

Abstract Interpretation of Dynamics of Biological Regulatory Networks

Loïc Paulevé, Morgan Magnin, Olivier Roux

{loic.pauleve,morgan.magnin,olivier.roux}@irccyn.ec-nantes.fr

IRCCyN / MOVES Team, Nantes (France)

The First International Workshop on
Static Analysis and Systems Biology (SASB 2010)
13th September 2010



The use of models for **dynamical concurrent systems** requires to formally **check** for **dynamical properties**:

- Validation of the model / control of the system.
- **Hard for large models** (state space explosion).

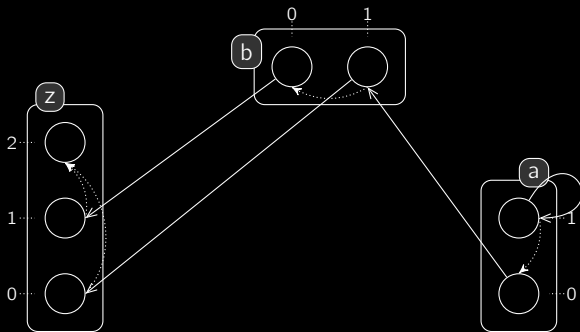
Some solutions

- State space exploration algorithm improvement.
- Automatic reduction of the model.
- **Static analysis of the model** (source code).

Outline

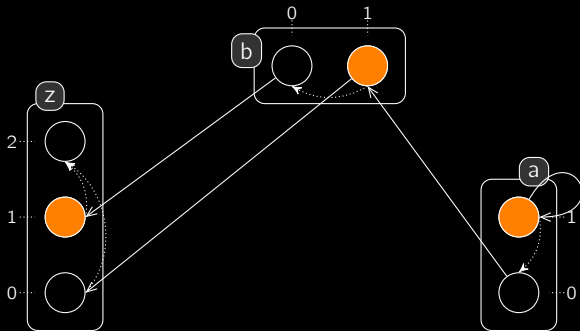
- ① Introduction of a formal framework, the **Process Hitting**.
- ② **Abstract interprtation and static analysis** of Process Hittings to prove **reachability properties**.
- ③ Application to **Biological Regulatory Networks** (BRNs) + on-going work.

The Process Hitting Framework



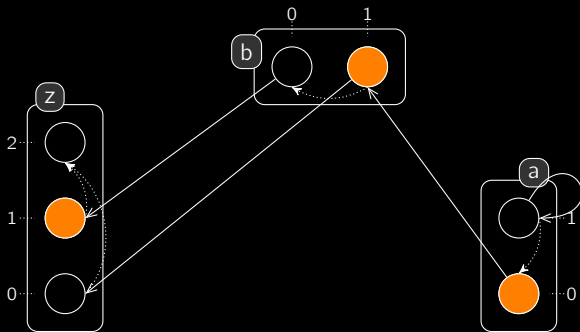
- **Sorts:** a, b, z ; **Processes:** $a_0, a_1, b_0, b_1, z_0, z_1, z_2$;
- **Actions:** a_0 hits b_1 to make it bounce to b_0, \dots ;
- **States:** $\langle a_1, b_1, z_1 \rangle, \langle a_0, b_1, z_1 \rangle, \langle a_0, b_0, z_1 \rangle, \dots$;
- Restriction of Communicating Finite-State Machines (CFSM).

The Process Hitting Framework



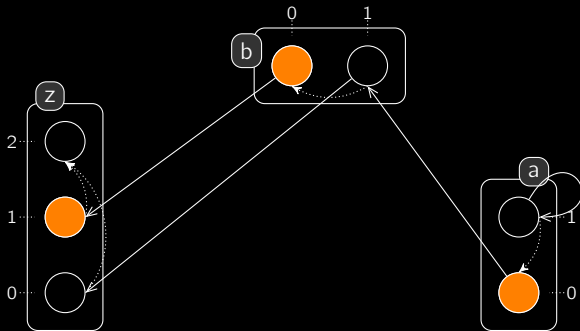
- **Sorts:** a, b, z ; **Processes:** $a_0, a_1, b_0, b_1, z_0, z_1, z_2$;
- **Actions:** a_0 hits b_1 to make it bounce to b_0, \dots ;
- **States:** $\langle a_1, b_1, z_1 \rangle, \langle a_0, b_1, z_1 \rangle, \langle a_0, b_0, z_1 \rangle, \dots$;
- Restriction of Communicating Finite-State Machines (CFSM).

