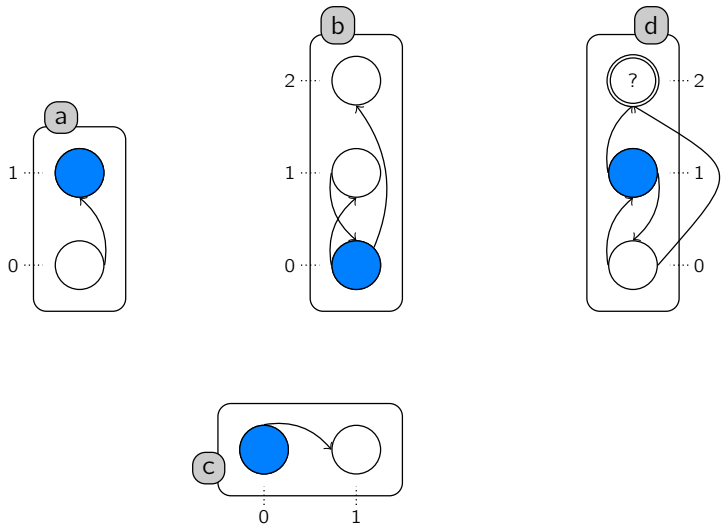
Over-: polynomial in the size of the LCG;
Under-: same or exp. with nb of solutions per objective.

Cut sets for a_i

- Sets of local states whose activity is necessary for $\gamma_\varsigma(a_i) \neq \emptyset$.
- Under-approximation using LCG (some are missed, some are non-minimal).
- General to any AN.

Necessary conditions for reachability

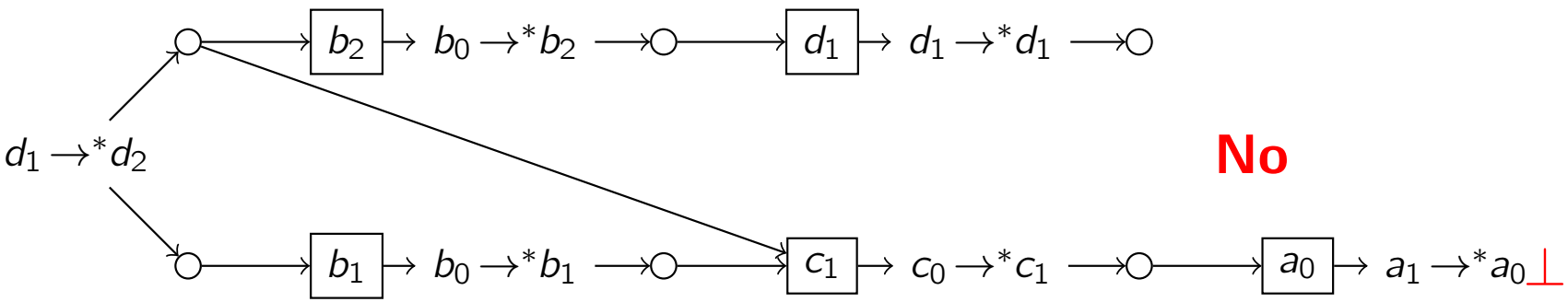
Example



Necessary condition for $\gamma_s(d_2) \neq \emptyset$:

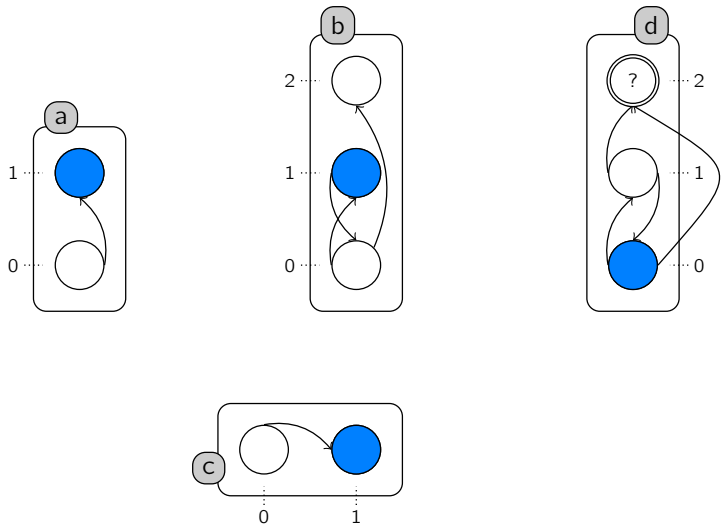
There exists a traversal of the LCG s.t.:

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



Necessary conditions for reachability

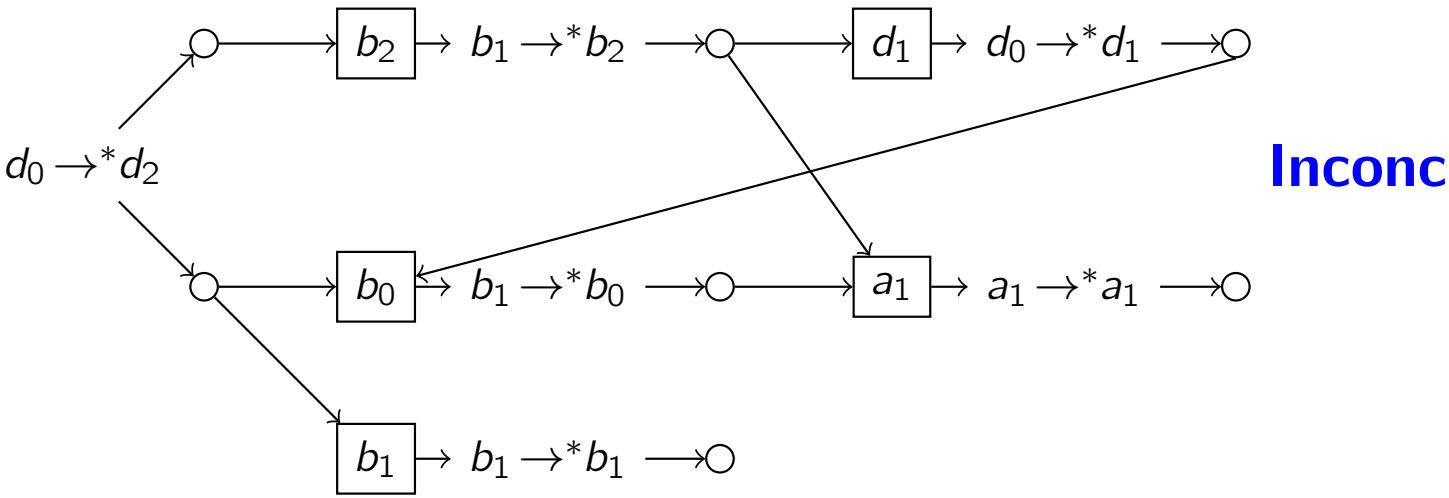
Example



Necessary condition for $\gamma_s(d_2) \neq \emptyset$:

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

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



Inconc

Necessary conditions for reachability

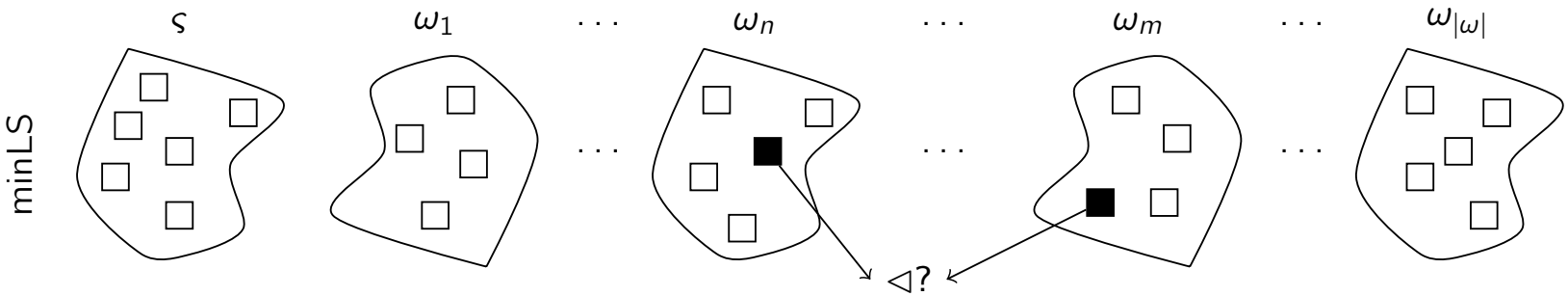
Stronger conditions for AANs

Scope: Asynchronous ANs (transitions change only one automaton).

Take sequentiality into account

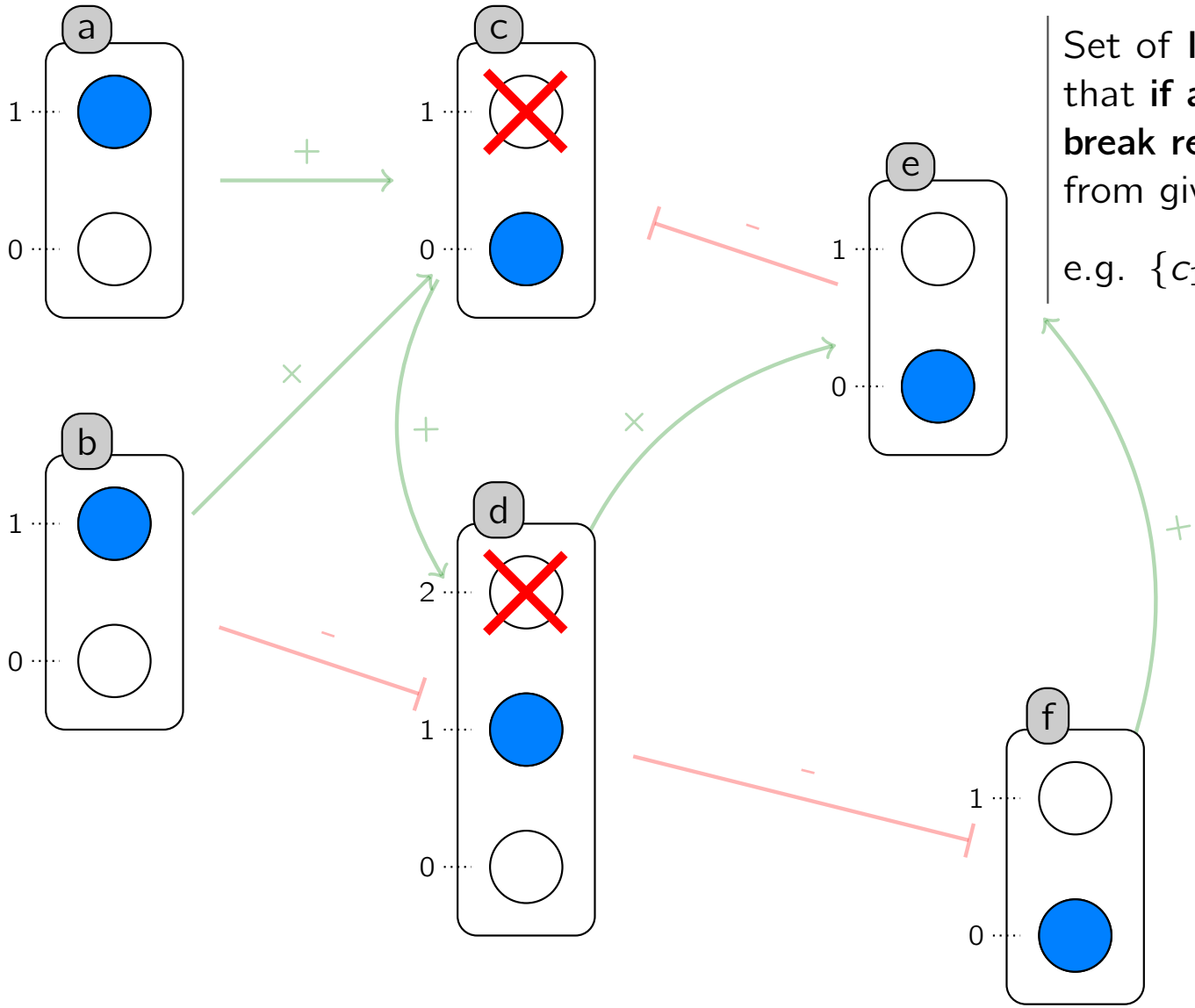
- $\gamma_\varsigma(a_i :: \omega) \neq \emptyset \Longrightarrow \gamma_{max_\varsigma}(\omega) \neq \emptyset$
- Over-approximate next context using LCG.
- (time polynomial with size of LCG).

Local states occurrence order constraints



- Extract local states necessarily used for reachability using LCG.
- Check ordering constraints (e.g.: $a_j \lhd a_i$ if $\text{sol}(a_i \rightarrow^* a_j) = \emptyset$).

