IEEE International Conference on

Bioinformatics and Biomedecine 2015

Exhaustive analysis of dynamical properties of Biological Regulatory Networks with Answer Set Programming

Emna BEN ABDALLAH

joint work with:

Maxime FOLSCHETTE, Olivier ROUX, Morgan MAGNIN

MeForBio / IRCCyN / École Centrale de Nantes (Nantes, France)
emna.ben-abdallah@irccyn.ec-nantes.fr

2015/11/11

Motivation

Motivation

- **Problem Statement:**
  Analysis of large Biological Regulatory Networks (BRNs) increases complexity therefore it is necessary to propose a new approach to bypass complexity.

Motivation

- **Problem Statement:**
  Analysis of large Biological Regulatory Networks (BRNs) increases complexity therefore it is necessary to propose a new approach to bypass complexity.
- **Aims:**
  - Exhaustive analysis of BRNs:
    - Simulation
    - Steady states
    - Reachability (all/shortest path(s))
  - Decrease the computational complexity
  - Framework:
    - Process Hitting: a new developed model
    - Answer Set Programming: logic programming

Plan

1) Process Hitting (PH)
   - Definition
   - PH through ASP
2) Answer set programming (ASP)
3) Fixed point enumeration
   - Definition
   - ASP implementation
4) Dynamical analysis
   - Network simulation
   - Reachability verification using ASP
5) Conclusion & prospect

The Process Hitting modeling

Sorts: biological components a, b, z

The Process Hitting modeling

Sorts: biological components a, b, z

Processes: local states / levels of expression z_0, z_1, z_2

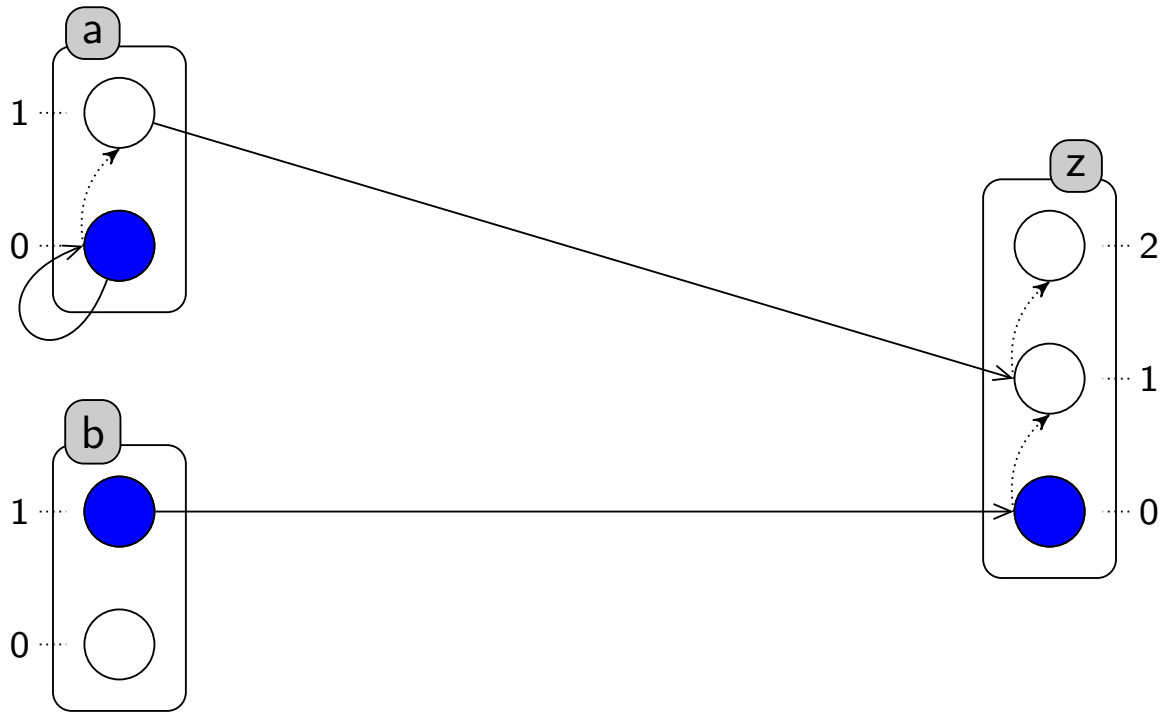
The Process Hitting modeling

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

Processes: local states / levels of expression $z_0,\ z_1,\ z_2$

States: sets of active processes $\langle a_0, b_1, z_0 \rangle$

The Process Hitting modeling



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

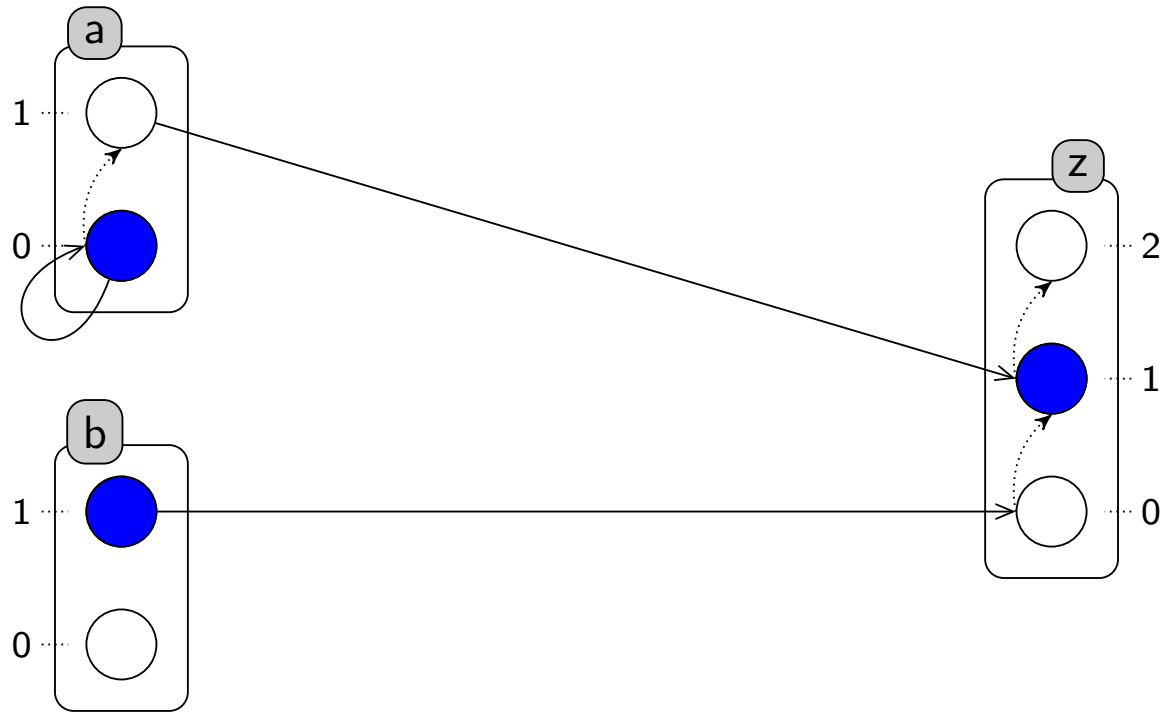
Processes: local states / levels of expression $z_0,\ z_1,\ z_2$

States: sets of active processes $\langle a_0, b_1, z_0 \rangle$

Actions: dynamics $b_1 \rightarrow z_0 \Uparrow z_1,\ a_0 \rightarrow a_0 \Uparrow a_1,\ a_1 \rightarrow z_1 \Uparrow z_2$

$\underline{b_1 \rightarrow z_0 \Uparrow z_1},\ a_0 \rightarrow a_0 \Uparrow a_1,\ a_1 \rightarrow z_1 \Uparrow z_2$