The Process Hitting Framework



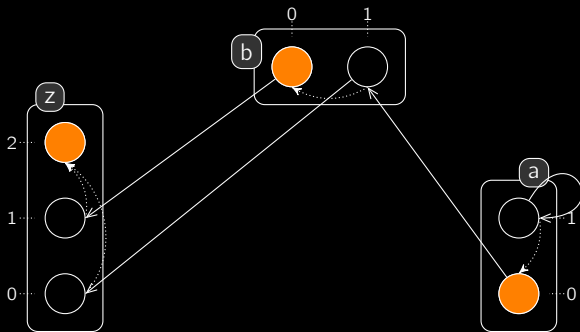
- **Sorts:** a, b, z ; **Processes:** $a_0, a_1, b_0, b_1, z_0, z_1, z_2$;
- **Actions:** a_0 hits b_1 to make it bounce to b_0, \dots ;
- **States:** $\langle a_1, b_1, z_1 \rangle, \langle a_0, b_1, z_1 \rangle, \langle a_0, b_0, z_1 \rangle, \dots$;
- Restriction of Communicating Finite-State Machines (CFSM).

The Process Hitting Framework



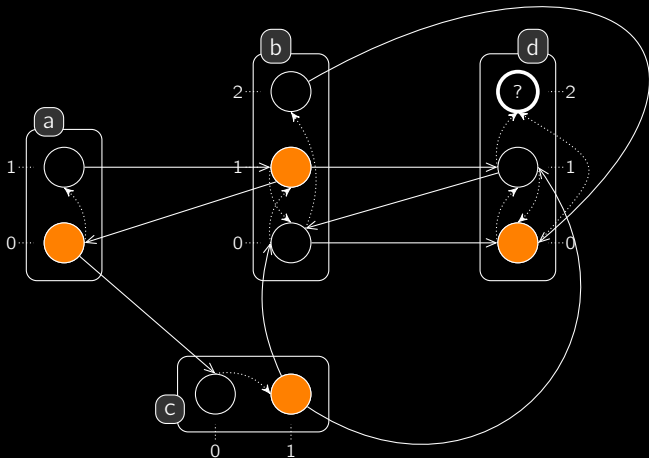
- **Sorts:** a, b, z ; **Processes:** $a_0, a_1, b_0, b_1, z_0, z_1, z_2$;
- **Actions:** a_0 hits b_1 to make it bounce to b_0, \dots ;
- **States:** $\langle a_1, b_1, z_1 \rangle, \langle a_0, b_1, z_1 \rangle, \langle a_0, b_0, z_1 \rangle, \dots$;
- Restriction of Communicating Finite-State Machines (CFSM).

The Process Hitting Framework

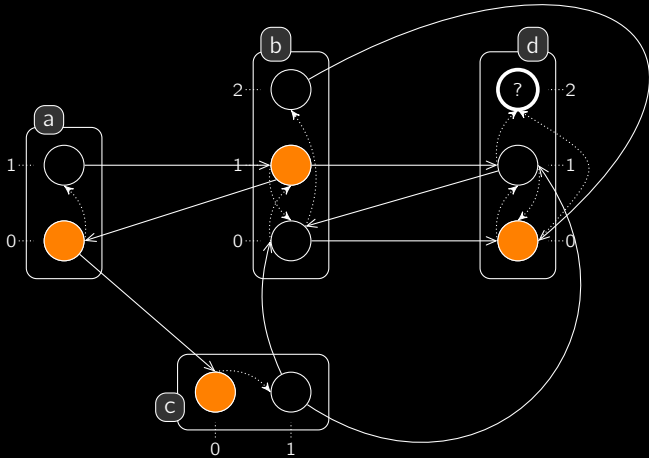


- **Sorts:** a, b, z ; **Processes:** $a_0, a_1, b_0, b_1, z_0, z_1, z_2$;
- **Actions:** a_0 hits b_1 to make it bounce to b_0, \dots ;
- **States:** $\langle a_1, b_1, z_1 \rangle, \langle a_0, b_1, z_1 \rangle, \langle a_0, b_0, z_1 \rangle, \dots$;
- Restriction of Communicating Finite-State Machines (CFSM).