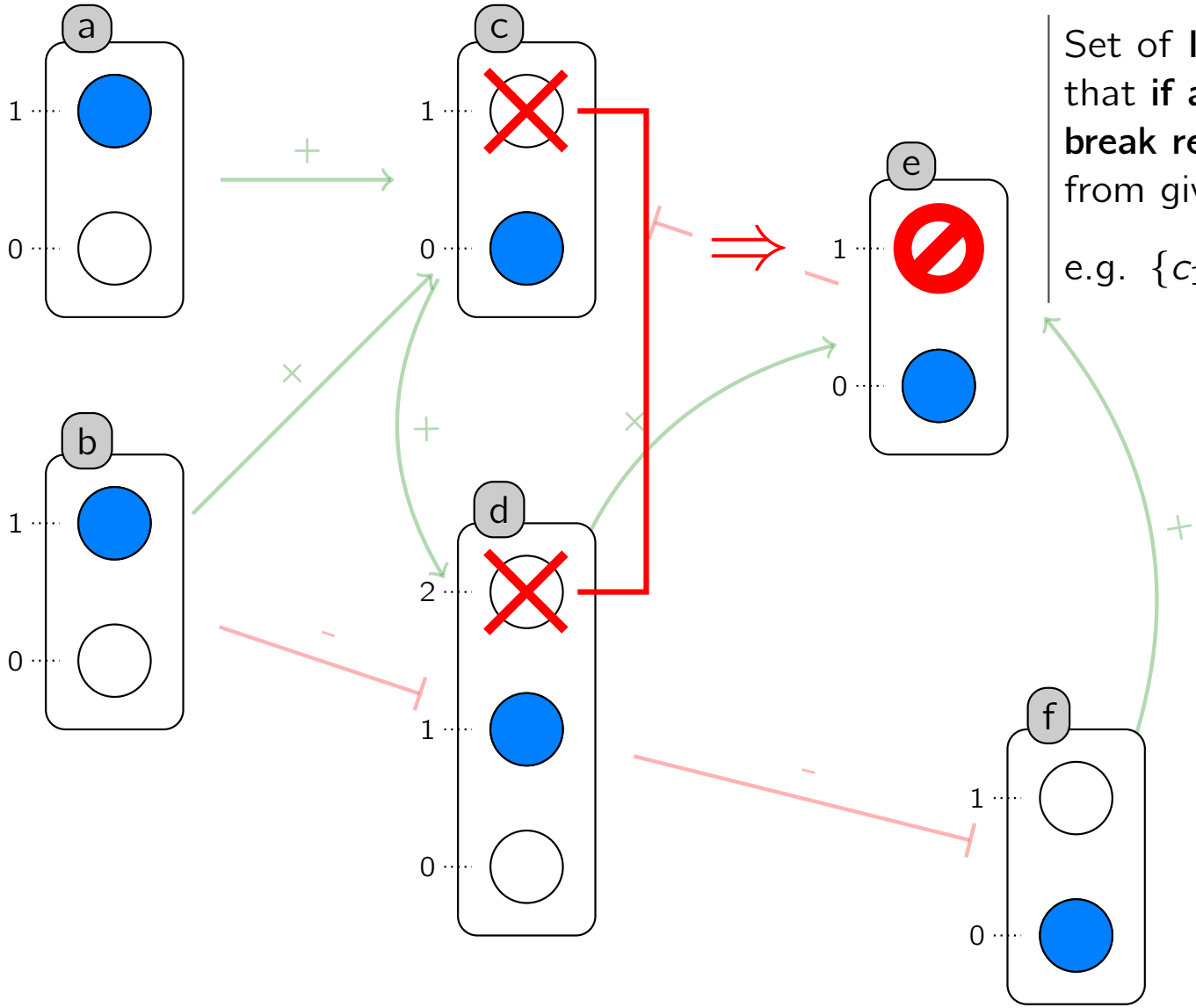
Cut Sets for Reachability



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

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

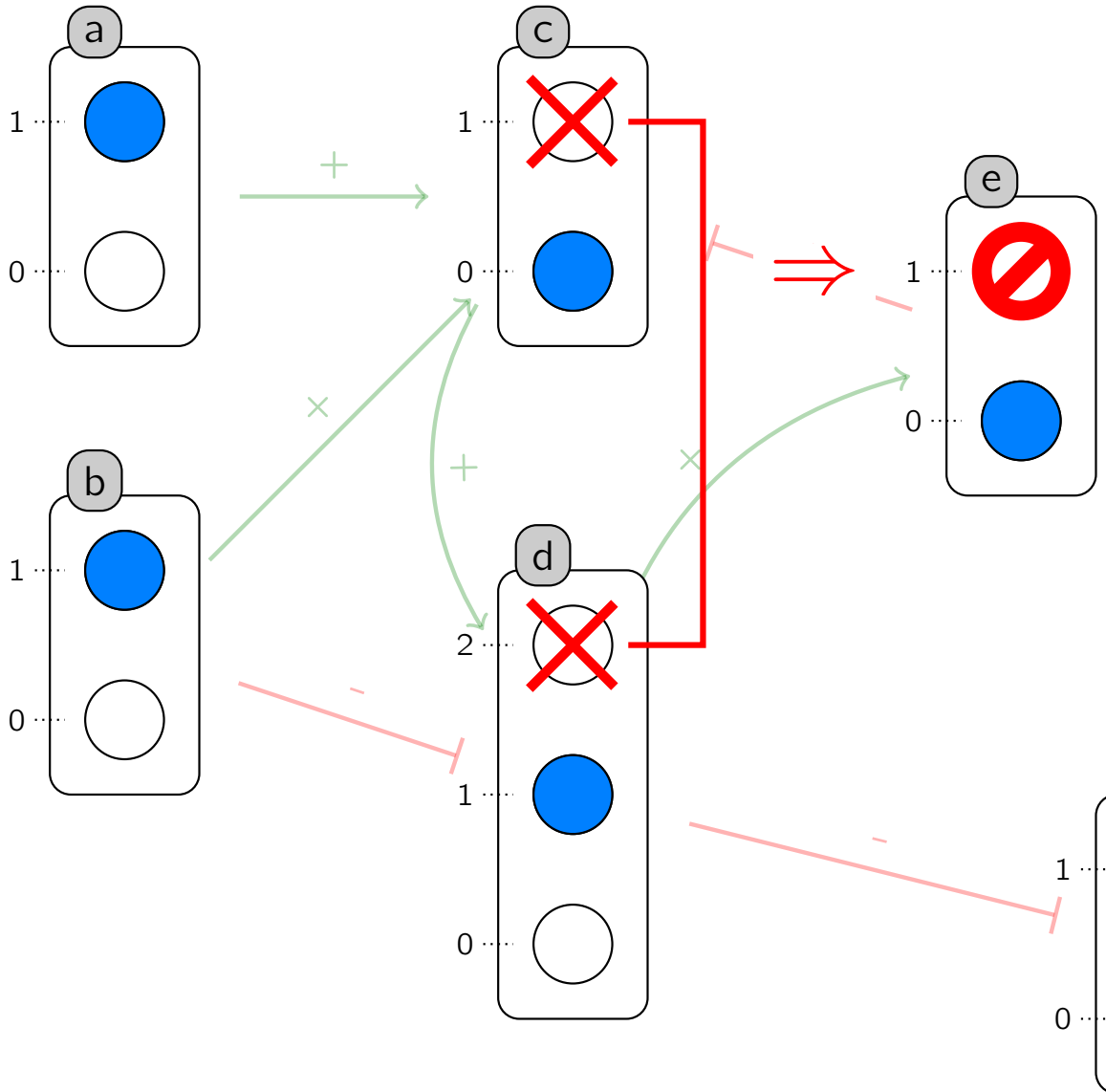
Cut Sets for Reachability



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

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

Cut Sets for Reachability



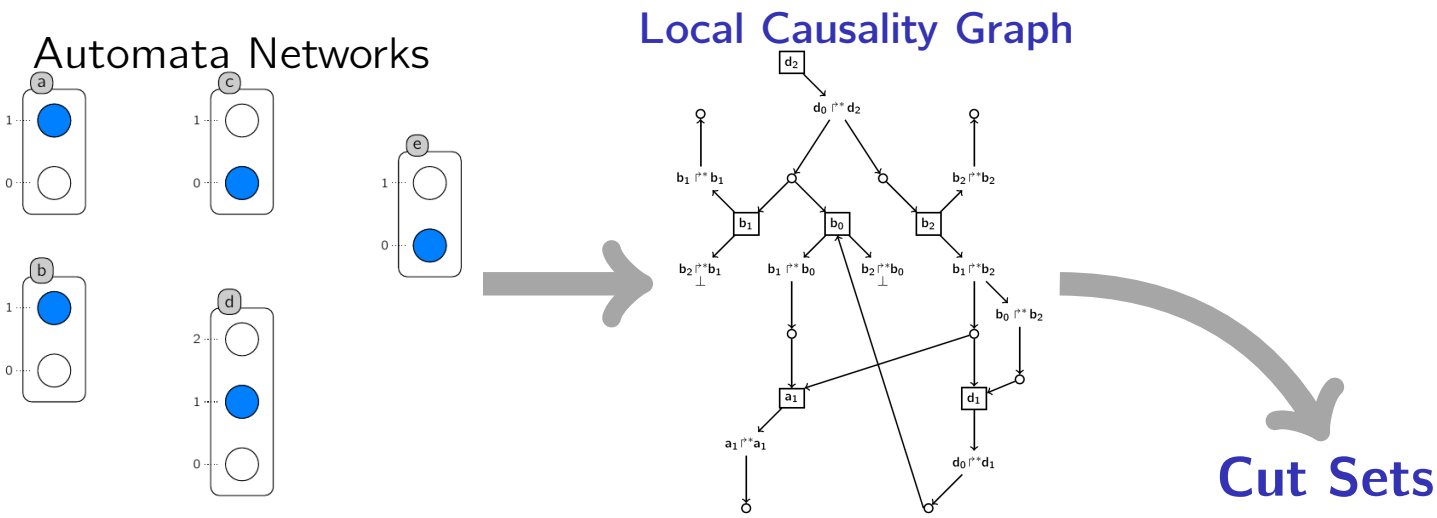
Set of **local states** that **if all disabled break reachability** from given initial states

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

Applications

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

Cut Sets for Reachability



Algorithm

- Graph flooding algorithm (main idea: break necessary condition).
- Computes all cut sets at once: no enumeration of candidates.
- Very efficient with large networks.

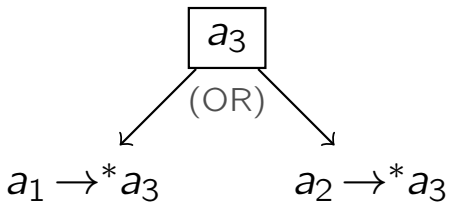
Returned cut sets

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

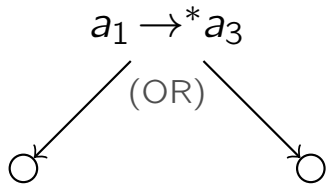
Cut Sets Under-Approximation

Associate to each node sets of local states intersecting *any* trace from given context.

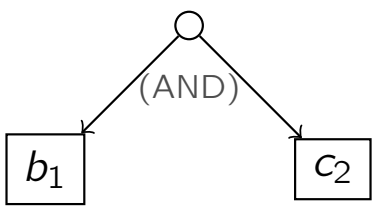
$$\mathbb{V} : \text{nodes} \mapsto \wp(\wp^{\leq N}(\mathcal{O}bs)), \mathcal{O}bs \subset \mathbf{LS}$$



$$\mathbb{V}(a_3) = \mathbb{V}(a_1 \rightarrow^* a_3) \tilde{\times} \mathbb{V}(a_2 \rightarrow^* a_3) \cup \{\{a_3\}\}$$



$$\mathbb{V}(a_1 \rightarrow^* a_3) = \mathbb{V}(sol^1) \tilde{\times} (sol^2)$$



$$\mathbb{V}(sol^1) = \mathbb{V}(b_1) \cup \mathbb{V}(c_2)$$

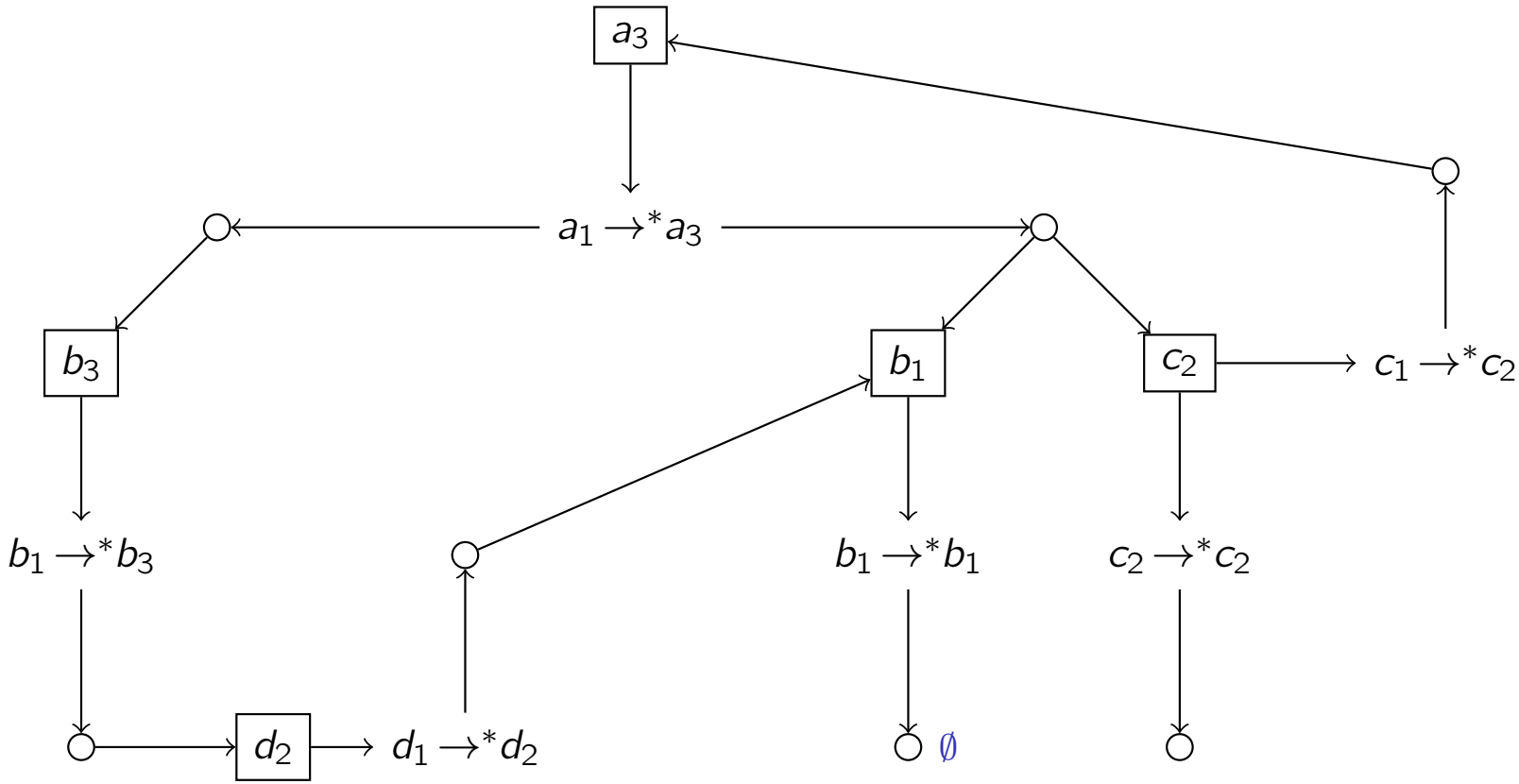
$$\{e^1, \ldots, e^n\} \tilde{\times} \{f^1, \ldots, f^m\} \triangleq \{e^i \cup f^j \mid i \in [1; n] \wedge j \in [1; m]\} \; ; \; e^i, f^j \in \wp^{\leq N}(\mathcal{O}bs)$$