The Process Hitting modeling



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

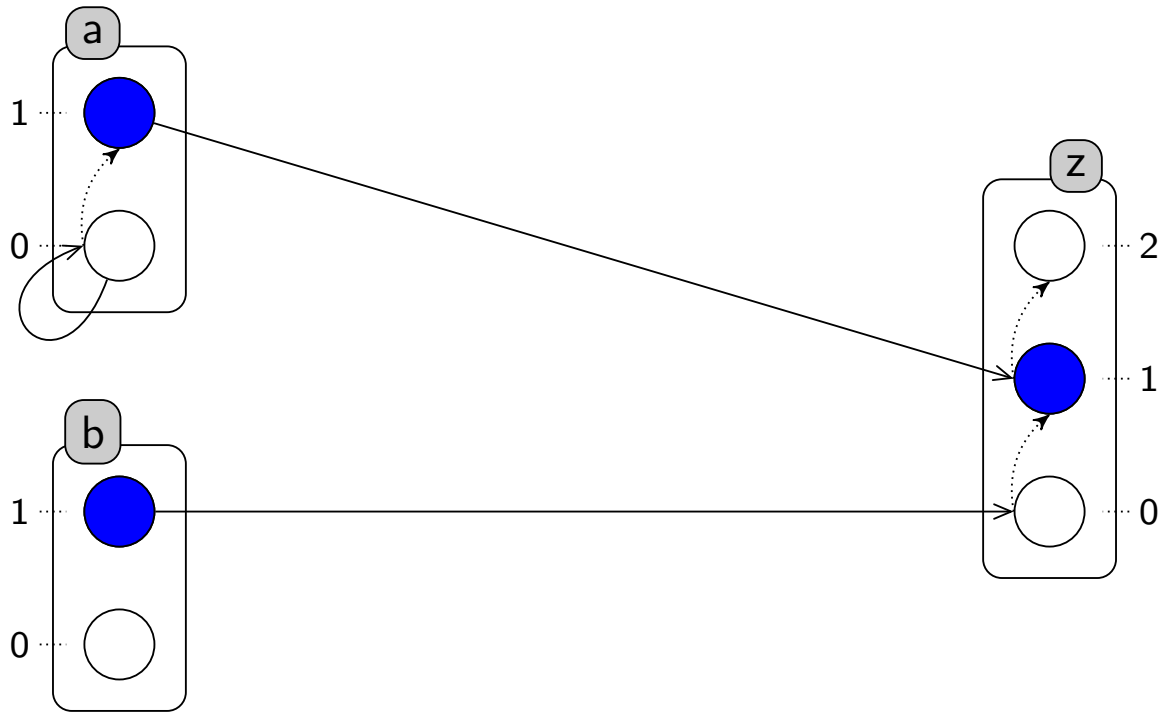
Processes: local states / levels of expression $z_0,\ z_1,\ z_2$

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

Actions: dynamics $b_1 \longrightarrow z_0 \uparrow z_1,\ a_0 \longrightarrow a_0 \uparrow a_1,\ a_1 \longrightarrow z_1 \uparrow z_2$

$b_1 \longrightarrow z_0 \uparrow z_1,\ \underline{a_0 \longrightarrow a_0 \uparrow a_1},\ a_1 \longrightarrow z_1 \uparrow z_2$

The Process Hitting modeling



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

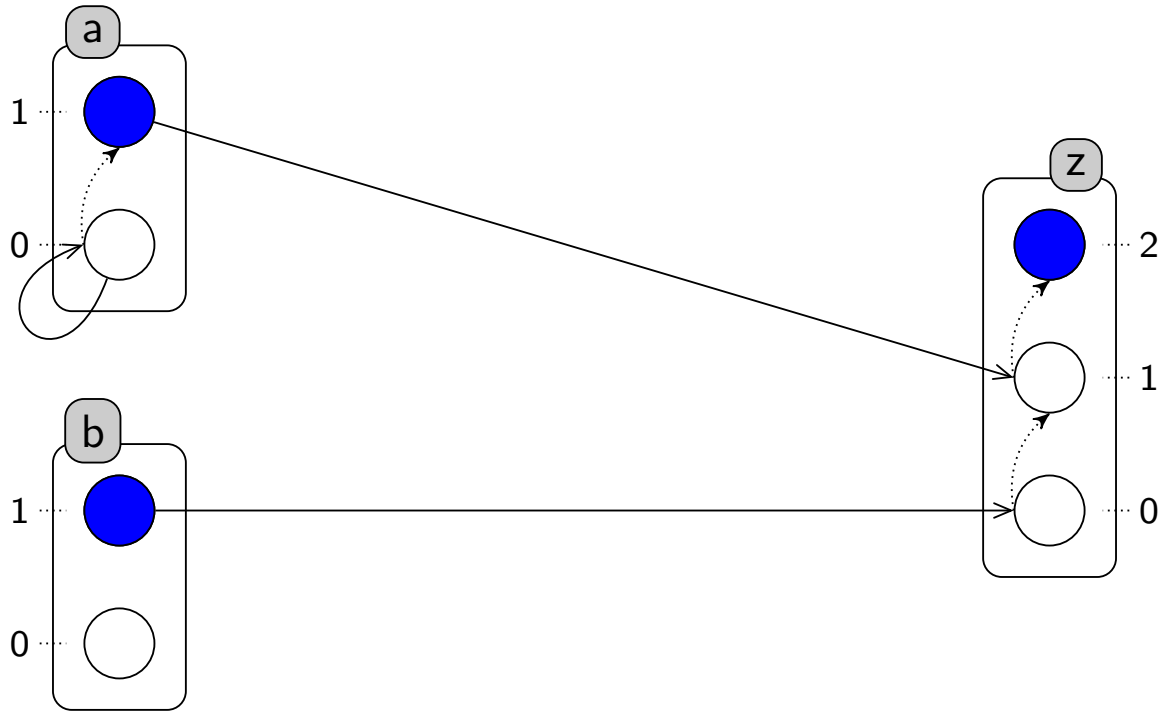
Processes: local states / levels of expression $z_0,\ z_1,\ z_2$

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

Actions: dynamics $b_1 \longrightarrow z_0 \uparrow z_1,\ a_0 \longrightarrow a_0 \uparrow a_1,\ a_1 \longrightarrow z_1 \uparrow z_2$

$b_1 \longrightarrow z_0 \uparrow z_1,\ a_0 \longrightarrow a_0 \uparrow a_1,\ \underline{a_1 \longrightarrow z_1 \uparrow z_2}$

The Process Hitting modeling



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

Processes: local states / levels of expression $z_0,\ z_1,\ z_2$

States: sets of active processes $\langle a_1, b_1, z_2 \rangle$

Actions: dynamics $b_1 \longrightarrow z_0 \restriction z_1,\ a_0 \longrightarrow a_0 \restriction a_1,\ a_1 \longrightarrow z_1 \restriction z_2$

$b_1 \longrightarrow z_0 \restriction z_1,\ a_0 \longrightarrow a_0 \restriction a_1,\ a_1 \longrightarrow z_1 \restriction z_2$

Answer Set Programming

ASP:

- Logic program written in language of AnsProlog*
- Form of rules :

$$head \leftarrow body.$$
$$L_0 \leftarrow L_1, ..., L_m, \textbf{ not } L_{m+1}, ..., \textbf{ not } L_n.$$

with each L_i : literal in the sense of classical logic.

Rule's meaning:

If $L_1, ..., L_m$ are **true** and if $L_{m+1}, ..., L_n$ are **false**
then L_0 is **true**.