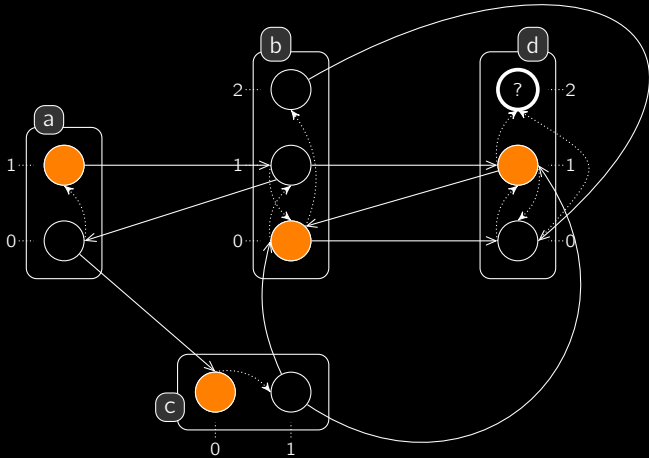
The Process Reachability Problem



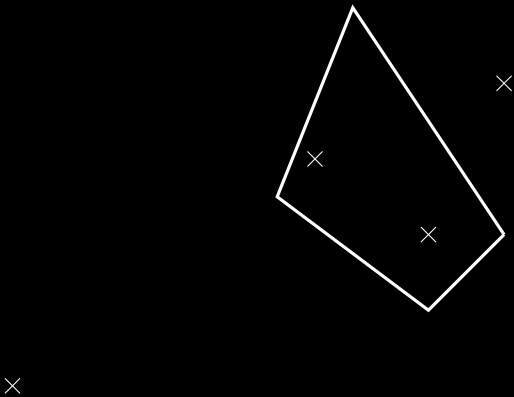
The Process Reachability Problem



The Process Reachability Problem

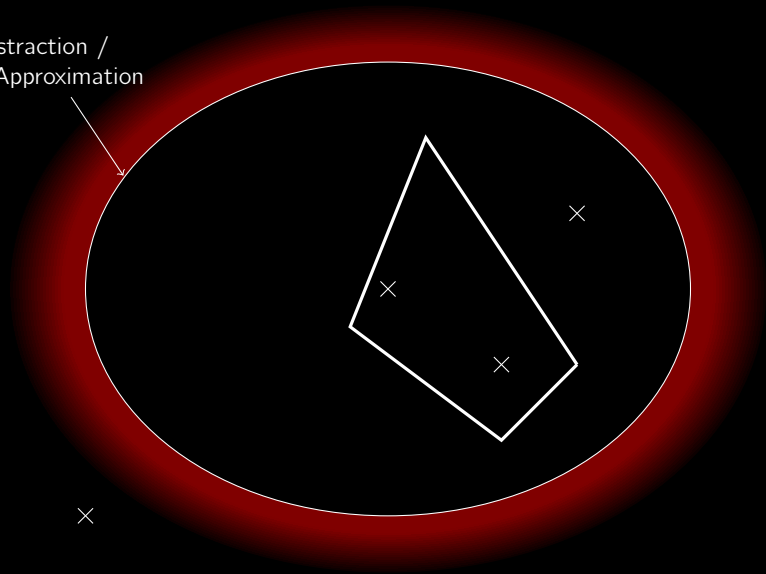


Overall Approach: Over- and Under-Approximation



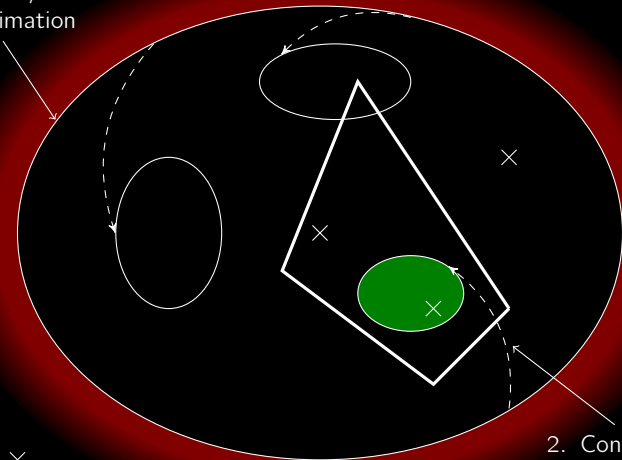
Overall Approach: Over- and Under-Approximation

1. Abstraction /
Over-Approximation



Overall Approach: Over- and Under-Approximation

1. Abstraction /
Over-Approximation



2. Concretions /
Under-Approximation

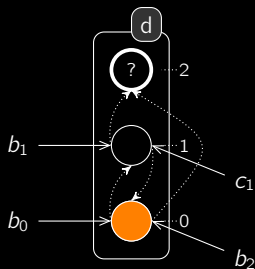
(Abstracted) Bounce Sequences

Objectives

$d_0 \overset{\cdot}{\mapsto}^* d_2, d_1 \overset{\cdot}{\mapsto}^* d_2, \dots$

Bounce Sequences

Only minimal sequences are kept.