Cut Sets Under-approximation

Example

Sketch

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

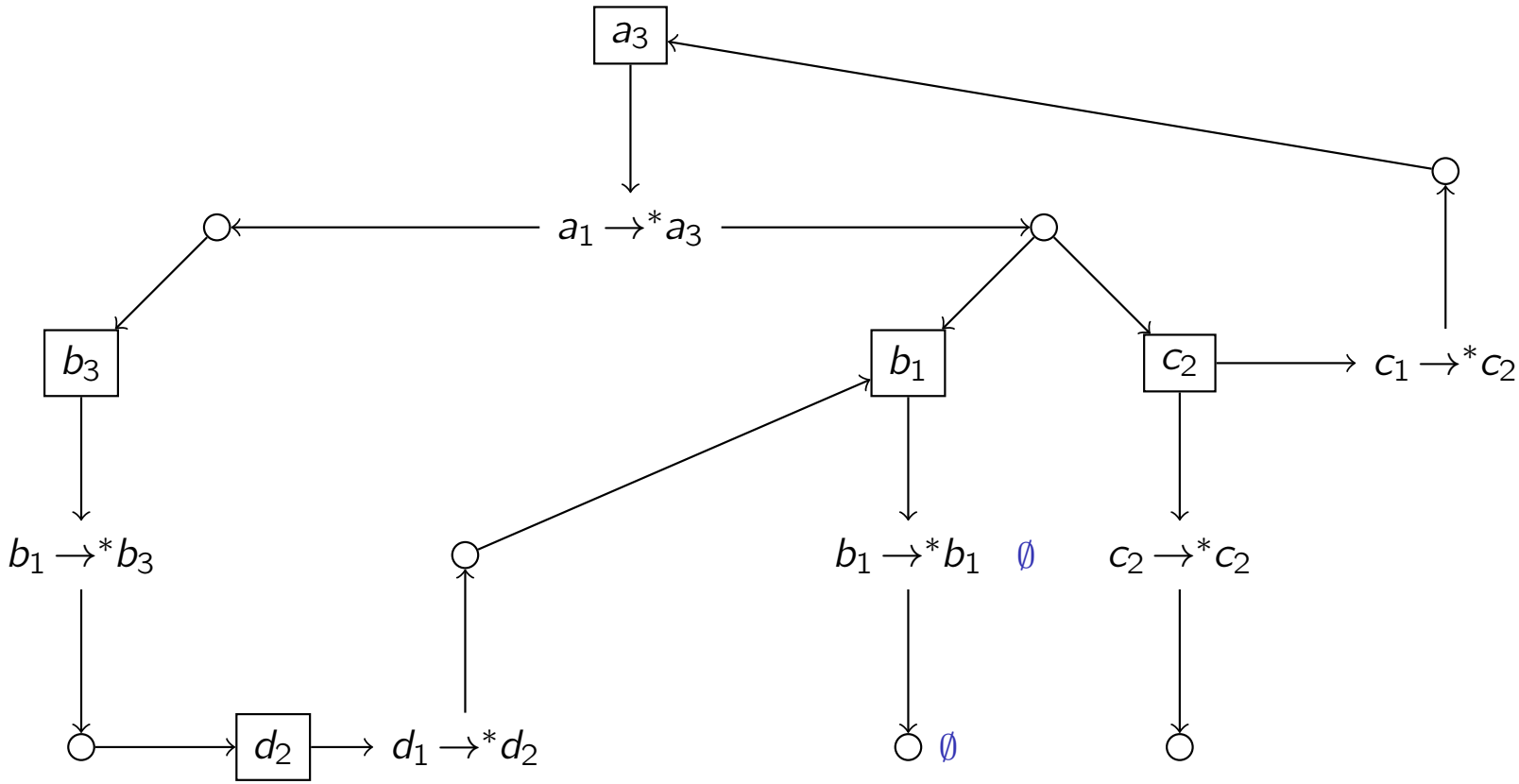


Cut Sets Under-approximation

Example

Sketch

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

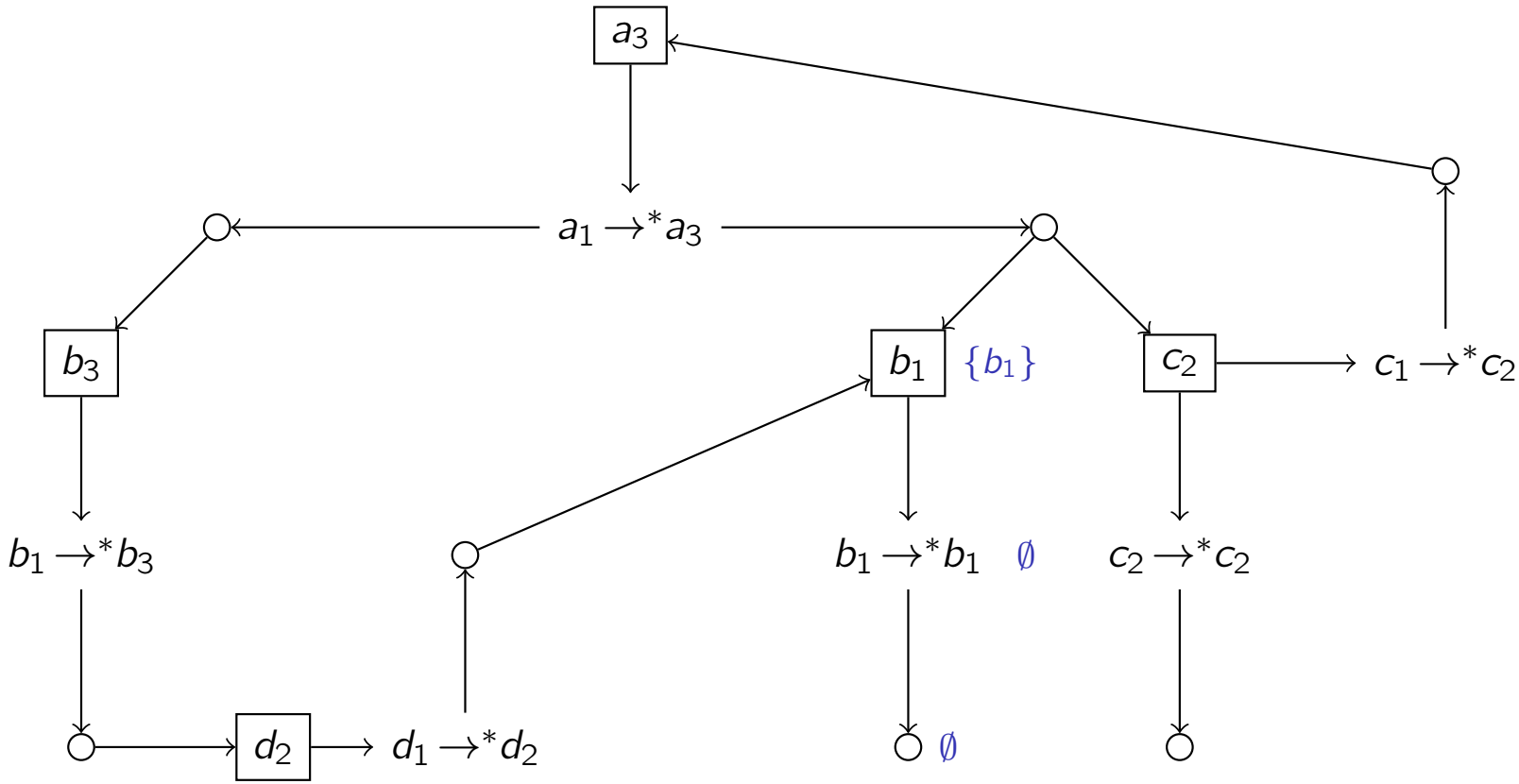


Cut Sets Under-approximation

Example

Sketch

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

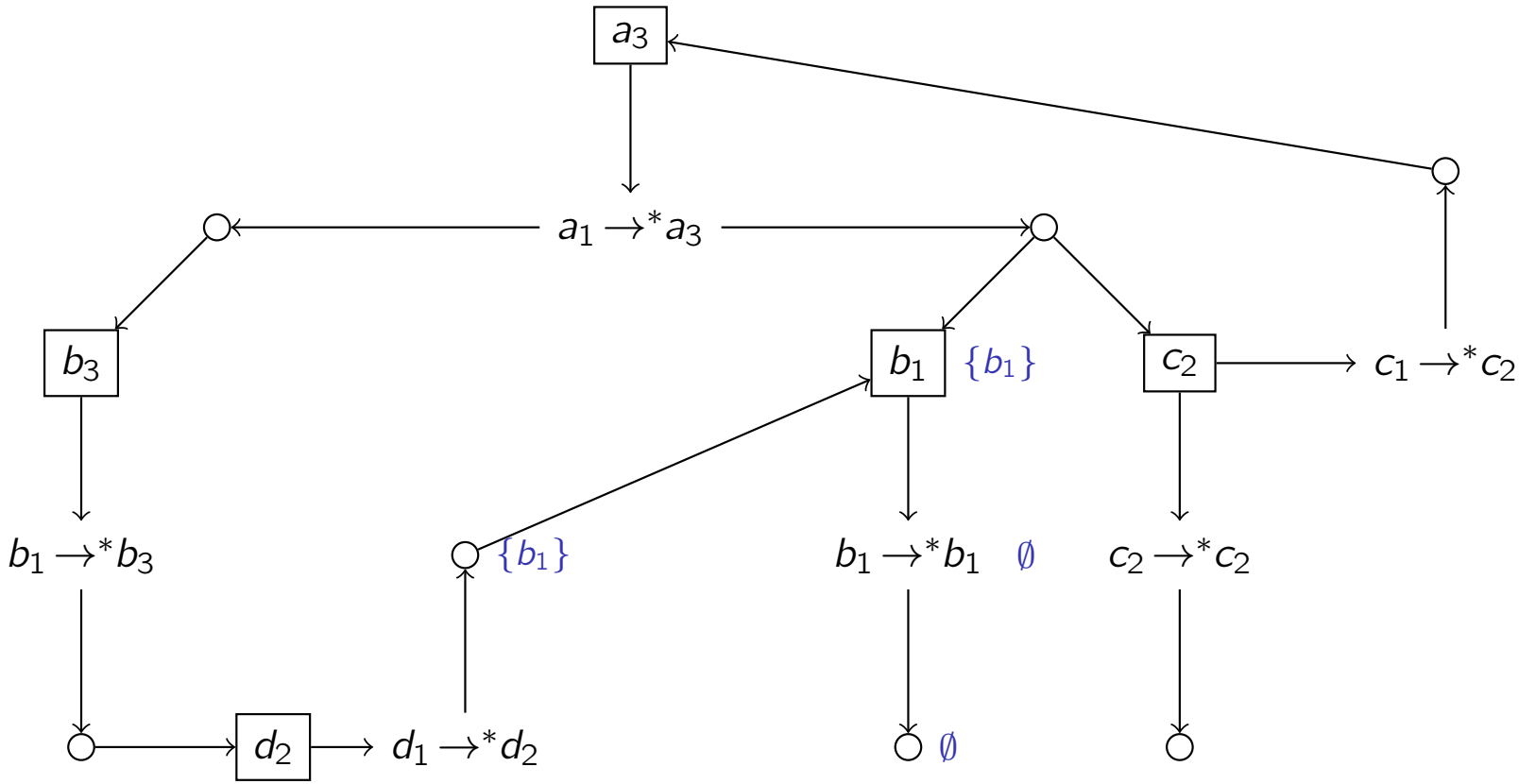


Cut Sets Under-approximation

Example

Sketch

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

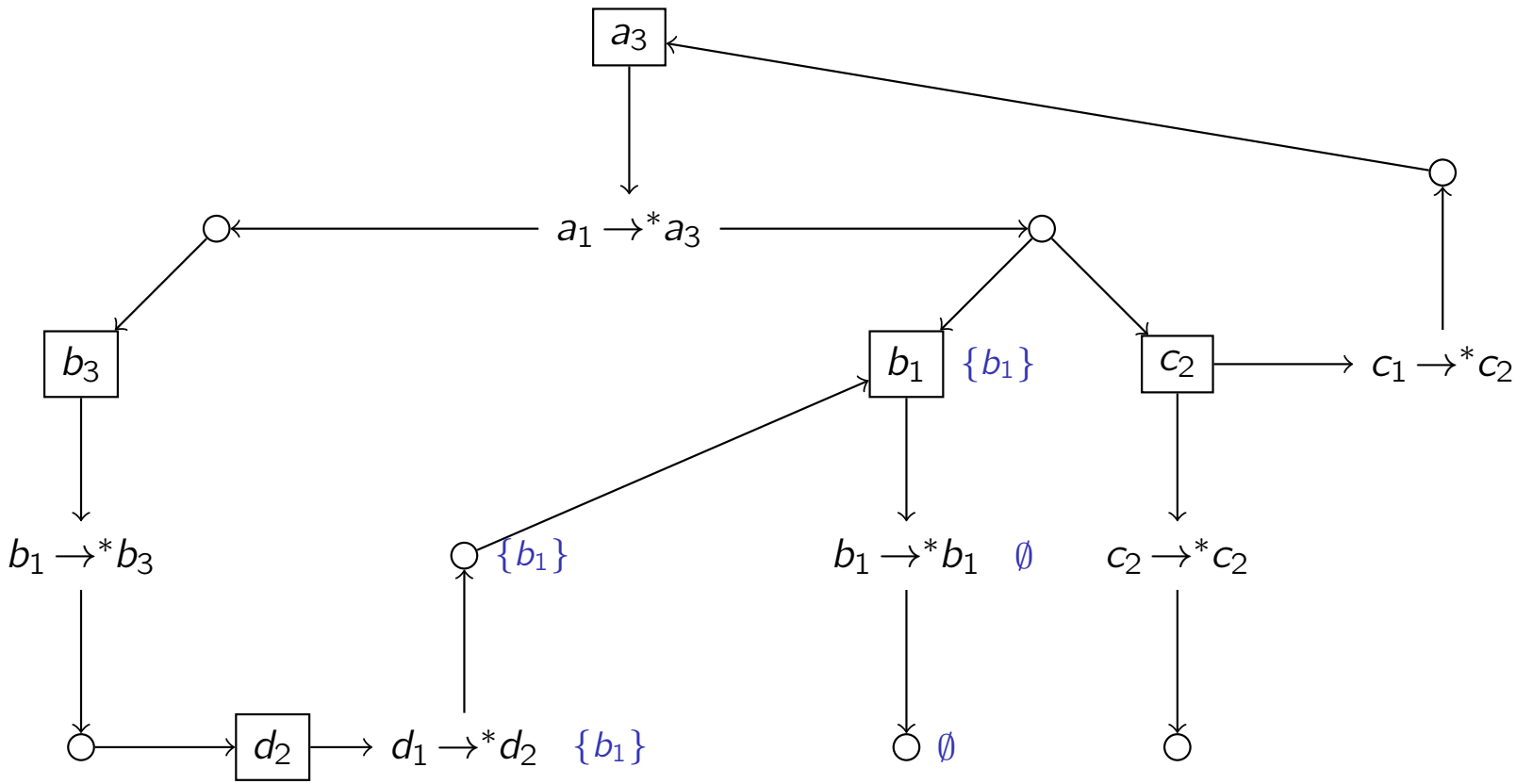


Cut Sets Under-approximation

Example