Answer Set Programming

Special types of rules:

- **Constraint** :

$$\leftarrow L_1, ..., L_m, \textbf{ not } L_{m+1}, ..., \textbf{ not } L_n.$$

- **Fact** :

$$L_0.$$

- **Cardinality** :

$$min\{L_0, ..., L_j\}max \leftarrow L_1, ..., L_m, \textbf{ not } L_{m+1}, ..., \textbf{ not } L_n.$$

The Process Hitting modeling

PH through ASP

Network traduction:

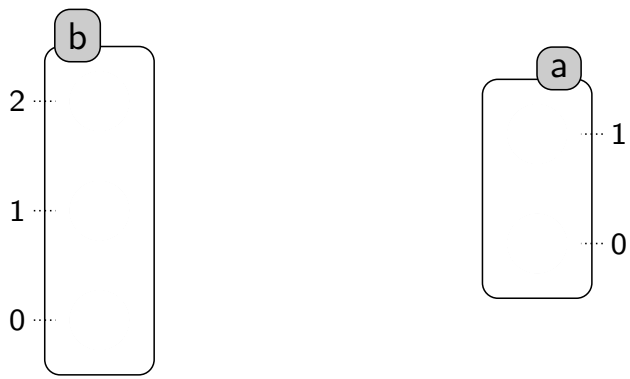
The Process Hitting modeling

PH through ASP

Network traduction:

- **Sort**: `sort(A) .`

Example :




```
sort(X) :- process(X,I).
```

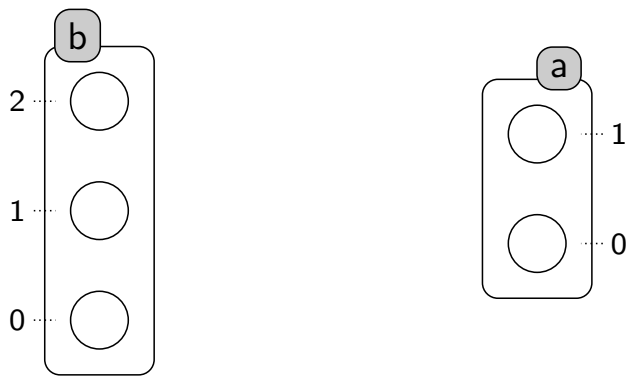
The Process Hitting modeling

PH through ASP

Network traduction:

- **Sort**: `sort(A) .`
- **Process**: `process(A,I).`

Example :




```
process("a", 0..1).
process("b", 0..2).
sort(X) :- process(X,I).
```

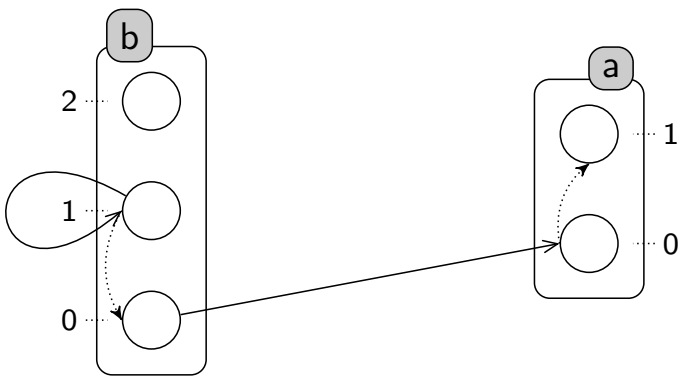
The Process Hitting modeling

PH through ASP

Network traduction:

- **Sort**: `sort(A) .`
- **Process**: `process(A,I).`
- **Action** $a_i \rightarrow b_j \not\uparrow b_k$: `action(A,I,B,J,K).`

Example :



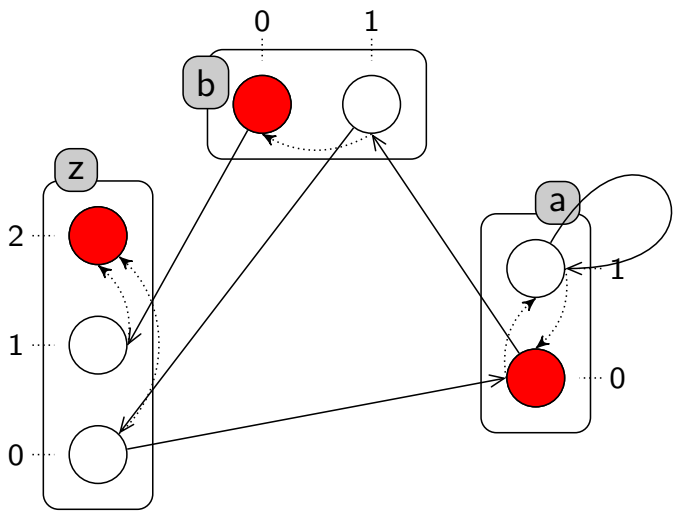

```
process("a", 0..1).
process("b", 0..2).
sort(X) :- process(X,I).
action("b",0,"a",0,1).
action("b",1,"b",1,0).
```

Static Analysis

Fixed Point

Definition:

state where:
- no model evolution is possible
- no action can be played

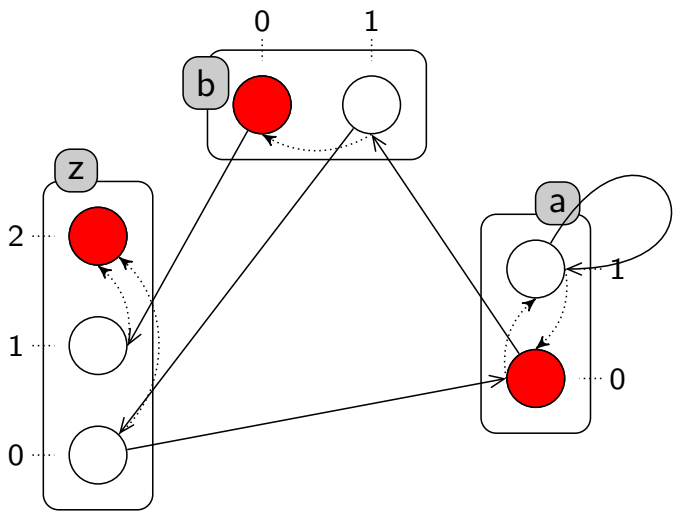


Static Analysis

Fixed Point through ASP

Search of fixed points:

```
hiddenProcess(A,I) :- action(A,I,A,I,K).
shownProcess(A,I) :- not hiddenProcess(A,I), process(A,I).
1 { selectProcess(A,I) : shownProcess(A,I) } 1 :- sort(A).
:- action(A,I,B,_), selectProcess(A,I), selectProcess(B,J), A!=B.
```

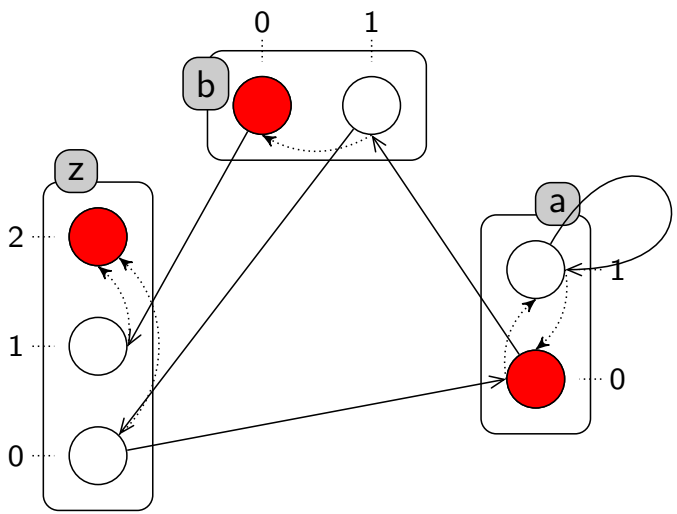


Static Analysis

Fixed Point through ASP

ASP program result:

```
Answer 1:  fixProcess(a,0), fixProcess(b,0), fixProcess(z,2).
```



Static analysis

Fixed Point

Comparison

Model	#sorts	#states	#fix-point	ASP-PH	PINT-PH
mvbrn	3	12	1	0.000s	0.006s
ERBB	42	2^{70}	3	0.000s	0.017s
tcrsig40	54	2^{73}	1	0.020s	0.021s
tcrsig94	133	2^{194}	0	0.060s	0.027s
egfr104	193	2^{320}	0	0.140s	0.074s