$$\mathcal{BS}(d_0 \overset{\cdot}{\mapsto}^* d_2) = \{ [b_0 \rightarrow d_0 \overset{\cdot}{\mapsto} d_1; b_1 \rightarrow d_1 \overset{\cdot}{\mapsto} d_2], \\ [b_2 \rightarrow d_0 \overset{\cdot}{\mapsto} d_2] \}$$

$$\mathcal{BS}(d_1 \overset{\cdot}{\mapsto}^* d_2) = \{ [b_1 \rightarrow d_1 \overset{\cdot}{\mapsto} d_2], \\ [c_1 \rightarrow d_1 \overset{\cdot}{\mapsto} d_0; b_2 \rightarrow d_0 \overset{\cdot}{\mapsto} d_2] \}$$

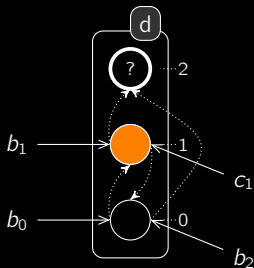
(Abstracted) Bounce Sequences

Objectives

$d_0 \overset{\cdot}{\mapsto}^* d_2, d_1 \overset{\cdot}{\mapsto}^* d_2, \dots$

Bounce Sequences

Only minimal sequences are kept.



$$\mathcal{BS}(d_0 \overset{\cdot}{\mapsto}^* d_2) = \{ [b_0 \rightarrow d_0 \overset{\cdot}{\mapsto} d_1; b_1 \rightarrow d_1 \overset{\cdot}{\mapsto} d_2], \\ [b_2 \rightarrow d_0 \overset{\cdot}{\mapsto} d_2] \}$$

$$\mathcal{BS}(d_1 \overset{\cdot}{\mapsto}^* d_2) = \{ [b_1 \rightarrow d_1 \overset{\cdot}{\mapsto} d_2], \\ [c_1 \rightarrow d_1 \overset{\cdot}{\mapsto} d_0; b_2 \rightarrow d_0 \overset{\cdot}{\mapsto} d_2] \}$$

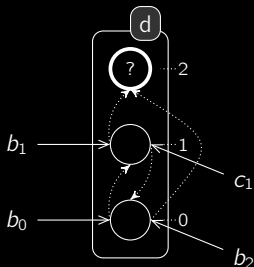
(Abstracted) Bounce Sequences

Objectives

$d_0 \mapsto^* d_2, d_1 \mapsto^* d_2, \dots$

Bounce Sequences

Only minimal sequences are kept.



$$\mathcal{BS}(d_0 \mapsto^* d_2) = \{[b_0 \rightarrow d_0 \mapsto d_1; b_1 \rightarrow d_1 \mapsto d_2], [b_2 \rightarrow d_0 \mapsto d_2]\}$$

$$\mathcal{BS}(d_1 \mapsto^* d_2) = \{[b_1 \rightarrow d_1 \mapsto d_2], [c_1 \rightarrow d_1 \mapsto d_0; b_2 \rightarrow d_0 \mapsto d_2]\}$$

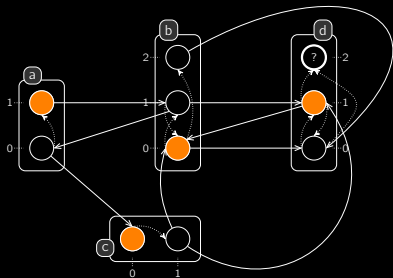
Abstracted Bounce Sequences

Only hitters are kept, and the order is forgotten.

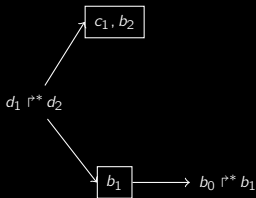
$$\mathcal{BS}^\wedge(d_0 \mapsto^* d_2) = \{\{b_0, b_1\}, \{b_2\}\}$$

$$\mathcal{BS}^\wedge(d_1 \mapsto^* d_2) = \{\{b_1\}, \{b_2, c_1\}\}$$

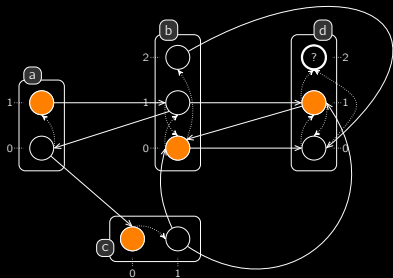
Over-approximation of Process Reachability



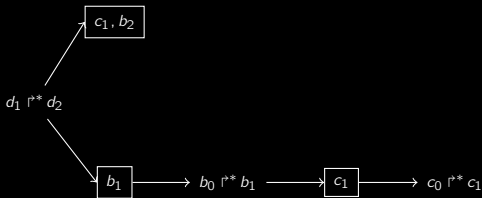
- $\mathcal{BS}^\wedge(d_1 \uparrow^* d_2) = \{\{b_1\}, \{b_2, c_1\}\}$