Sketch

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

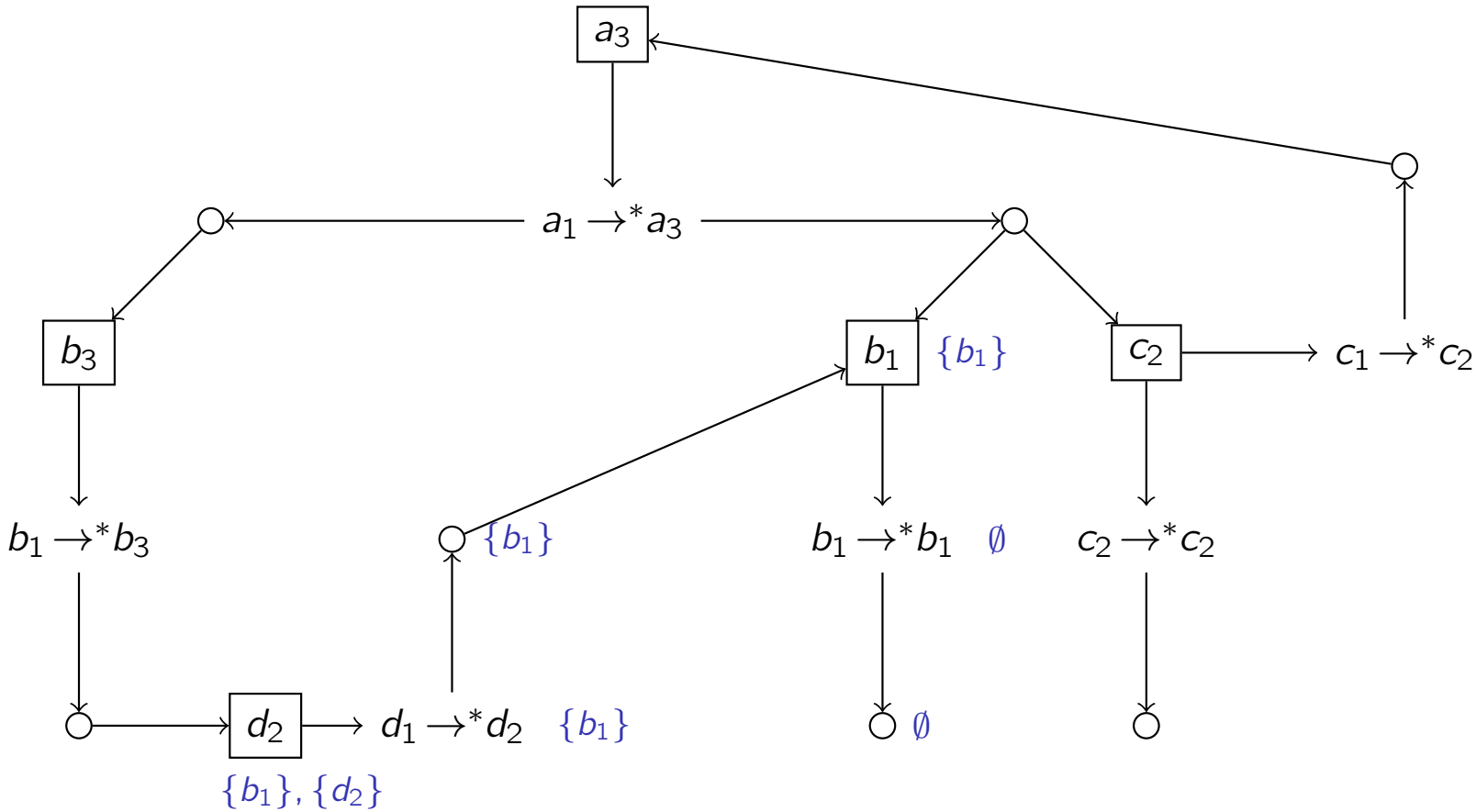


Cut Sets Under-approximation

Example

Sketch

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

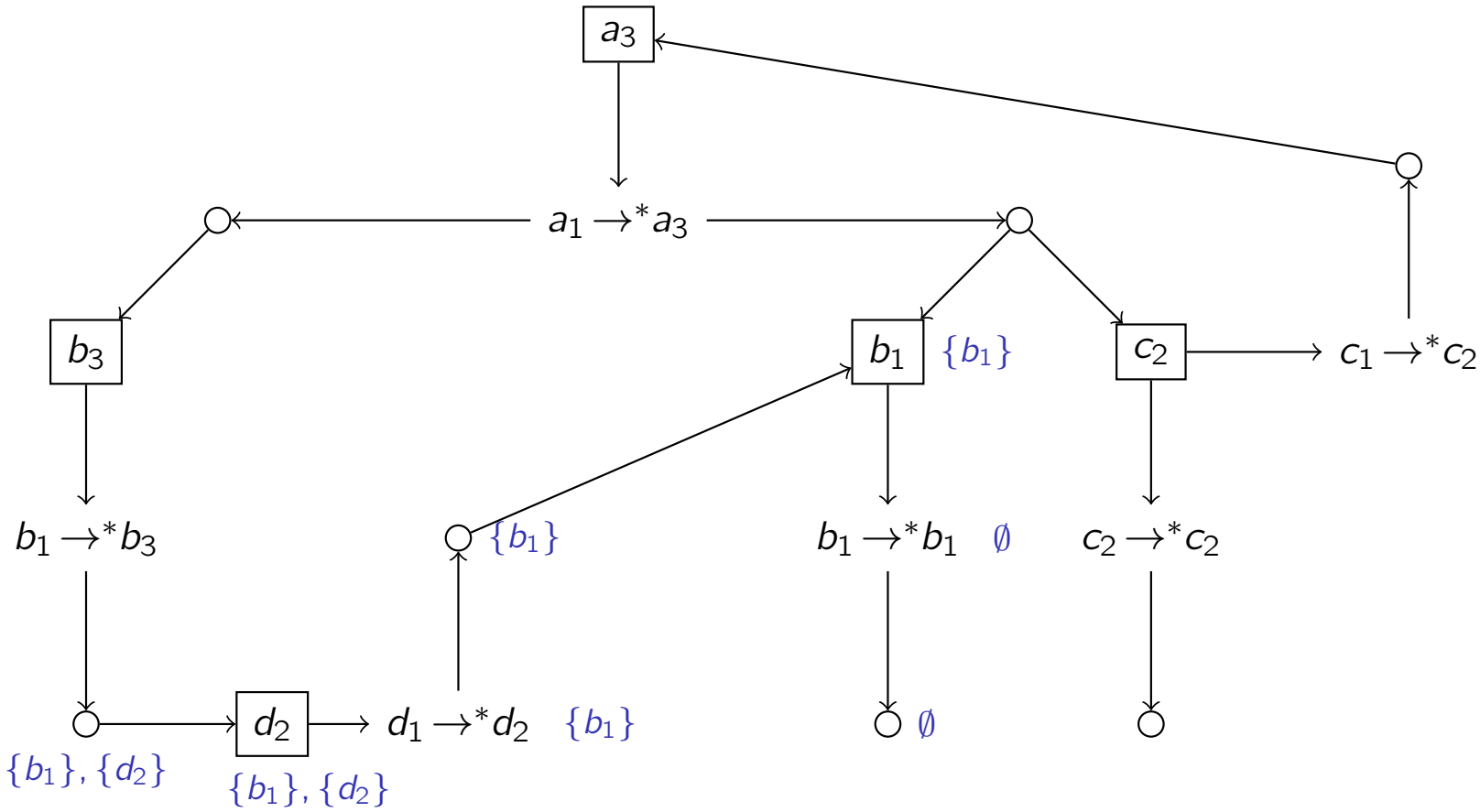


Cut Sets Under-approximation

Example

Sketch

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

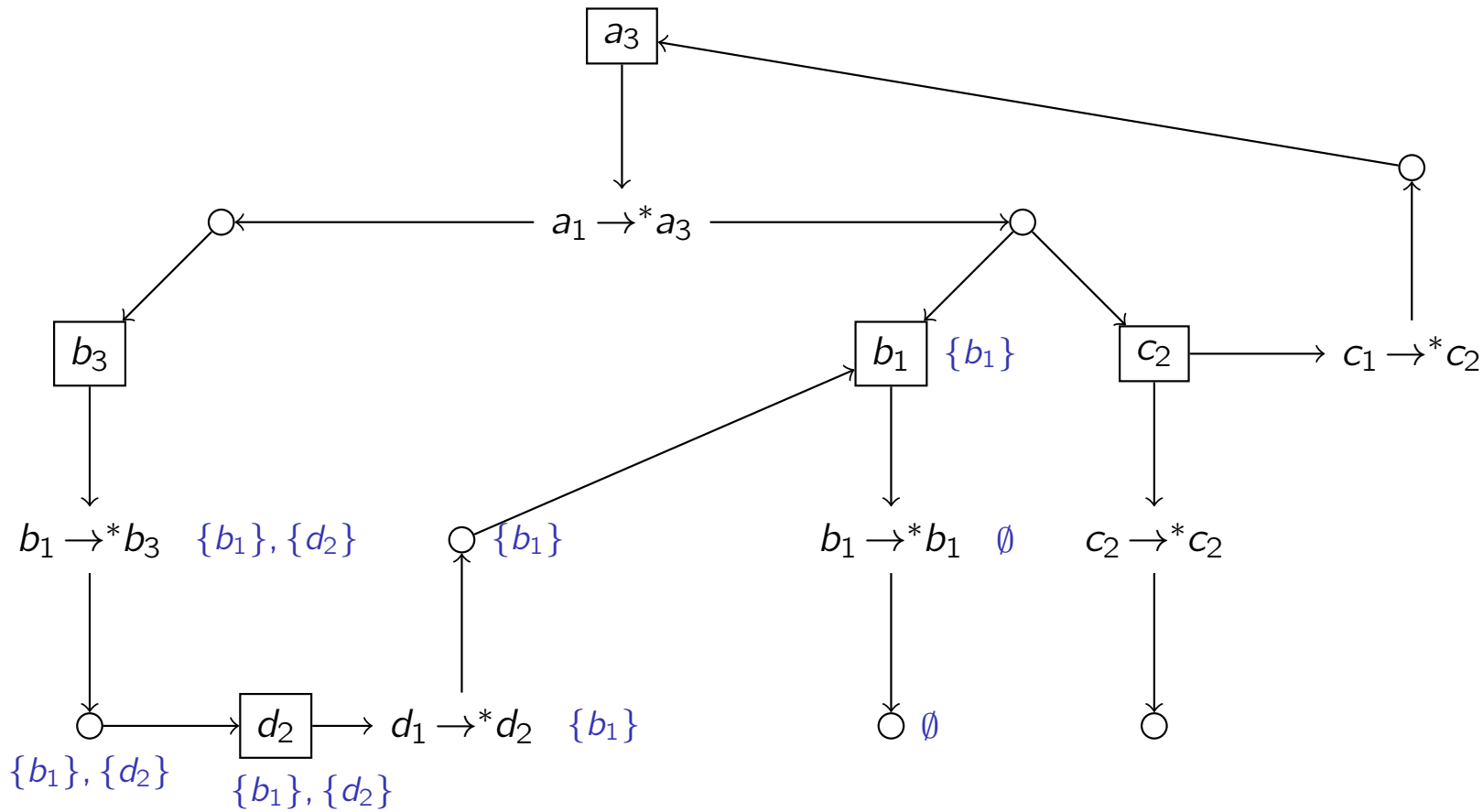


Cut Sets Under-approximation

Example

Sketch

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

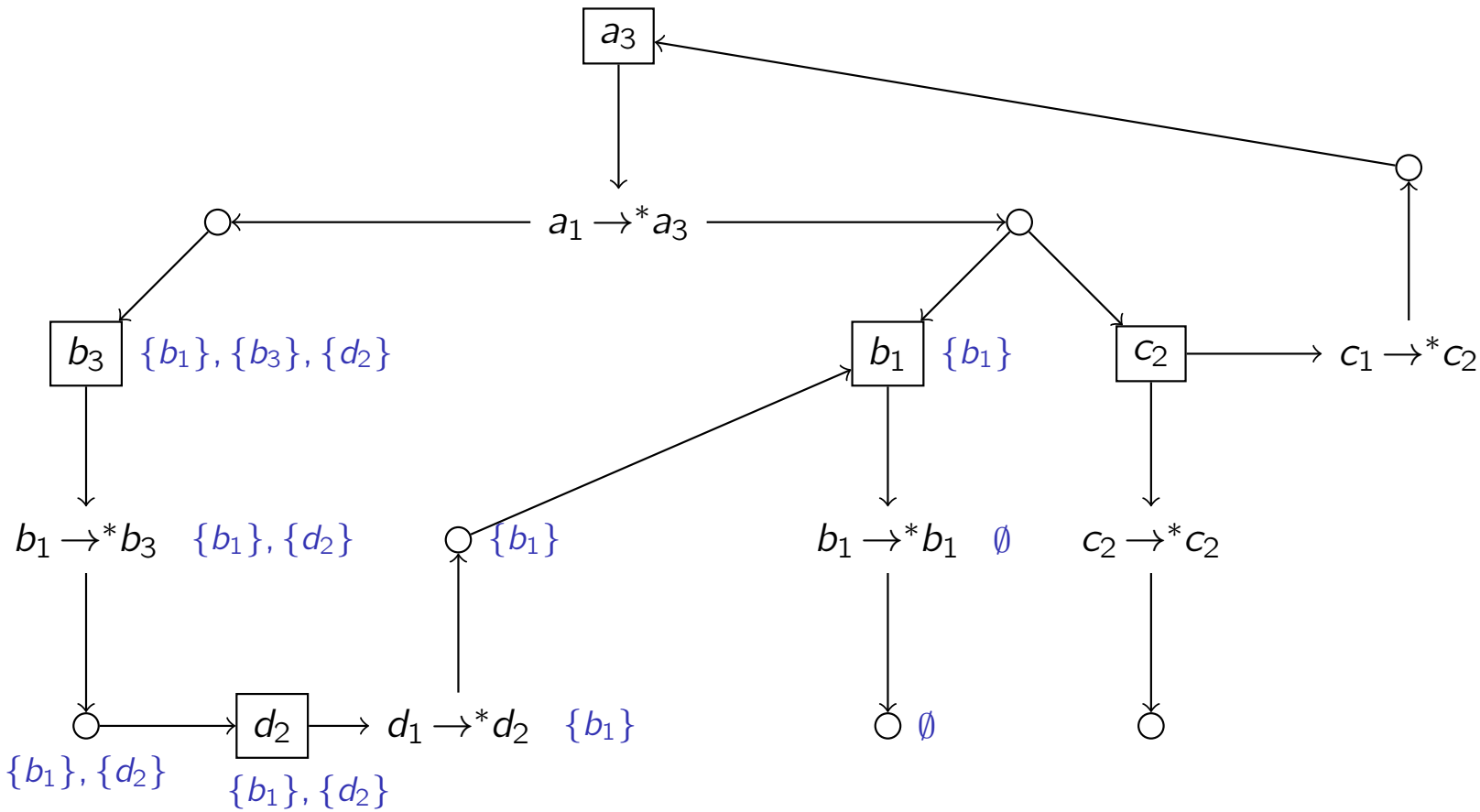


Cut Sets Under-approximation

Example

Sketch

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