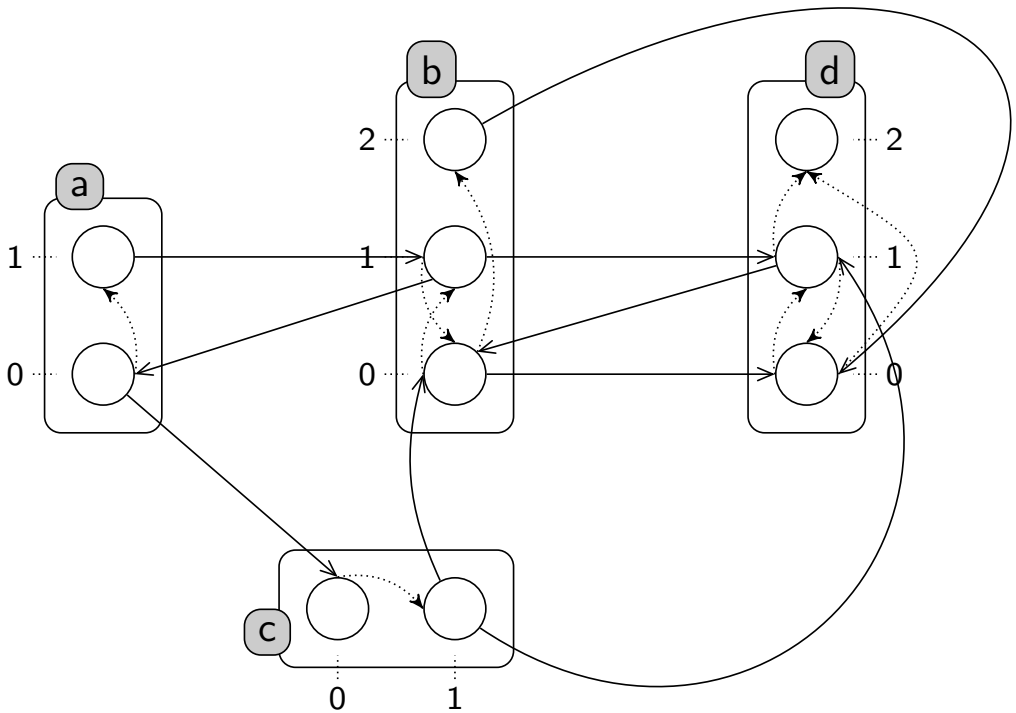
Figure: Excecution time of ASP metod and PINT applied for biological networks with a desktop computer (core i5 and 4GB RAM).

PINT: a library developed to parse and study PH models.

Dynamic analysis

Reachability

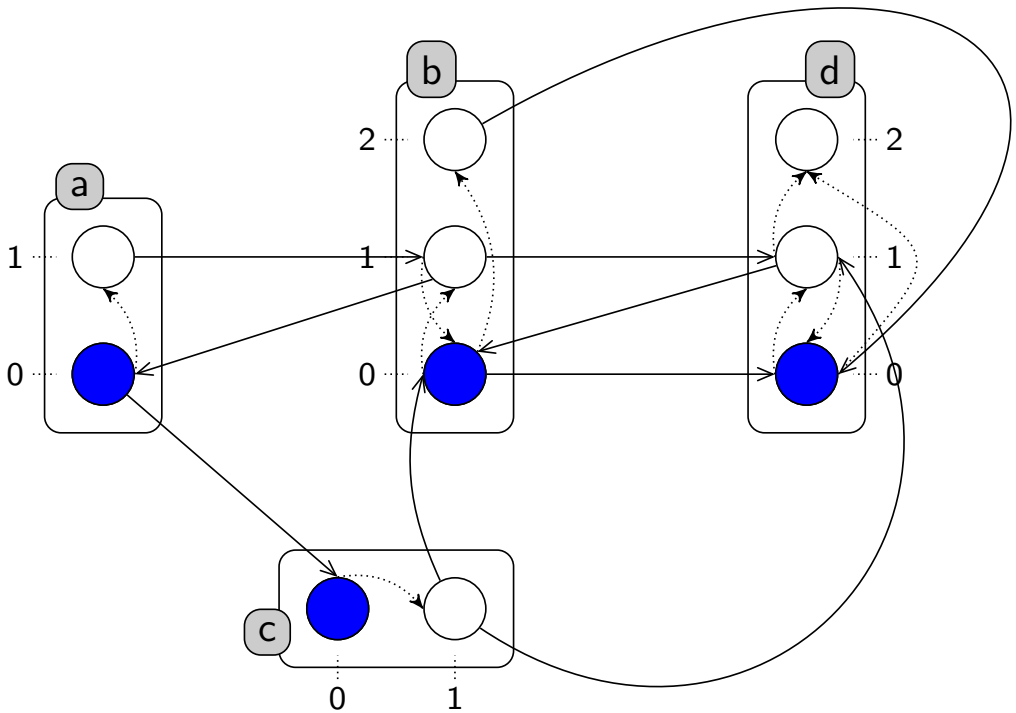
Reachability of processes for PH models:



Dynamic analysis

Reachability

Reachability of processes for PH models:



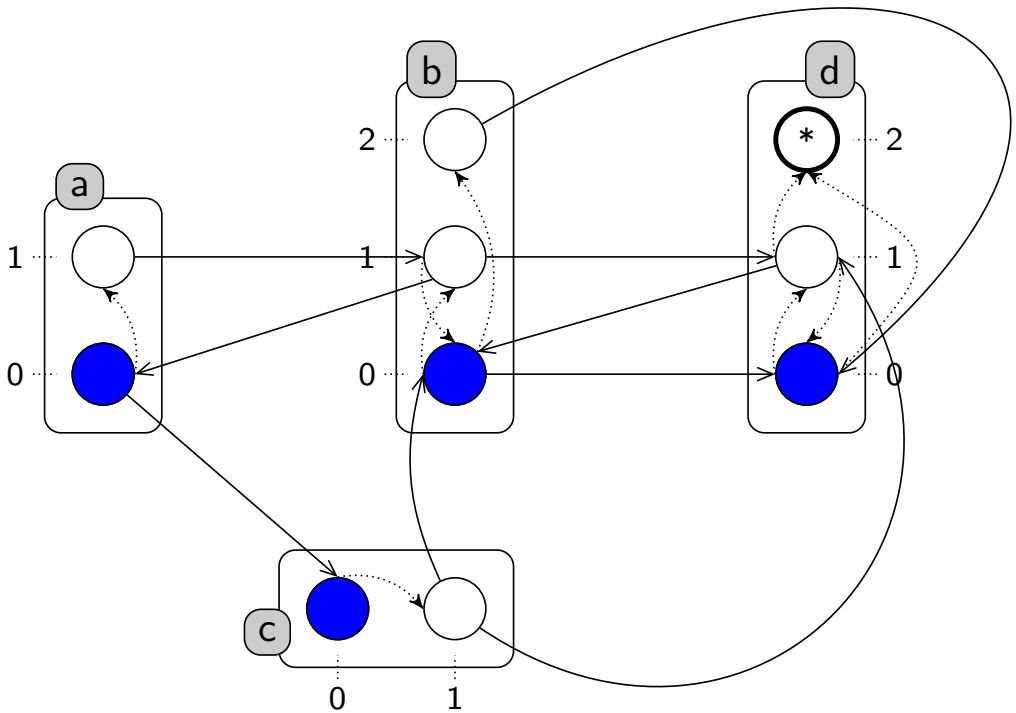
- Initial context

$\langle a_0, b_0, c_0, d_0 \rangle$

Dynamic analysis

Reachability

Reachability of processes for PH models:



- Initial context

$$\langle a_0, b_0, c_0, d_0 \rangle$$

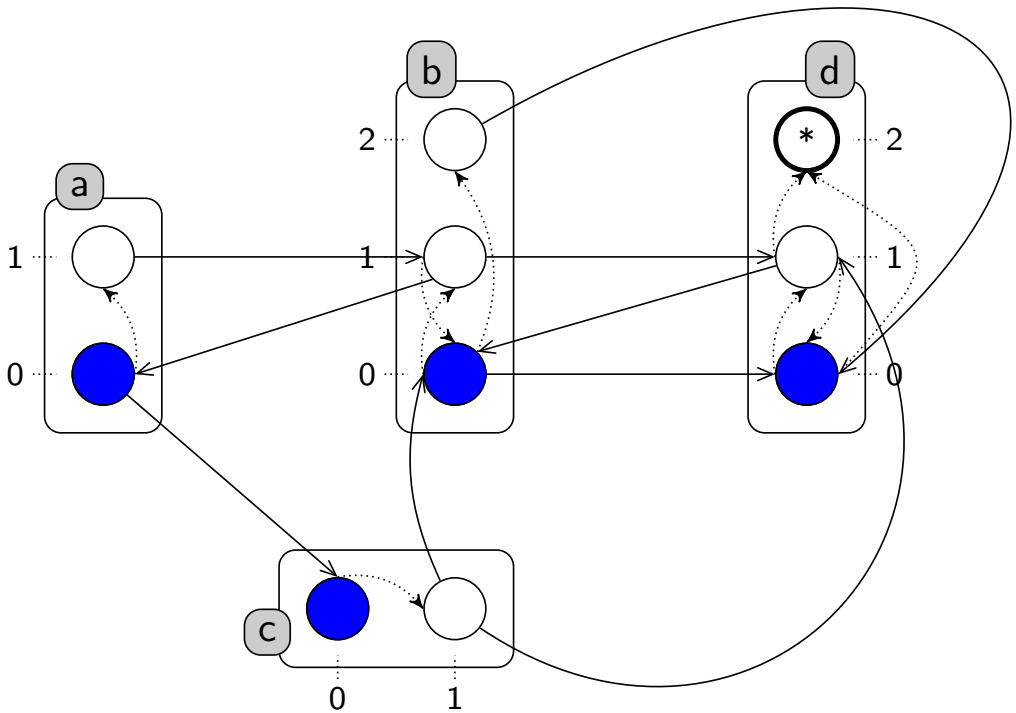
- Objectives

$$[\ \Rsh\ d_2\ ]$$

Dynamic analysis

Reachability

Reachability of processes for PH models:



- Initial context

$$\langle a_0, b_0, c_0, d_0 \rangle$$

- Objectives

$$[\ \Rsh\ d_2\ ]$$

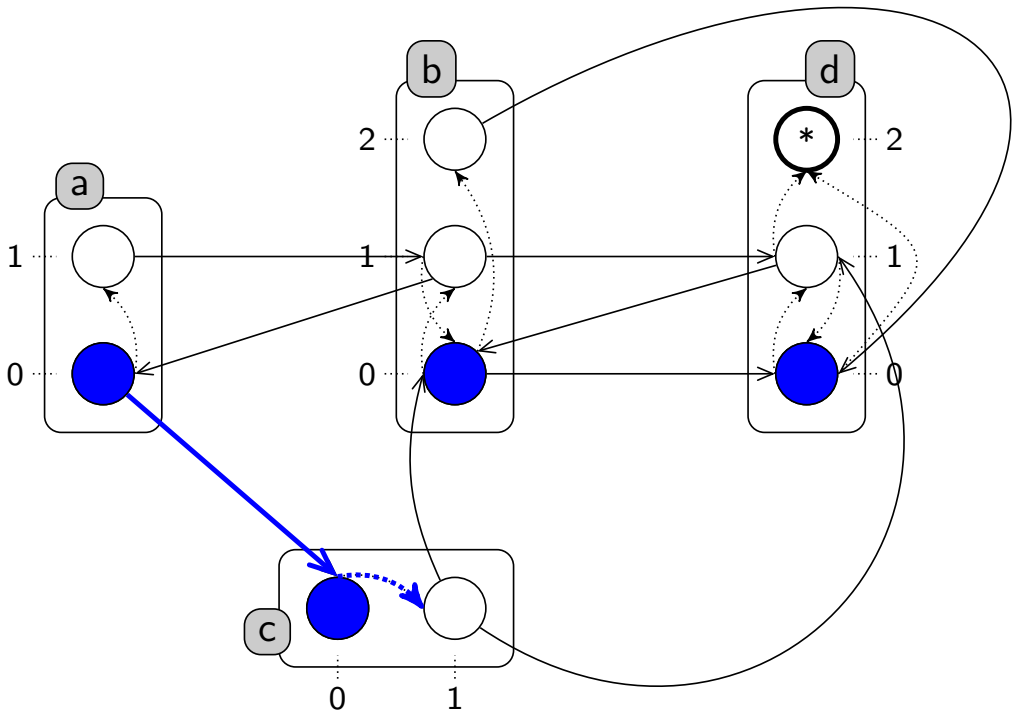
$\rightarrow$ Concretization of the objective = scenario

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

Dynamic analysis

Reachability

Reachability of processes for PH models:



- Initial context

$$\langle a_0, b_0, c_0, d_0 \rangle$$

- Objectives

$$[\ \Rsh\ d_2\ ]$$

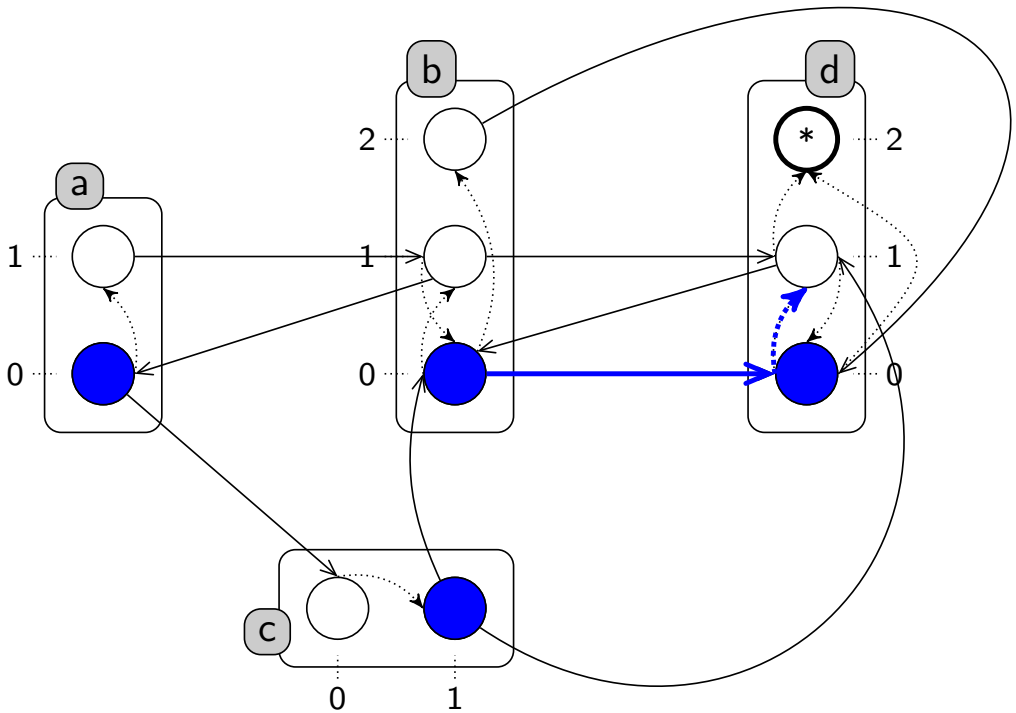
$\rightarrow$ Concretization of the objective = scenario

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

Dynamic analysis

Reachability

Reachability of processes for PH models:



- Initial context

$$\langle a_0, b_0, c_0, d_0 \rangle$$

- Objectives

$$[\ \Rsh\ d_2\ ]$$

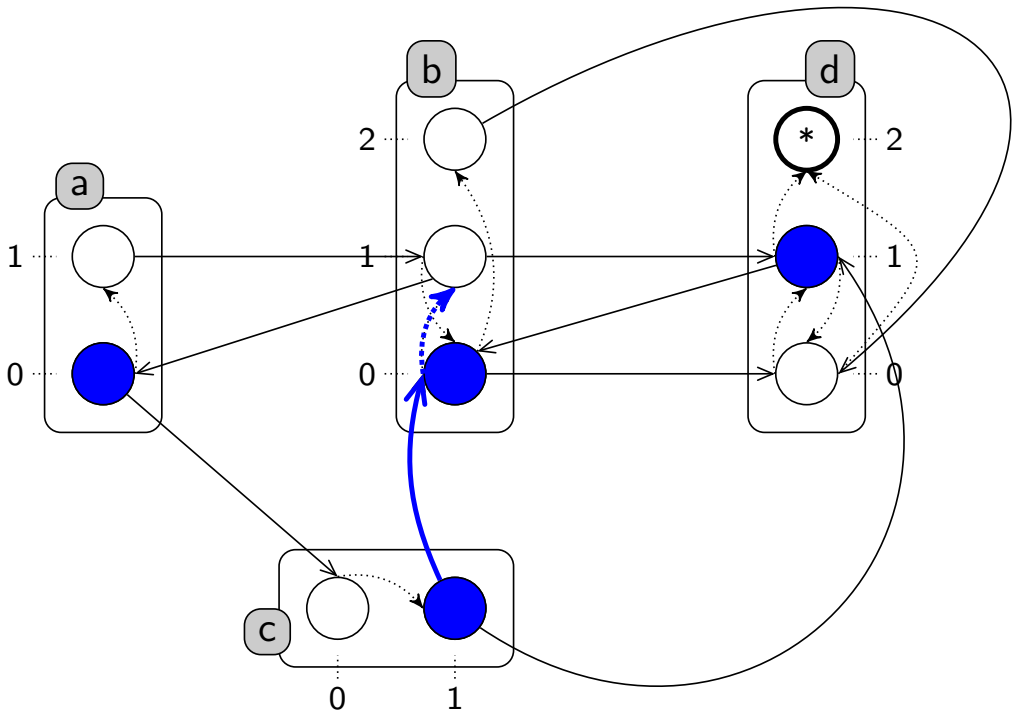
$\rightarrow$ Concretization of the objective = scenario

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

Dynamic analysis

Reachability

Reachability of processes for PH models:



- Initial context

$$\langle a_0, b_0, c_0, d_0 \rangle$$

- Objectives

$$[\ \Rsh\ d_2\ ]$$

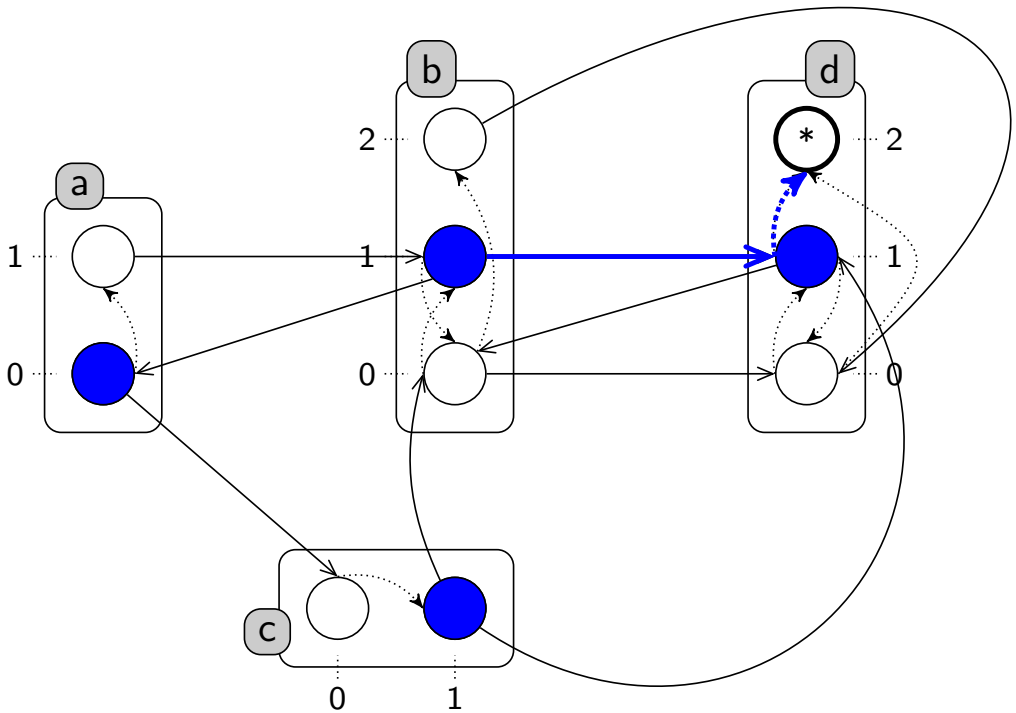
$\rightarrow$ Concretization of the objective = scenario

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

Dynamic analysis

Reachability

Reachability of processes for PH models:



- Initial context

$$\langle a_0, b_0, c_0, d_0 \rangle$$

- Objectives

$$[\ \Rsh\ d_2\ ]$$

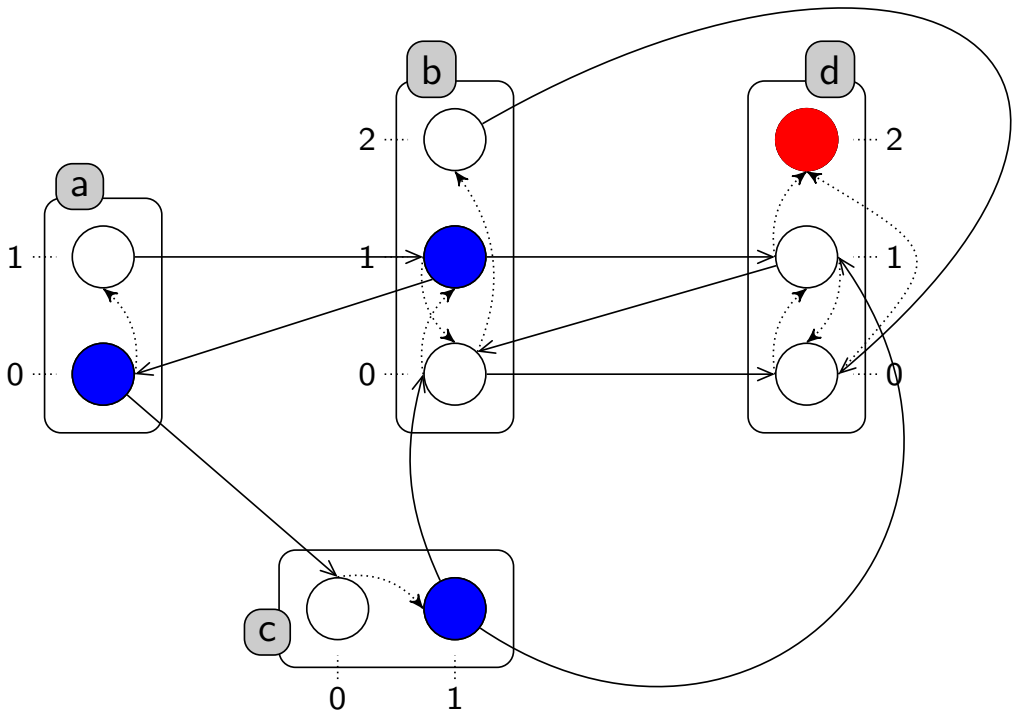
$\rightarrow$ Concretization of the objective = scenario

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

Dynamic analysis

Reachability

Reachability of processes for PH models:



- Initial context

$$\langle a_0, b_0, c_0, d_0 \rangle$$

- Objectives

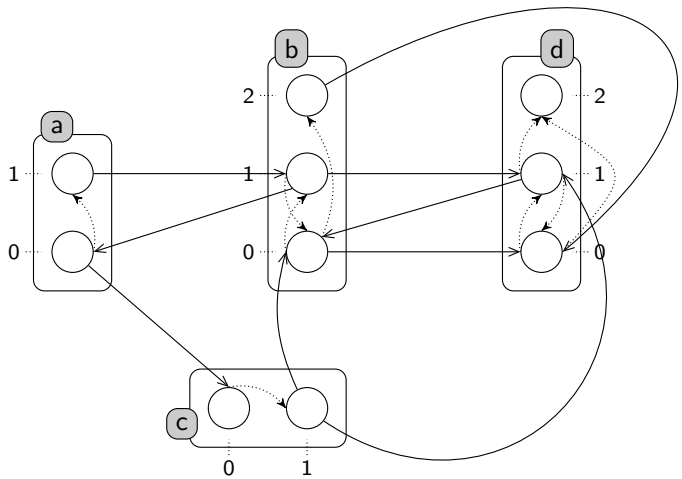
$$[\ \Rsh\ d_2\ ]$$

$\rightarrow$ Concretization of the objective = scenario

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

Dynamic analysis

Network Simulation through ASP

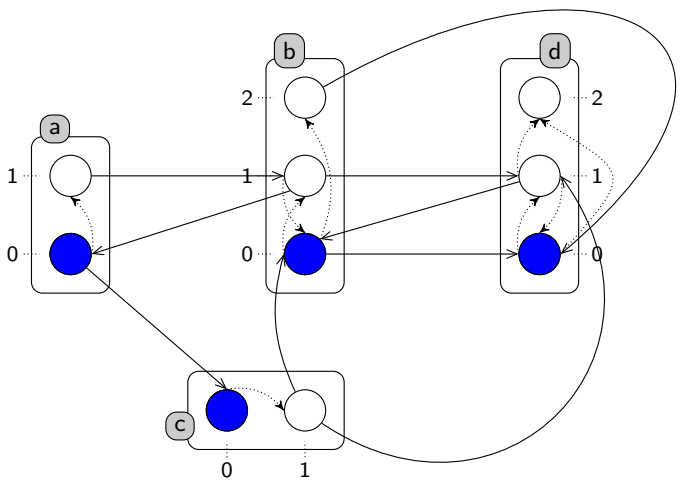


Dynamic analysis

Network Simulation through ASP

Initializing :

`init(activeProcess("a",0)).` with `a:` sort, `0:` index of the process

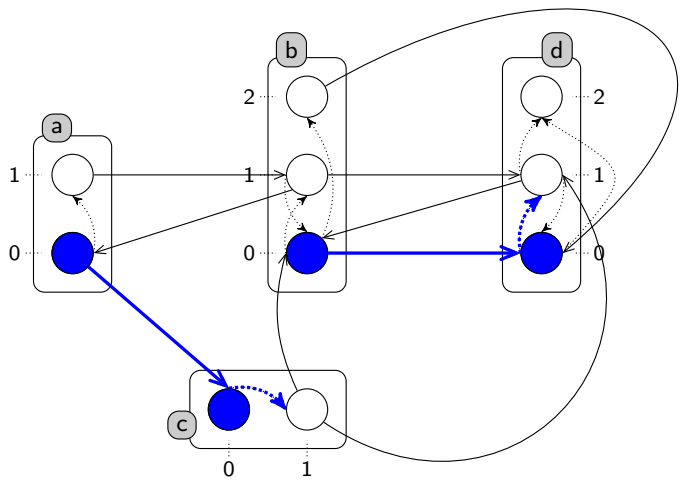


Dynamic analysis

Network Simulation through ASP

Playable actions at step T :

```
playable(action(A,I,B,J,K),T) :- action(A,I,B,J,K),
                                 instate(activeProcess(A,I),T),
                                 instate(activeProcess(B,J),T),
                                 time(T).
```

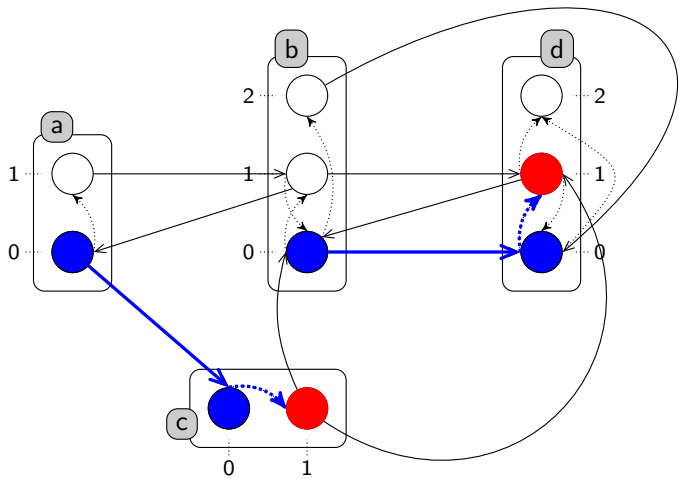


Dynamic analysis

Network Simulation through ASP

Asynchronous evolution:

```
0 {play(Act,T)} 1 :- playable(Act,T), time(T).
:- 2{play(Act,T)}, time(T).
change(B,T+1) :- play(action(_,_,B,_,_),T), time(T).
```

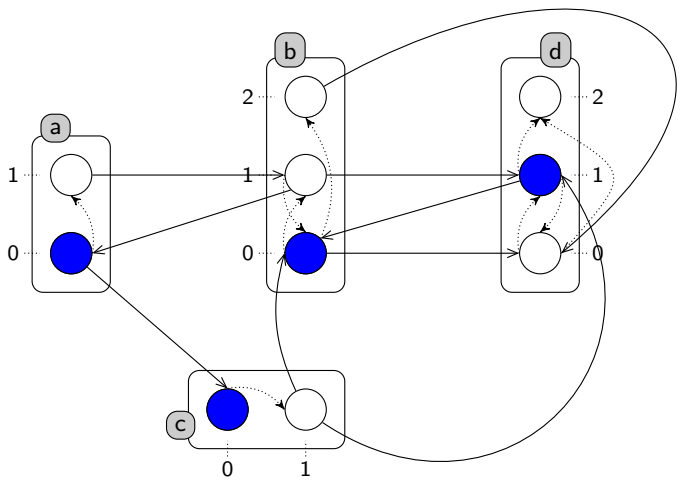


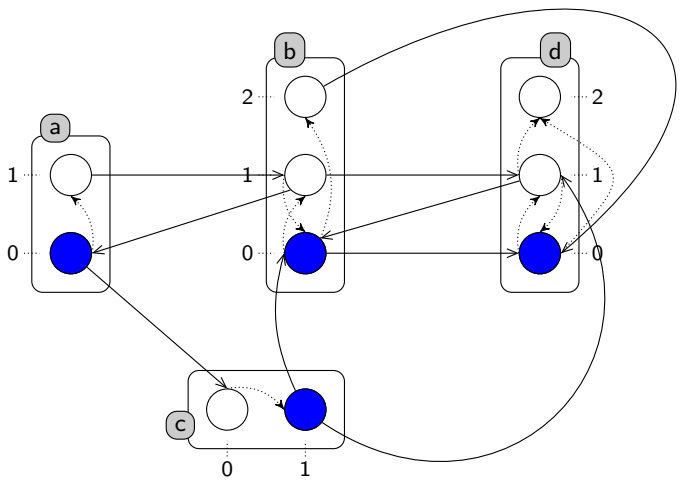
Dynamic analysis

Network Simulation through ASP

Active processes at next step (T+1) :

```
instate(activeProcess(B,K),T+1) :- play(action(_,_,B,_,K),T), time(T).
instate(activeProcess(B,K),T+1) :- not change(B,T+1),
                                   instate(activeProcess(B,K),T), time(T).
```





Dynamic analysis

Network Simulation through ASP

Results ($N = 3$) :

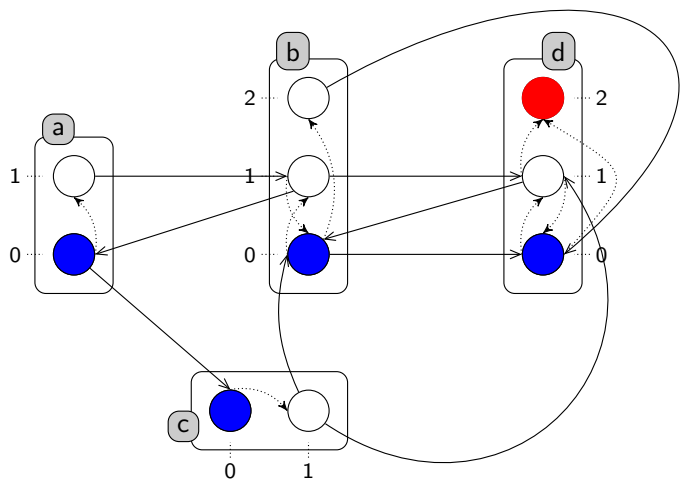
Answer 1: `playable(action("b",0,"d",0,1),0)`
`playable(action("a",0,"c",0,1),1) playable(action("c",1,"b",0,1),2).`
Answer 2: `playable(action("b",0,"d",0,1),0)`
`playable(action("d",1,"b",0,2),1)`
Answer 3: `playable(action("a",0,"c",0,1),0)`
`playable(action("b",0,"d",0,1),1) playable(action("c",1,"d",1,0),2)`
`playable(action("c",1,"b",0,1),3)`
...
Answer 29: `playable(action("a",0,"c",0,1),0)`
`playable(action("c",1,"b",0,1),1) playable(action("b",1,"a",0,1),2)`