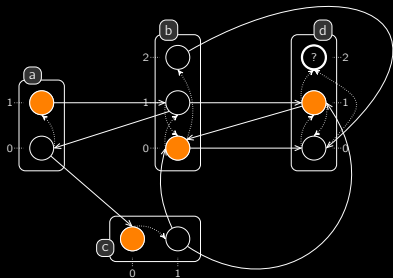
Over-approximation of Process Reachability



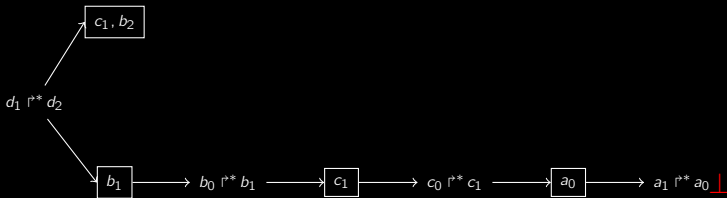
- $\mathcal{BS}^\wedge(d_1 \uparrow^* d_2) = \{\{b_1\}, \{b_2, c_1\}\}$
- $\mathcal{BS}^\wedge(b_0 \uparrow^* b_1) = \{\{c_1\}\}$



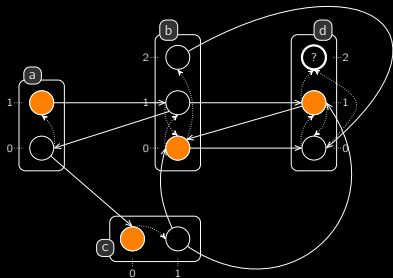
Over-approximation of Process Reachability



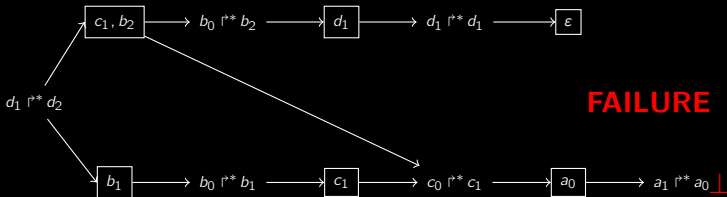
- $\mathcal{BS}^\wedge(d_1 \uparrow^* d_2) = \{\{b_1\}, \{b_2, c_1\}\}$
- $\mathcal{BS}^\wedge(b_0 \uparrow^* b_1) = \{\{c_1\}\}$
- $\mathcal{BS}^\wedge(c_0 \uparrow^* c_1) = \{\{a_0\}\}$
- $\mathcal{BS}^\wedge(a_1 \uparrow^* a_0) = \emptyset$



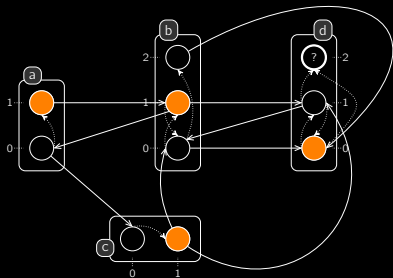
Over-approximation of Process Reachability



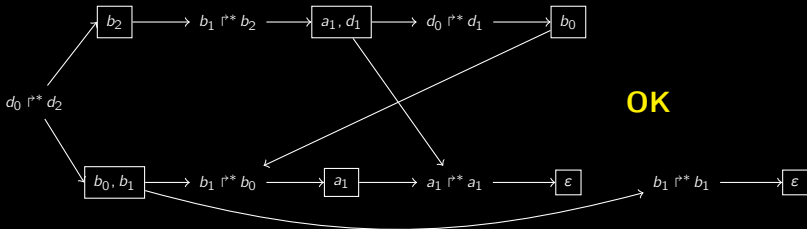
- $\mathcal{BS}^\wedge(d_1 \uparrow^* d_2) = \{\{b_1\}, \{b_2, c_1\}\}$
- $\mathcal{BS}^\wedge(b_0 \uparrow^* b_1) = \{\{c_1\}\}$
- $\mathcal{BS}^\wedge(c_0 \uparrow^* c_1) = \{\{a_0\}\}$
- $\mathcal{BS}^\wedge(a_1 \uparrow^* a_0) = \emptyset$
- $\mathcal{BS}^\wedge(b_0 \uparrow^* b_2) = \{\{d_1\}\}$
- $\mathcal{BS}^\wedge(d_1 \uparrow^* d_1) = \{\varepsilon\}$



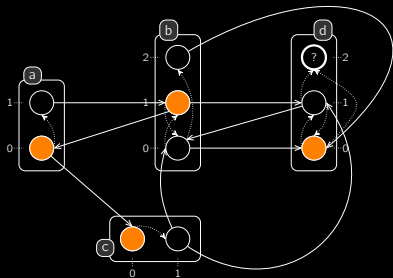
Over-approximation of Process Reachability