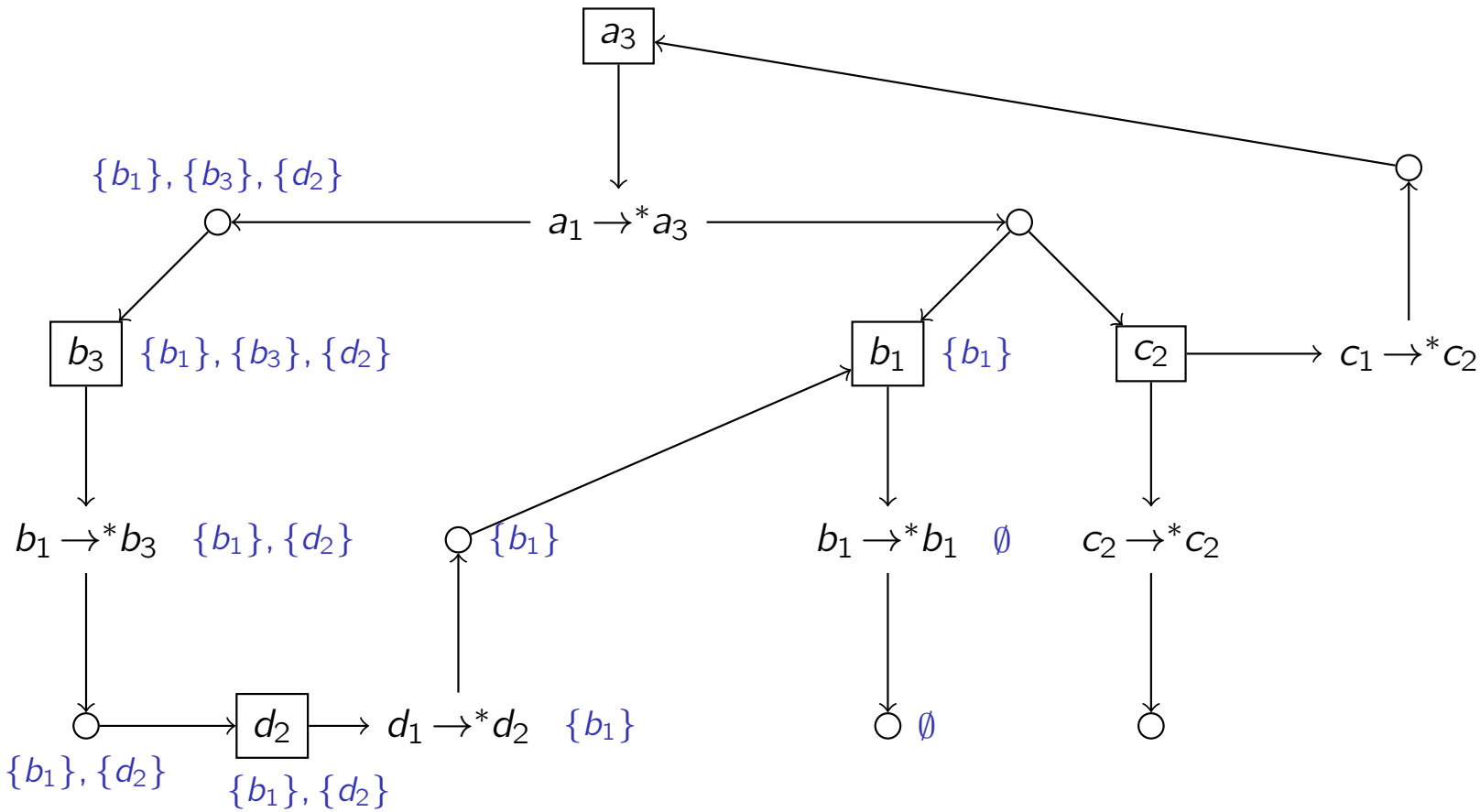


Cut Sets Under-approximation

Example

Sketch

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

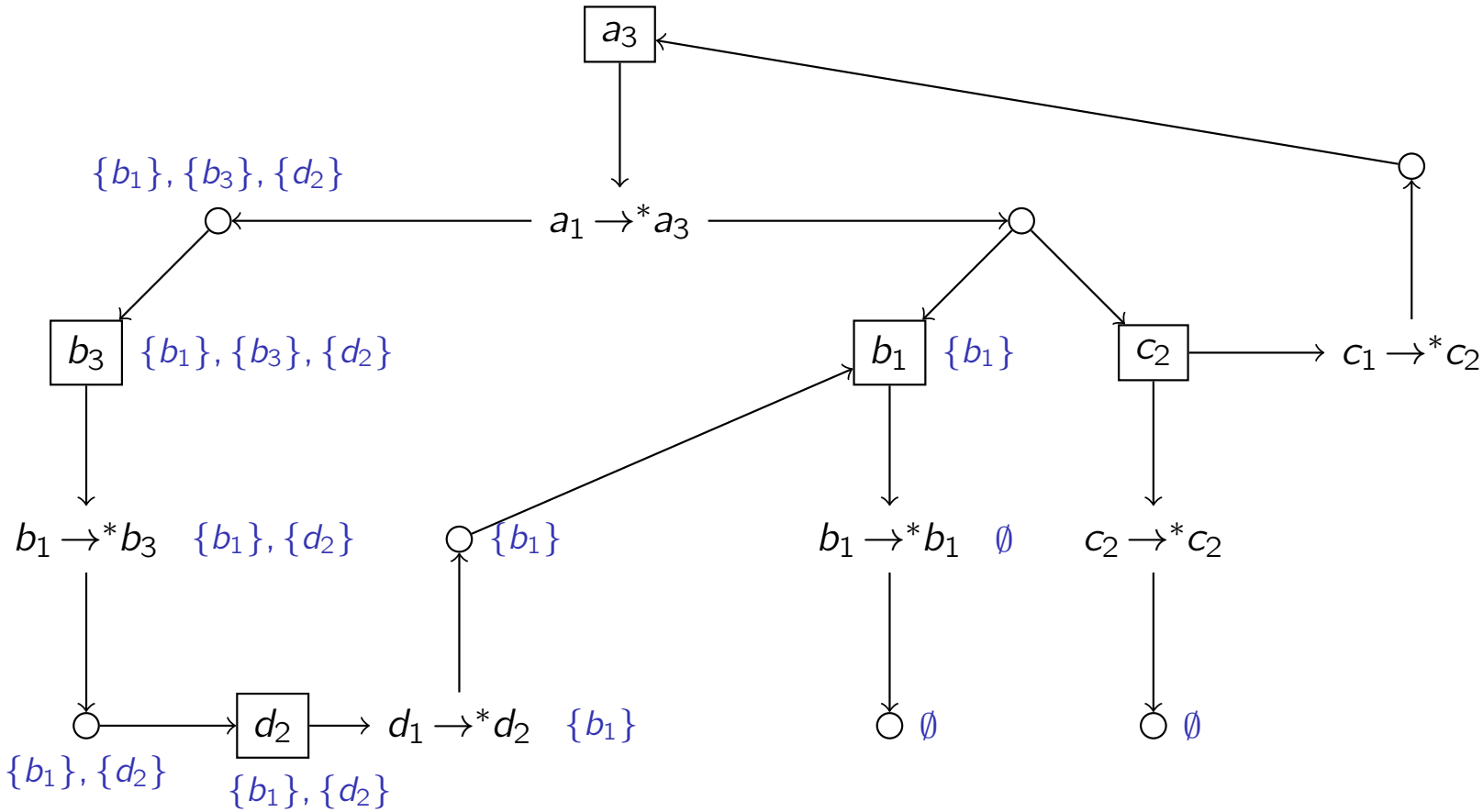


Cut Sets Under-approximation

Example

Sketch

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

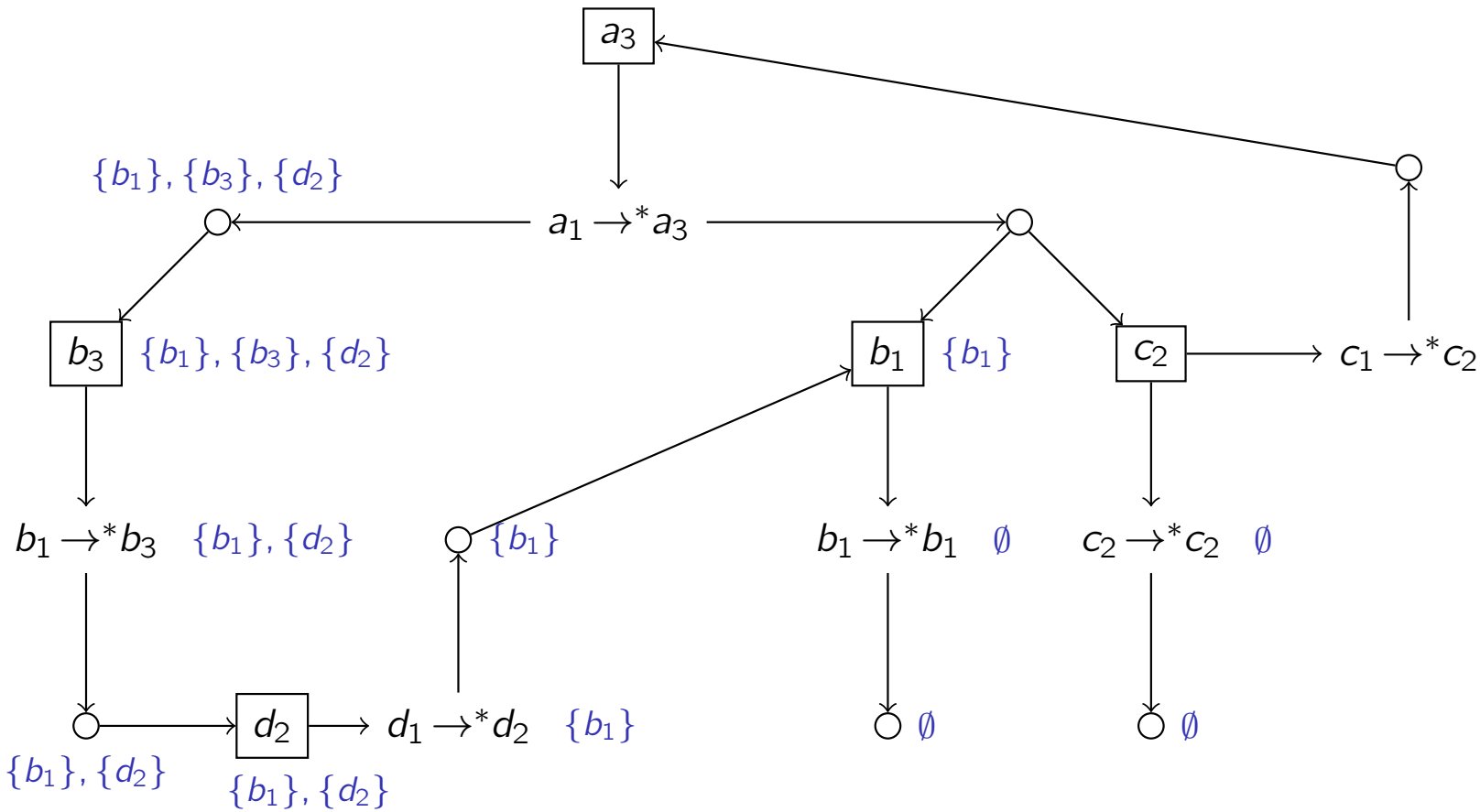


Cut Sets Under-approximation

Example

Sketch

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

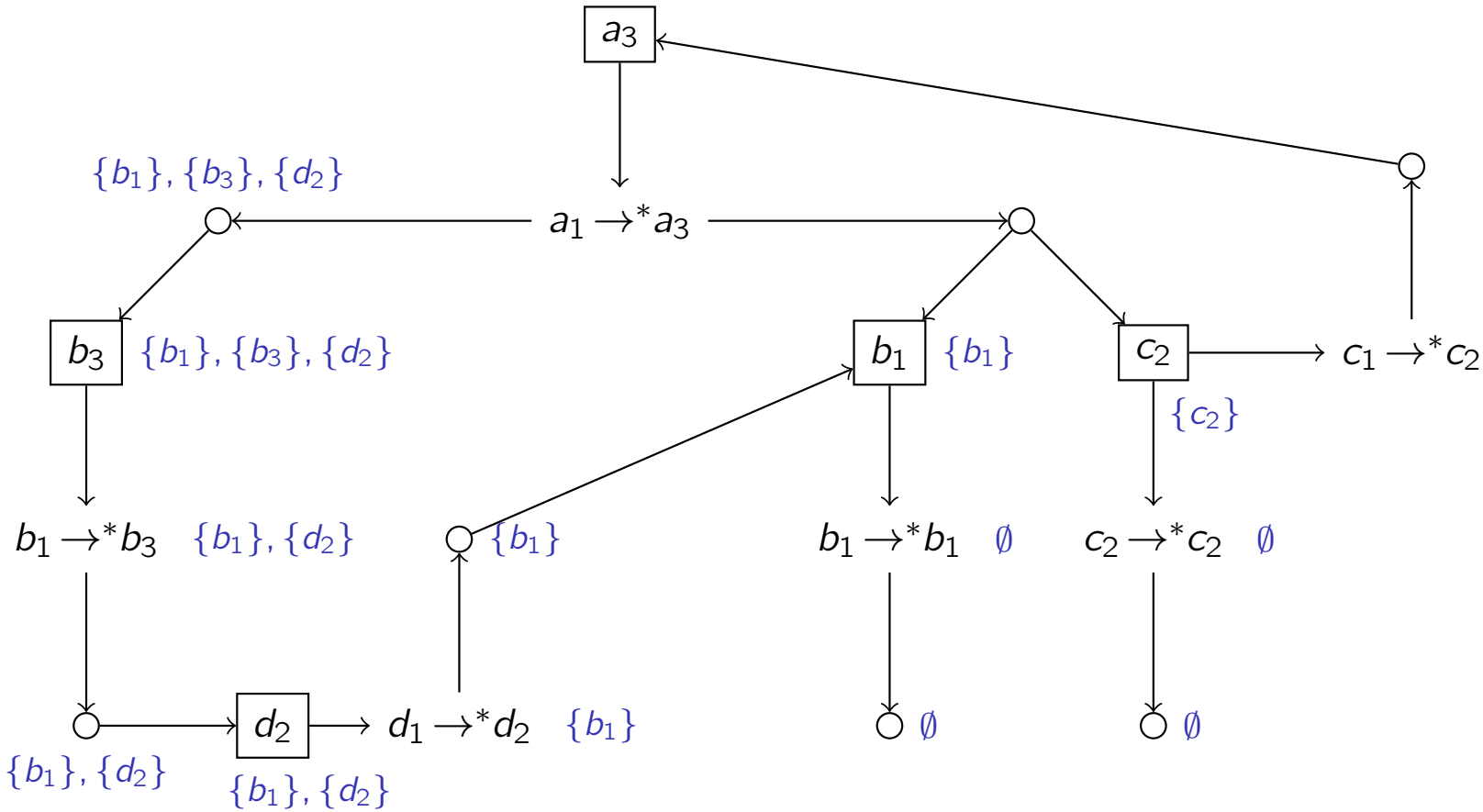


Cut Sets Under-approximation

Example

Sketch

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

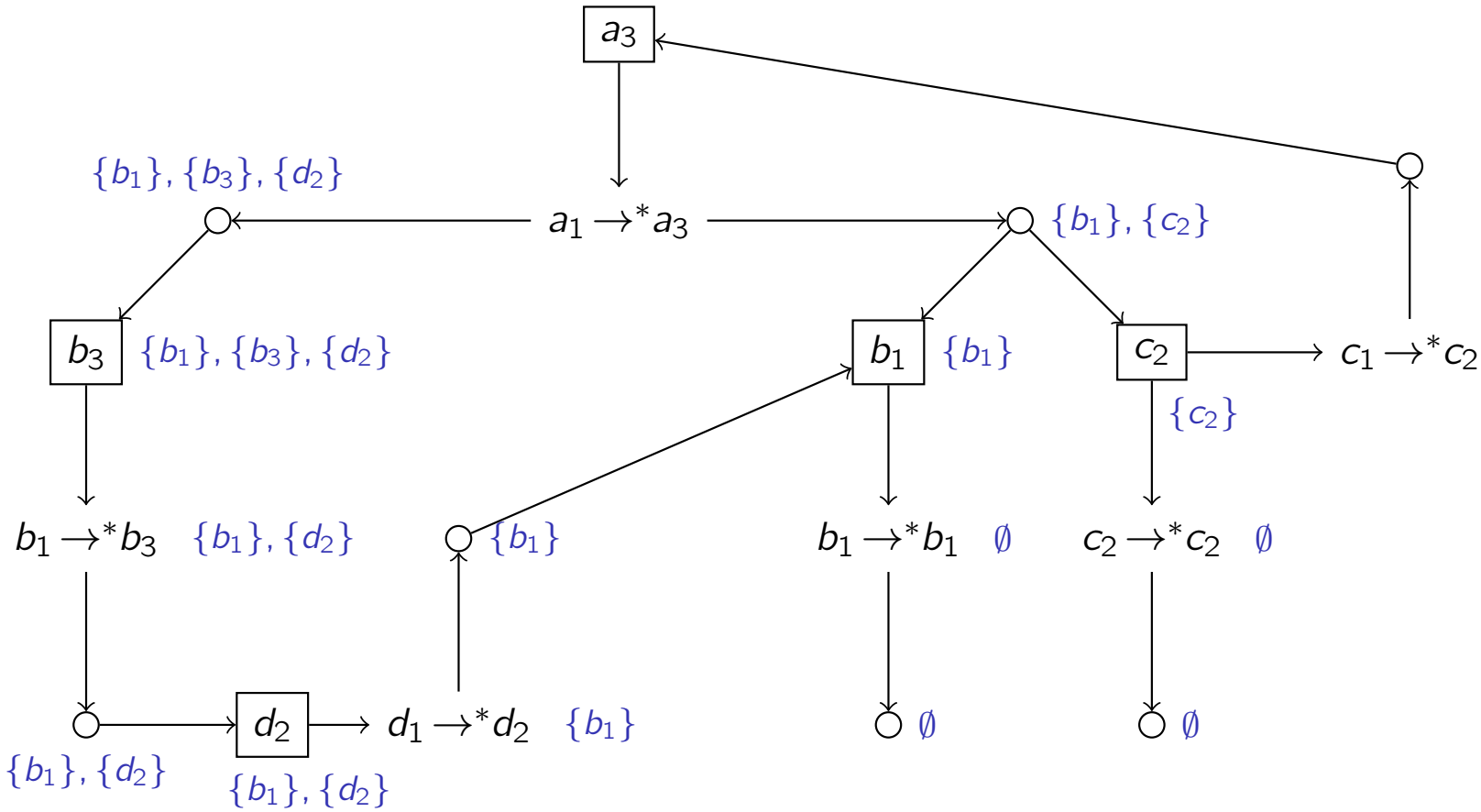