Dynamic analysis

Reachability through ASP

Verifying the reachability:

```
goal(activeProcess("d",2)).
reached(APr, T) :- goal(APr), instate(APr, T).
:- not allReached.
```



Dynamic analysis

Reachability through ASP

Results for ($N = 2$) :

```
UNSATISFIABLE
```

Results for ($N = 3$) :

Answer 1: `playable(action("a",0,"c",0,1),0)`
`playable(action("b",0,"d",0,1),1), playable(action("c",1,"b",0,1),2),`
`playable(action("b",1,"d",1,2,3).`

Answer 2: `playable(action("b",0,"d",0,1),0)`
`playable(action("a",0,"c",0,1),1) playable(action("c",1,"b",0,1,2)`
`playable(action("b",1,"d",1,2,3)`

```
SATISFIABLE
```

Dynamic analysis

Reachability through ASP

Results for ($N = 2$) :

```
UNSATISFIABLE
```

Results for ($N = 3$) :

Answer 1: `playable(action("a",0,"c",0,1),0)`
`playable(action("b",0,"d",0,1),1), playable(action("c",1,"b",0,1),2),`
`playable(action("b",1,"d",1,2,3).`
Answer 2: `playable(action("b",0,"d",0,1),0)`
`playable(action("a",0,"c",0,1),1) playable(action("c",1,"b",0,1,2)`
`playable(action("b",1,"d",1,2,3)`

```
SATISFIABLE
```

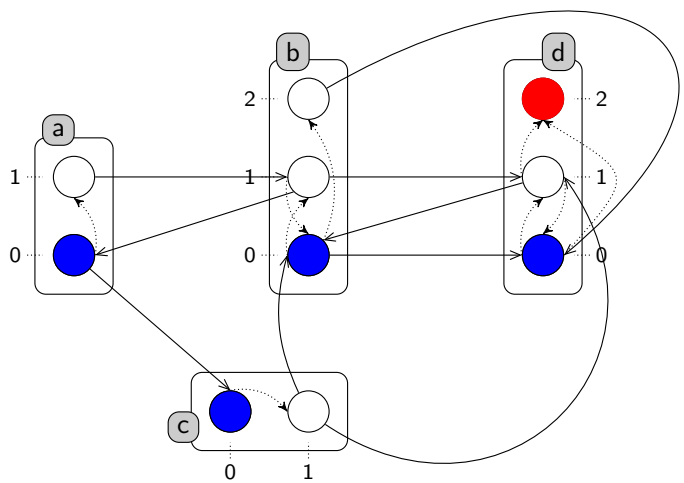
time(0..N). : **predict N –>** **Inconvenient**

Dynamic analysis

Reachability through ASP

Testing reachability (*incrementative ASP*):

```
goal(activeProcess("d",2)).
#program base.
instate(F,0) :- init(F).
#program step(t).
playable(action(A,I,B,J,K),t), play(Act,t),change(B,T+1),
instate(activeProcess(A, I),t + 1)...
#program check(t).
notReached(t) :- goal(F), not instate(F,t).
:- notReached(t).
```



Dynamic analysis

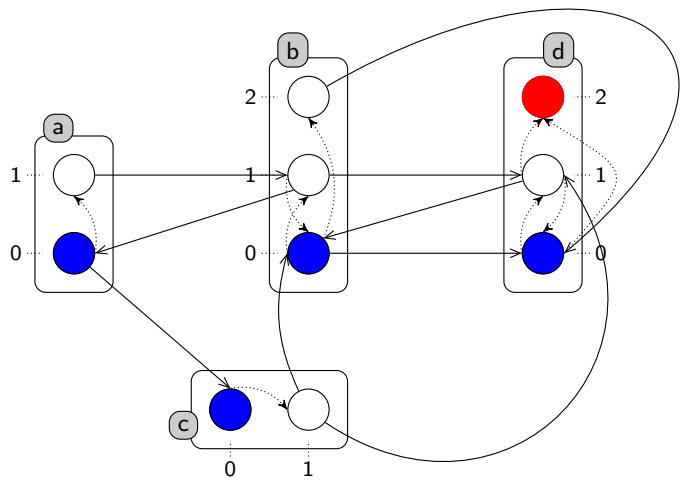
Reachability through ASP

Testing reachability (*ASP incrementative*):

Results:

Answer 1: `playable(action("a",0,"c",0,1),0)`
`playable(action("b",0,"d",0,1),1), playable(action("c",1,"b",0,1),2),`
`playable(action("b",1,"d",1,2,3).`

Answer 2: `playable(action("b",0,"d",0,1),0)`
`playable(action("a",0,"c",0,1),1) playable(action("c",1,"b",0,1,2)`
`playable(action("b",1,"d",1,2,3)`
`SATISFIABLE`



Dynamic analysis

Reachability

Comparison:

Initializing biological models components and the objectives.

Model-target	#sorts	PINT	LIBDDD	GINSIM	ASPI-PH
ERBB-whole	20	out	1m55.38s	2m31.64s	0m11.84s
ERBB-sub	20	0m0.03s	1m54.96s	-	0m5.02s
TCR-whole	40	Inconc	out	out	4m27.93s
TCR-sub	40	0m0.02s	out	-	1m35.08s

Figure: Compared performances of PINT, LIBDDD, GINSIM and our new itirative method ASPI-PH.

PINT:: library developed to parse and study PH models.
LIBDDD: is a C++ library for manipulation of decision diagrams with the construction of state-transition graph.
GINSIM: software for the edition, simulation and analysis of gene interaction networks.

Conclusion & Prospects

- New dynamic analysis of Process Hitting models:
  - Fixed point enumaration
  - Network simulation
  - Reachability verification
- All programs are available online at: `https://github.com/EmnaBenAbdallah/verification-of-dynamical-properties_PH`
- Prospects:
  - Adaptation on extended PH (plural actions, delayed actions, neutralizing actions, priorities)
  - Adaptation on other models (PN, model of Thomas...)
  - Search of attractors
  - Reverse reachability ($goal \rightarrow I_0$?)
  - Extending the method to universal properties (AF operator in CTL)

Bibliography

[1]Christian Anger, Kathrin Konczak, Thomas Linke, and Torsten Schaub. A glimpse of answer set programming. KI, 19(1) :12, 2005.

[2] Chitta Baral. Knowledge representation, reasoning and declarative problem solving. *Cambridge university press*, 2003.

[3] Loïc Paulevé, Morgan Magnin, 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.

[4] Zohra Khalis, Jean-Paul Comet, Adrien Richard, and Gilles Bernot. The smbionet method for discovering models of gene regulatory networks. *Genes, Genomes and Genomics*, 3(1) :15-22, 2009.

[5] Regina Samaga, Julio Saez-Rodriguez, Leonidas G Alexopoulos, Peter K. Sorger, and Steffen Klamt. The logic of egfr/erbb signaling : Theoretical properties and analysis of high-throughput data. *PLoS Computational Biology*, 5(8) :e1000438, 2009.

[6] Steffen Klamt, Julio Saez-Rodriguez, Jonathan Lindquist, Luca Simeoni and Ernst Gilles A methodology for the structural and functional analysis of signaling and regulatory networks, *BMC Bioinformatics*, vol. 7, no. 1, p. 56, 2006.

Thank you