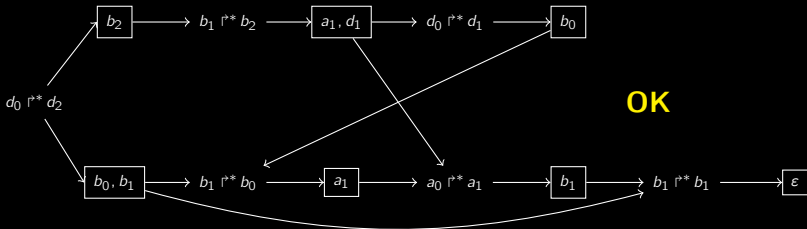
- $\mathcal{BS}^\wedge(d_1 \uparrow^* d_2) = \{\{b_1\}, \{b_2, c_1\}\}$
- $\mathcal{BS}^\wedge(b_0 \uparrow^* b_1) = \{\{c_1\}\}$
- $\mathcal{BS}^\wedge(c_0 \uparrow^* c_1) = \{\{a_0\}\}$
- $\mathcal{BS}^\wedge(a_1 \uparrow^* a_0) = \emptyset$
- $\mathcal{BS}^\wedge(b_0 \uparrow^* b_2) = \{\{d_1\}\}$
- $\mathcal{BS}^\wedge(d_1 \uparrow^* d_1) = \{\varepsilon\}$
- $\mathcal{BS}^\wedge(d_0 \uparrow^* d_2) = \{\{b_0, b_1\}, \{b_2\}\}$



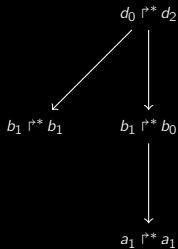
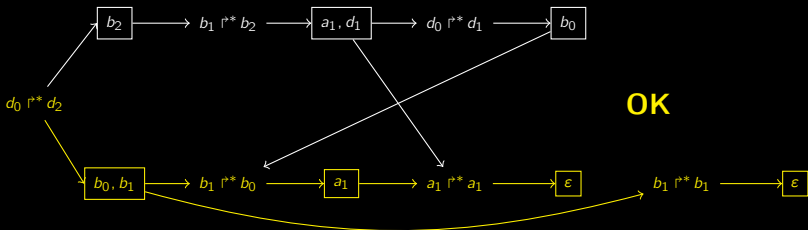
Over-approximation of Process Reachability



- $\mathcal{BS}^\wedge(d_1 \uparrow^* d_2) = \{\{b_1\}, \{b_2, c_1\}\}$
- $\mathcal{BS}^\wedge(b_0 \uparrow^* b_1) = \{\{c_1\}\}$
- $\mathcal{BS}^\wedge(c_0 \uparrow^* c_1) = \{\{a_0\}\}$
- $\mathcal{BS}^\wedge(a_1 \uparrow^* a_0) = \emptyset$
- $\mathcal{BS}^\wedge(b_0 \uparrow^* b_2) = \{\{d_1\}\}$
- $\mathcal{BS}^\wedge(d_1 \uparrow^* d_1) = \{\varepsilon\}$
- $\mathcal{BS}^\wedge(d_0 \uparrow^* d_2) = \{\{b_0, b_1\}, \{b_2\}\}$



Building concretions



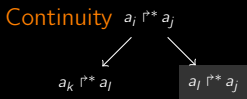
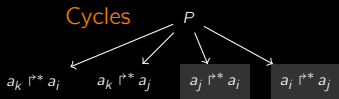
- arbitrary select one way to resolve an objective;
 - there must be no cycle.
- \Rightarrow ensure all dependencies are reachable:
- whatever the order of resolution;
 - whatever the required number of occurrence.

Under-approximation of Process Reachability

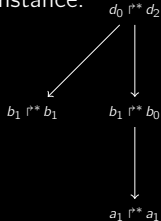
From previous abstraction:

- **arbitrary select** one way to resolve an objective;
- apply the **saturation** procedure;
- if **all required objectives** are possible, then **SUCCESS**;
- else **INCONCLUSIVE**, try another concretion.

Saturation procedure:



Instance:

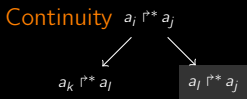
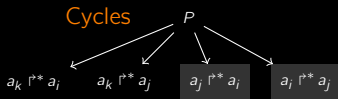


Under-approximation of Process Reachability

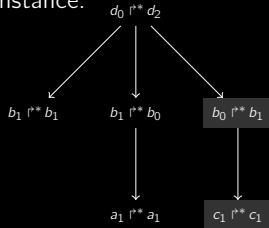
From previous abstraction:

- arbitrary select one way to resolve an objective;
- apply the saturation procedure;
- if all required objectives are possible, then SUCCESS;
- else INCONCLUSIVE, try another concretion.