Cut Sets Under-approximation

Example

Sketch

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

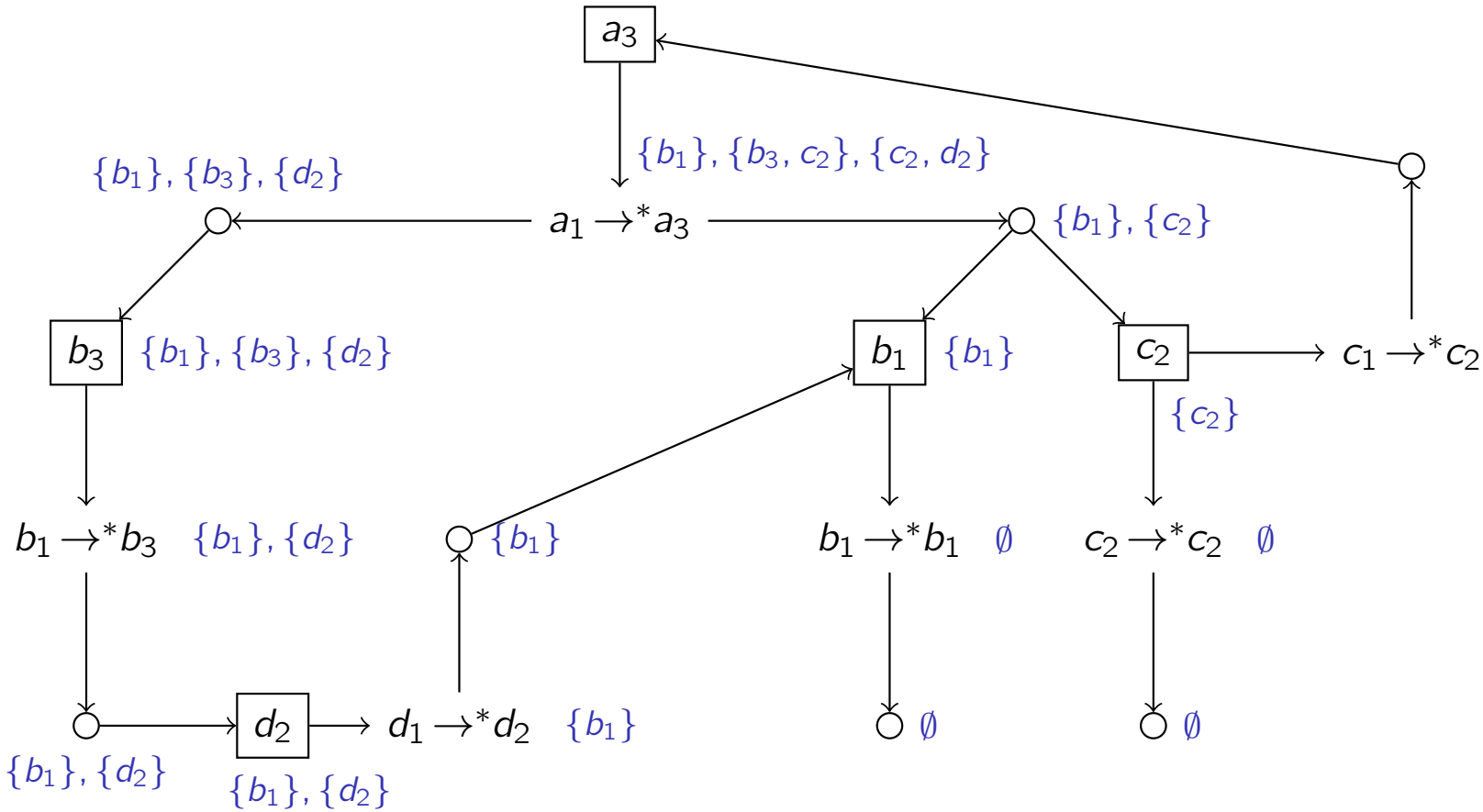


Cut Sets Under-approximation

Example

Sketch

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

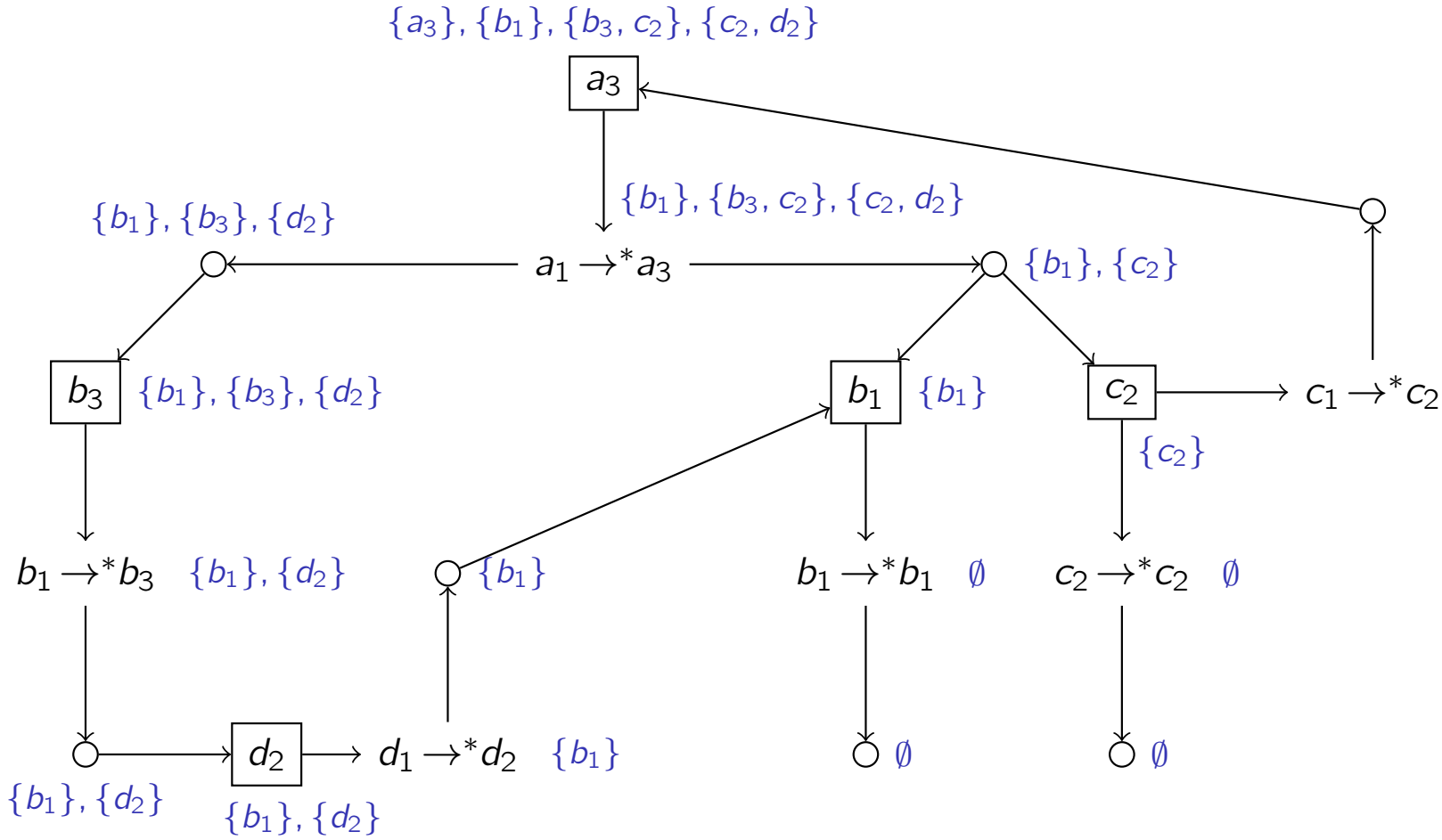


Cut Sets Under-approximation

Example

Sketch

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

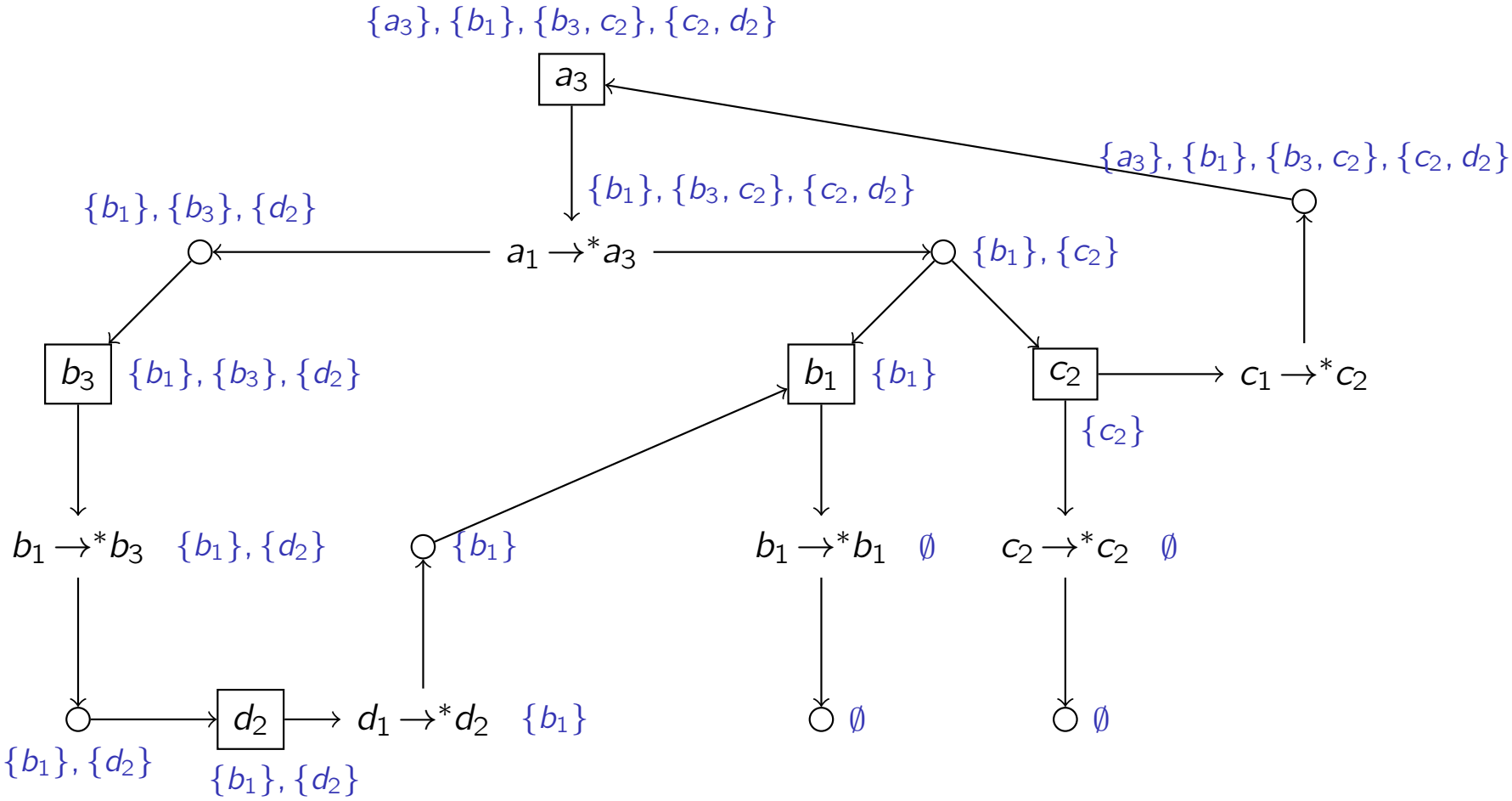


Cut Sets Under-approximation

Example

Sketch

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

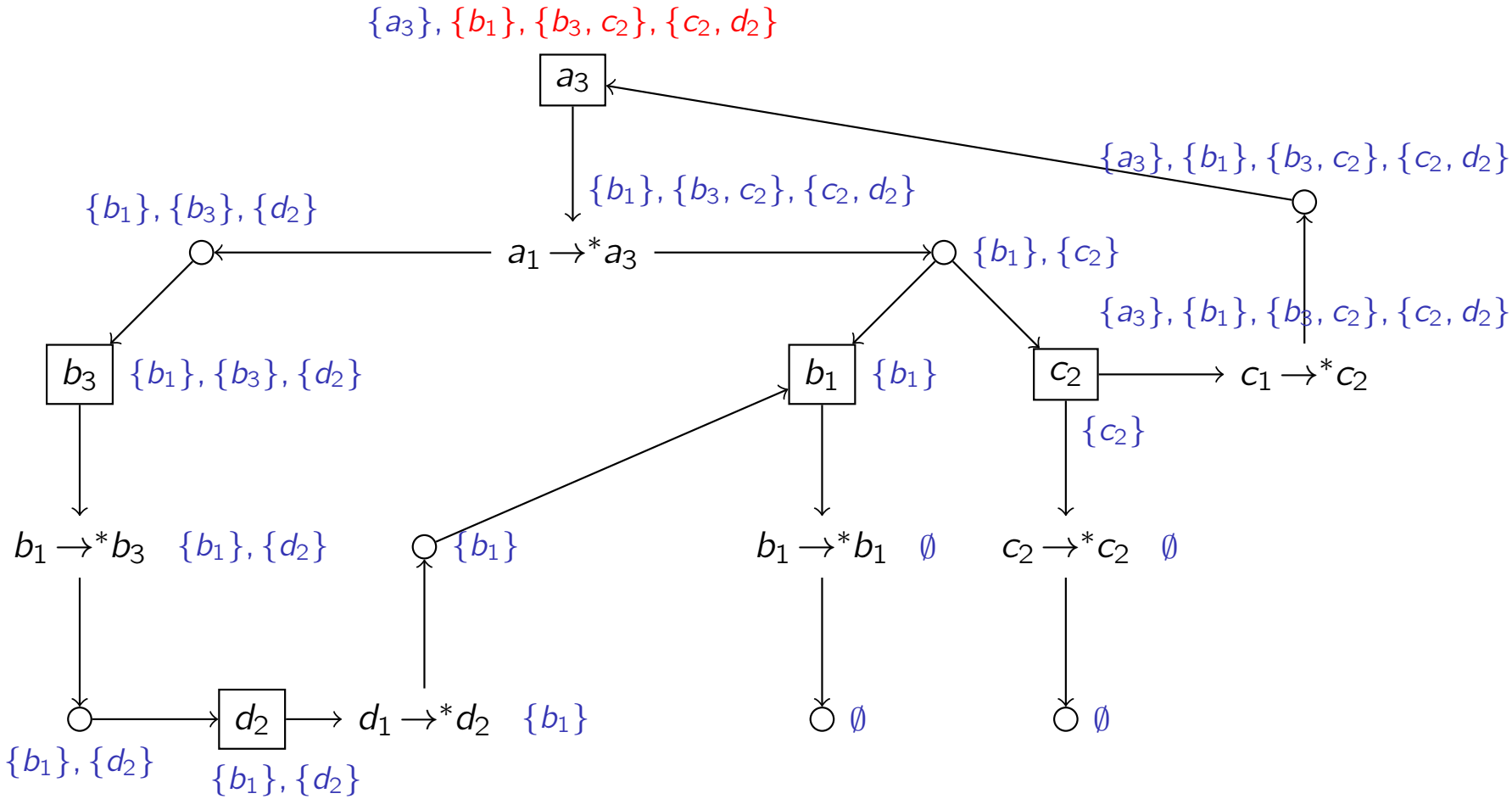


Cut Sets Under-approximation