Saturation procedure:



Instance:

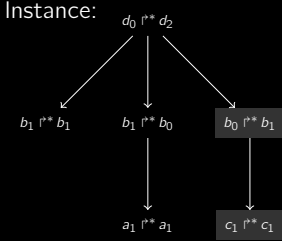
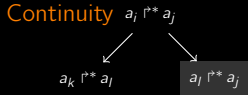
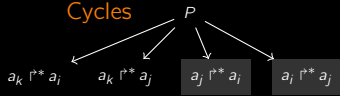


Under-approximation of Process Reachability

From previous abstraction:

- arbitrary select one way to resolve an objective;
- apply the saturation procedure;
- if all required objectives are possible, then SUCCESS;
- else INCONCLUSIVE, try another concretion.

Saturation procedure:



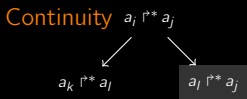
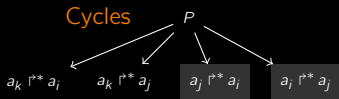
SUCCESS

Under-approximation of Process Reachability

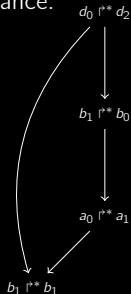
From previous abstraction:

- **arbitrary select** one way to resolve an objective;
- apply the **saturation** procedure;
- if **all required objectives** are possible, then **SUCCESS**;
- else **INCONCLUSIVE**, try another concretion.

Saturation procedure:



Instance:

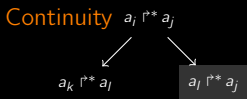
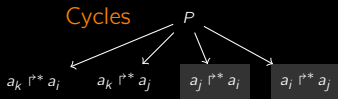


Under-approximation of Process Reachability

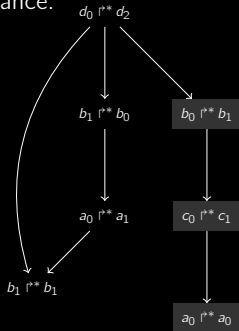
From previous abstraction:

- arbitrary select one way to resolve an objective;
- apply the saturation procedure;
- if all required objectives are possible, then **SUCCESS**;
- else **INCONCLUSIVE**, try another concretion.

Saturation procedure:



Instance:

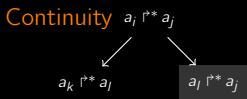
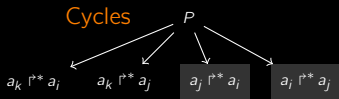


Under-approximation of Process Reachability

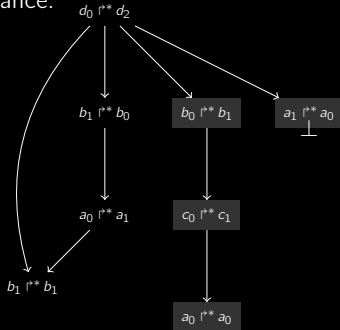
From previous abstraction:

- arbitrary select one way to resolve an objective;
- apply the saturation procedure;
- if all required objectives are possible, then **SUCCESS**;
- else **INCONCLUSIVE**, try another concretion.

Saturation procedure:



Instance:

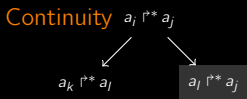
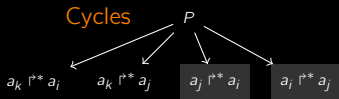


Under-approximation of Process Reachability

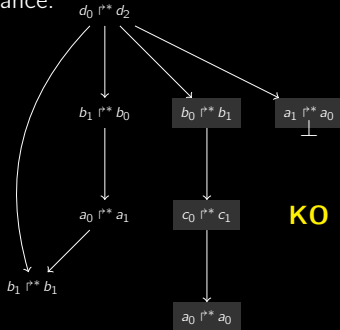
From previous abstraction:

- arbitrary select one way to resolve an objective;
- apply the saturation procedure;
- if all required objectives are possible, then **SUCCESS**;
- else **INCONCLUSIVE**, try another concretion.

Saturation procedure:



Instance:



Pre-Conclusion

Comments

- Quite **simple approximations** from Process Hittings source code;
- **prevent explicit state space** exploration;
- abstract structures provide **information on required steps** for reachability.

Complexities

- Computing \mathcal{BS}^\wedge is exponential in the size of the sort;
- other operations are \approx **polynomial in the number of sorts**.

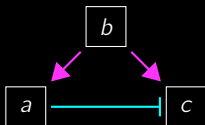
Outlook

- $\mathcal{BS}^\wedge(a_i \xrightarrow{*} a_j) = \emptyset \Rightarrow a_i$ **cannot appear after** a_j ;
- refine over- and under-approximations with this knowledge:
- **reduce inconclusiveness**.

Biological Regulatory Networks (BRNs)

- Relates **components** with actions of **activation** and **inhibition**;
- each component has a **finite number of ordered states**;
- the next state of a component is function of its activators/inhibitors.

Interaction graph

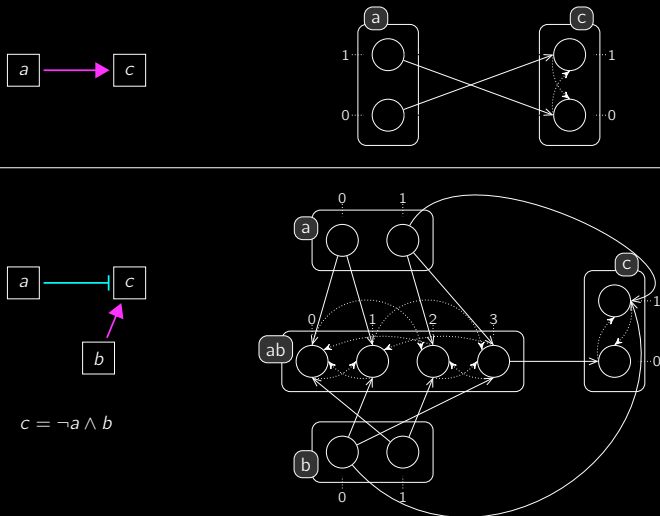


Cooperations

For instance (boolean case): $c = \neg a \wedge b$.

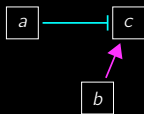
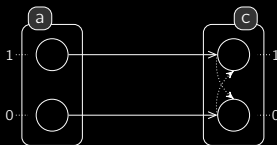
[René Thomas in *Journal of Theoretical Biology*, 1973] [A. Richard, J.-P. Comet, G. Bernot in *Modern Formal Methods and Applications*, 2006]

From BRNs to Process Hittings

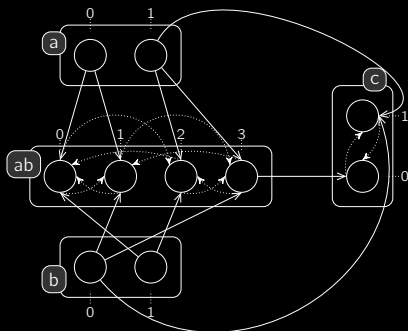


[Paulevé, *et al.* in *Trans. in Computational Systems Biology*, 2010]

From BRNs to Process Hittings



$$c = \neg a \wedge b$$



[Paulevé, *et al.* in *Trans. in Computational Systems Biology*, 2010]

Experimentations

T-Cell Receptor Signalling Pathway (40 components)

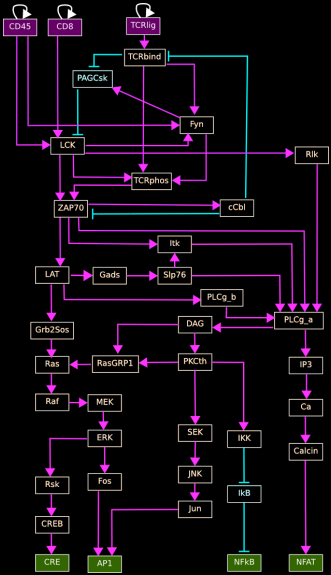
Process Hitting: 53 sorts, 176 processes, 416 actions, largest sort: 32 processes; total: $\approx 10^{22}$ states.

Reachability analysis:

- always conclusive;
- execution time: around 1s.
- libddd framework: from a few seconds to many hours (or out of memory).

Bottleneck: \mathcal{BS}^{\wedge} computation (large sorts).

[Klamt, et al. in BMC Bioinformatics, 2006]
 [Naldi, Thieffry, Chaouiya in CMSB 2007]
<http://ddd.lip6.fr>



Conclusion and TODOs

Conclusion

- Efficient method to (semi-)decide process reachability;
- static analysis and abstract interpretation from the model;
- exploit particular structures of the language used (Process Hitting);
- no explicit state space construction;
- promising scalability.

On-going work

- Reduce inconclusiveness: exploit partial order of process appearance.
- Extract necessary processes for achieving reachabilities: towards control.
- Software soon at <http://www.irccyn.ec-nantes.fr/~pauleve>.

Future work

- How does the analysis apply to less restricted CFSM?
- Quantitative analysis (eg: probability of reaching a process within a time interval).