Example

Sketch

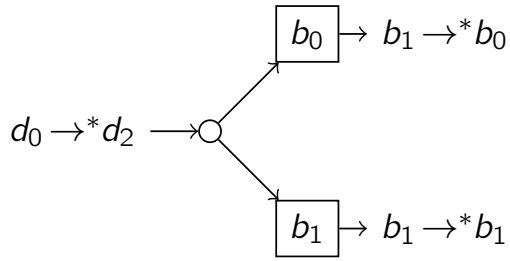
- Follow the topological order of LCG.
- SCCs: arbitrary/random order for updating nodes having child modified.
- Always converges.



Outline

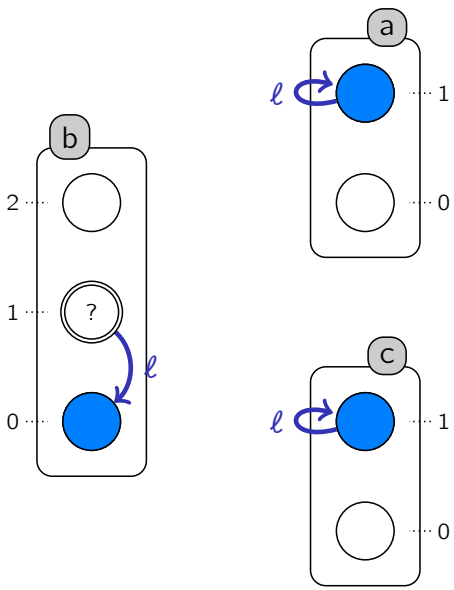
Approach for sufficient conditions

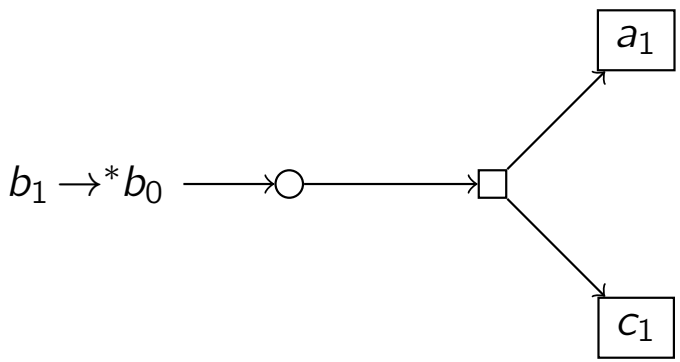
Over-approximation gives:



We want to ensure

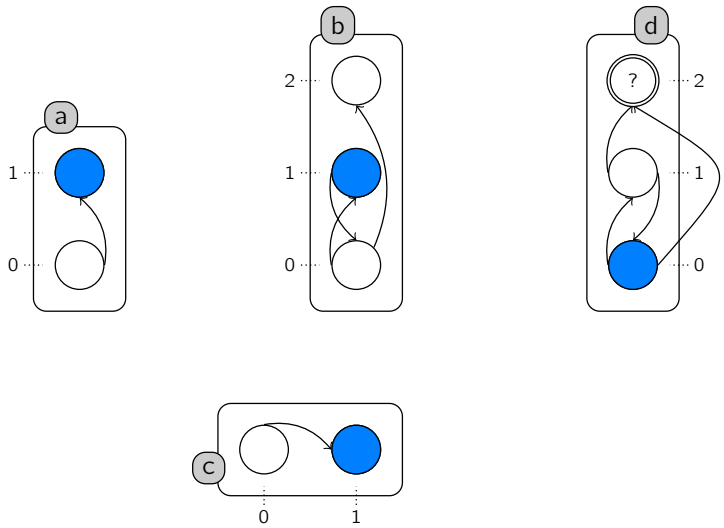
- Insensitivity to reachability order (e.g., $b1$ is reachable from b_0).
  $\Rightarrow$ saturate context of LCG.
- If transition arity > 2, absence of conflicts [Folschette et al. at CS2Bio'13].





Sufficient condition for reachability

Example

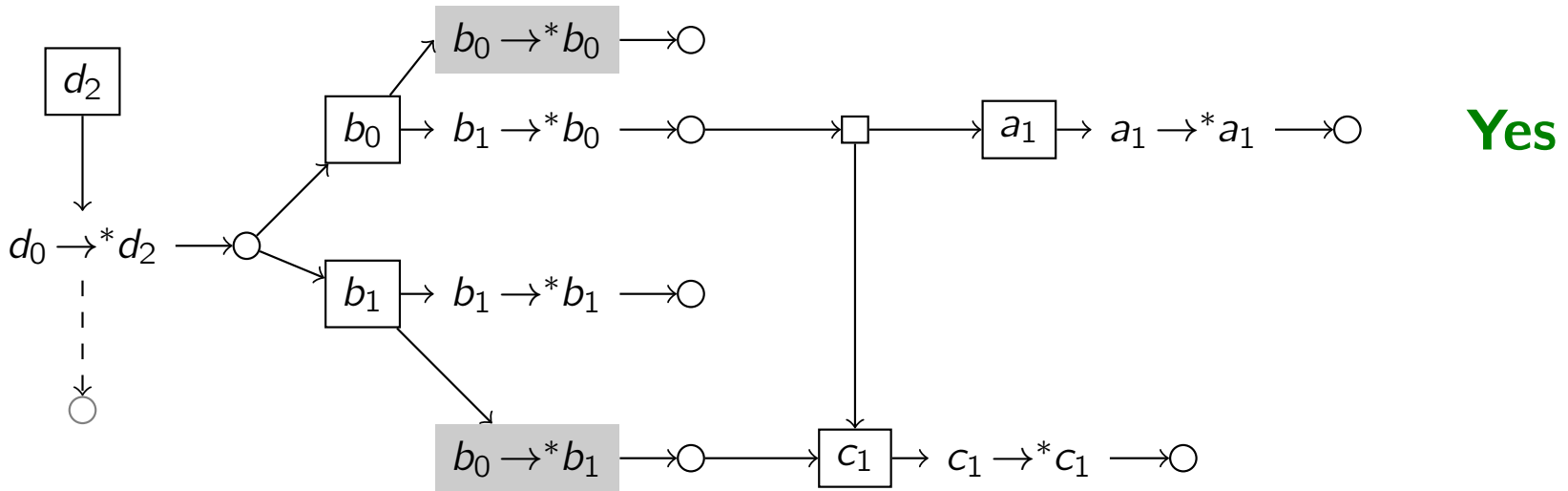


Scope

Asynchronous ANs

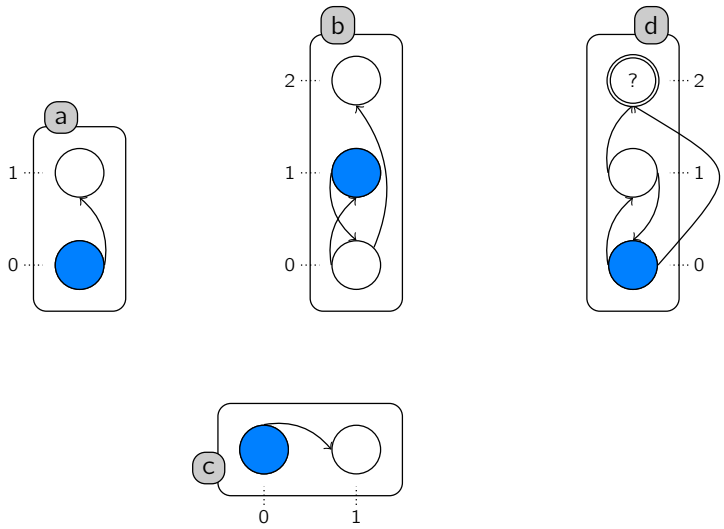
Sufficient condition for $\gamma_s(d_2) \neq \emptyset$:

- LCG' has no cycle;
- each objective has at least one solution.
- local states of a same pre-condition have no conflicts.



Sufficient condition for reachability

Example

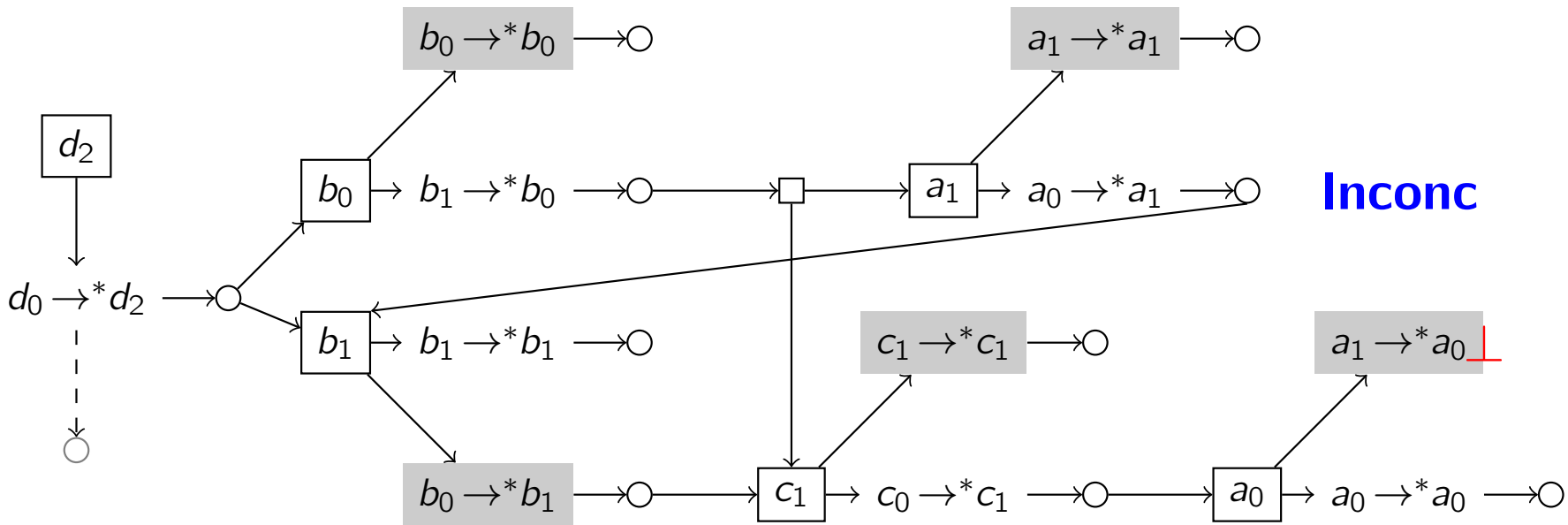


Scope

Asynchronous ANs

Sufficient condition for $\gamma_s(d_2) \neq \emptyset$:

- LCG' has no cycle;
- each objective has at least one solution.
- local states of a same pre-condition have no conflicts.

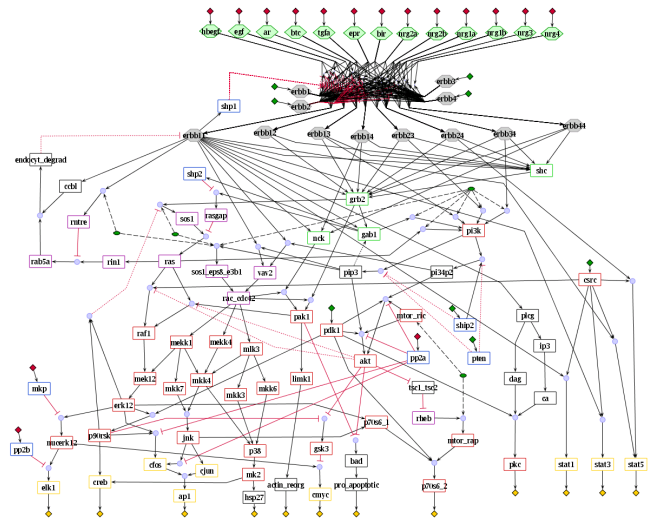


Outline

Experiments in Large Biological Networks

- Signalling networks.
- Wide-range of reachability properties.
- Always conclusive.

Model	NuSMV[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://nusmv.fbk.eu`
[2] `http://move.lip6.fr/software/DDD`
[3] `http://loicpauleve.name/pint`

Cut sets in the whole PID database

Pathway Interaction Database

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

Local Causality Graph

- From AAN 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

Pint Software

http://loicpauleve.name/pint

Pint

- Textual language for Asynchronous Automata Network
- OCaml library + command-line tools for analysis.

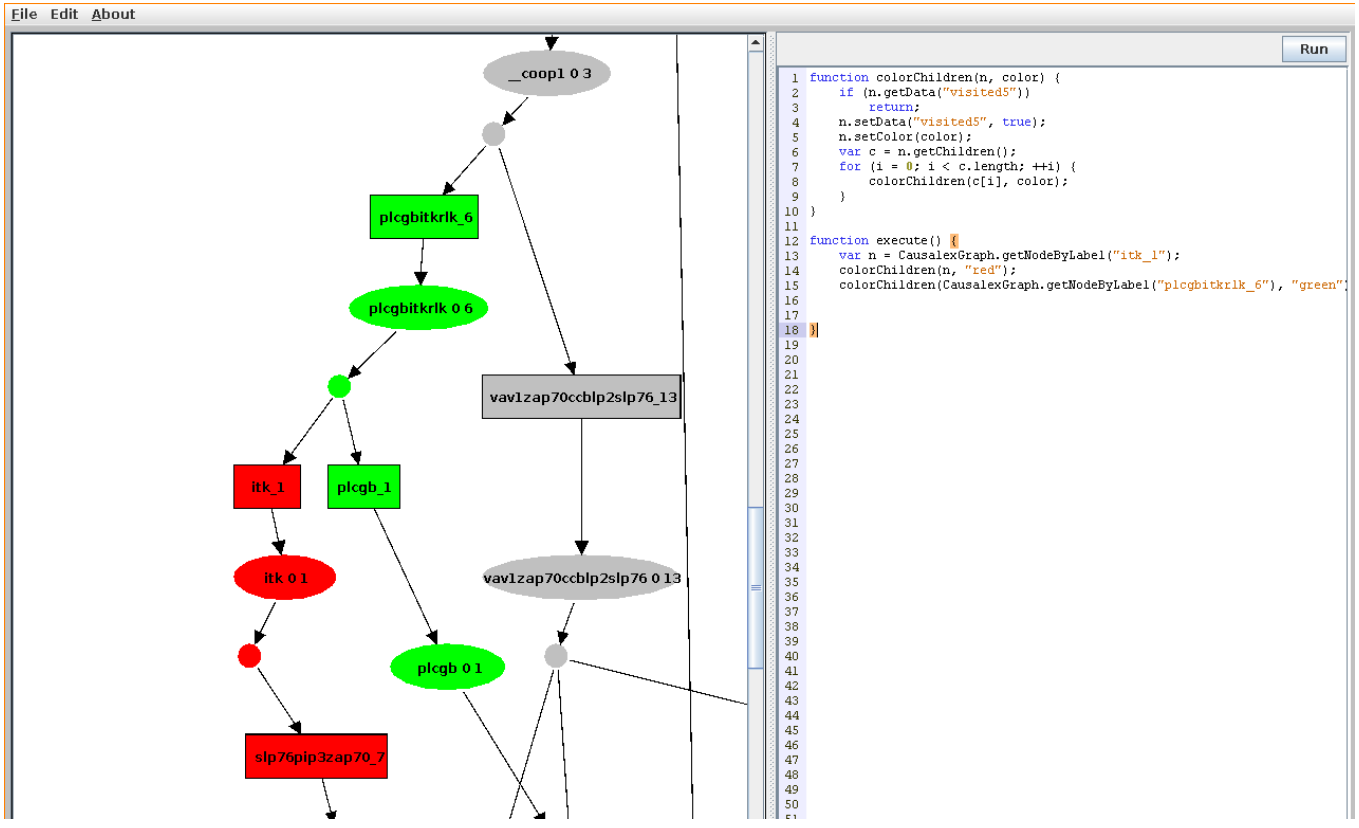
Main features

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

Graphical interfaces start to come out. . .

CausalEx Software

Available on request



Graphical interface for exploring Local Causality Graphs

- Navigation
- Interactive scripting (javascript)
- Algorithm visualization

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

Outline

Summary

Local Causality Graph (LCG)

- Abstract representation of the traces of ANs.
- Compact: poly(nb. automata), exp(size of 1 automaton).
- Exploits concurrency and causality.

Static analysis using LCG

- Necessary conditions for reachability in ANs and AANs.
- Sufficient conditions for reachability in AANs.
- Under-approximation of cut sets for reachability in ANs.

Applications in Systems Biology

- Approach by over-approximation fits well with the typical knowledge (supports incomplete knowledge on parameters, etc.)
- .. and questions, notably with cut sets (mutation prediction).
- Allow to address very large networks (detailed models, databases, ..)
- Gives guaranteed results: may be used to refute models.
- LCG gives comprehensive proofs: points out what is missing/wrong for satisfying a property.

Directions

More properties from the Local Causality Graph

- delimit complex attractors.
- time scales (priorities between transitions).
- cut sets that conserve a given property.

Model reduction w.r.t. reachability property

- From LCG, extract a sub-set of transitions that..
- guarantee to include all minimal traces satisfying given property.
- → *goal-driven* model-checking / simulation.

Related approach: **Petri net unfolding**

- Acyclic net that does not suffer from transitions interleaving
- Finite complete prefix (cut-offs): contains all traces
- . . . but may explode in size.
- Improve unfoldings in the case of Boolean networks?
- Mix LCG with Petri net unfoldings?

⇒ towards a scalable theory of causality and concurrency.

Thank you for your attention.