The NII-Yamanashi-LRI Workshop

Exhaustive analysis of the dynamics of Process Hitting through Answer Set Programming

Emna BEN ABDALLAH & Maxime FOSCHETTE

joint work Olivier ROUX & Morgan MAGNIN

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

Année: 2013/2014
Context and Aims

**MeForBio** team:
Algebraic modelling to study
complex dynamical biological systems
Exhaustive analysis of the dynamics of Process Hitting through ASP

Introduction

Context and Aims

MeForBio team:
Algebraic modelling to study complex dynamical biological systems

1) What are the models?
   Biological Regulatory Networks (BRNs): Studying gene interactions with mathematical tools;
   Process Hitting (PH): a new developed model.
Exhaustive analysis of the dynamics of Process Hitting through ASP

Introduction

Context and Aims

MeForBio team:
Algebric modelling to study complex dynamical biological systems

1) What are the models?
   Biological Regulatory Networks (BRNs): Studying gene interactions with mathematical tools;
   Process Hitting (PH): a new developed model.

2) What did I do?
   Predicting the evolutions of the network.
Exhaustive analysis of the dynamics of Process Hitting through ASP

Introduction

Context and Aims

MeForBio team:
Algebraic modelling to study complex dynamical biological systems

1) What are the models?
   Biological Regulatory Networks (BRNs): Studying gene interactions with mathematical tools;
   Process Hitting (PH): a new developed model.

2) What did I do?
   Predicting the evolutions of the network.

3) What for?
   searching of PH properties through ASP (Fixed points, reachability).
Exhaustive analysis of the dynamics of Process Hitting through ASP

Plan

1) Answer set programming (ASP)
   - Definition
   - Example of an ASP program

2) Process Hitting (PH)
   - Definition
   - PH through ASP

3) Fixed point
   - Definition
   - ASP implementation
   - Optimization of ASP implementation
   - Comparison

4) Reachability
   - Definition
   - ASP implementation
   - Iterative ASP implementation
   - Comparison

5) Conclusion & prospect
Exhaustive analysis of the dynamics of Process Hitting through ASP

Plan

1) Answer set programming (ASP)
   - Definition
   - Example of an ASP program

2) Process Hitting (PH)
   - Definition
   - PH through ASP

3) Fixed point
   - Definition
   - ASP implementation
   - Optimization of ASP implementation
   - Comparison

4) Reachability
   - Definition
   - ASP implementation
   - Iterative ASP implementation
   - Comparison

5) Conclusion & prospect
Plan

1) Answer set programming (ASP)
   - Definition
   - Example of an ASP program
Exhaustive analysis of the dynamics of Process Hitting through ASP

Plan

1) Answer set programming (ASP)
   - Definition
   - Example of an ASP program

2) Process Hitting (PH)
   - Definition
   - PH through ASP
Plan

1) Answer set programming (ASP)
   - Definition
   - Example of an ASP program

2) Process Hitting (PH)
   - Definition
   - PH through ASP

3) Fixed point
   - Definition
   - ASP implementation
   - Optimization of ASP implementation
   - Comparaison
Exhaustive analysis of the dynamics of Process Hitting through ASP

Plan

1) Answer set programming (ASP)
   - Definition
   - Example of an ASP program

2) Process Hitting (PH)
   - Definition
   - PH through ASP

3) Fixed point
   - Definition
   - ASP implementation
   - Optimization of ASP implementation
   - Comparison

4) Reachability
   - Definition
   - ASP implementation
   - Iterative ASP implementation
   - Comparison
Plan

1) Answer set programming (ASP)
   – Definition
   – Example of an ASP program

2) Process Hitting (PH)
   – Definition
   – PH through ASP

3) Fixed point
   – Definition
   – ASP implementation
   – Optimization of ASP implementation
   – Comparaison

4) Reachability
   – Definition
   – ASP implementation
   – Iterative ASP implementation
   – Comparaison

5) Conclusion & prospect
Answer Set Programming

ASP:

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

\[
\text{head} \leftarrow \text{body}.
\]
\[
L_0 \leftarrow L_1, \ldots, L_m, \text{not } L_{m+1}, \ldots, \text{not } L_n.
\]

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

Rule’s meaning:

If \( L_1, \ldots, L_m \) are true and if \( L_{m+1}, \ldots, L_n \) are false then \( L_0 \) is true.
Exhaustive analysis of the dynamics of Process Hitting through ASP ○ Answer Set Programming ○ Definition

Answer Set Programming

Special types of rules:

- **Constraint** :
  \[ \leftarrow L_1, \ldots, L_m, \textbf{not} L_{m+1}, \ldots, \textbf{not} L_n. \]

- **Fact** :
  \[ L_0. \]

- **Cardinality** :
  \[ \min\{L_0, \ldots, L_j\} \max \leftarrow L_1, \ldots, L_m, \textbf{not} L_{m+1}, \ldots, \textbf{not} L_n. \]
Answer Set Programming

Example:

\[
\begin{align*}
\text{bird}(X) & \leftarrow \text{lays_egg}(X). \\
\text{mammal}(X) & \leftarrow \text{engender}(X). \\
\text{fly}(X) & \leftarrow \text{bird}(X), \textbf{not} \ \text{mammal}(X). \\
\text{lays_egg}(\text{tweety}).
\end{align*}
\]
Exhaustive analysis of the dynamics of Process Hitting through ASP

Answer Set Programming

Example:

\[
\begin{align*}
\text{bird}(X) & \leftarrow \text{lays\_egg}(X). \\
\text{mammal}(X) & \leftarrow \text{engender}(X). \\
\text{fly}(X) & \leftarrow \text{bird}(X), \quad \text{not} \quad \text{mammal}(X). \\
\text{lays\_egg}(\text{tweety}).
\end{align*}
\]

Solution:

\[
\begin{align*}
\text{bird}(\text{tweety}) & \leftarrow \text{True}. \\
\text{mammal}(\text{tweety}) & \leftarrow \text{unknown}.
\end{align*}
\]
Answer Set Programming

Example:

\[
\begin{align*}
\text{bird}(X) & \leftarrow \text{lays\_egg}(X). \\
\text{mammal}(X) & \leftarrow \text{engender}(X). \\
\text{fly}(X) & \leftarrow \text{bird}(X), \text{not mammal}(X). \\
\text{lays\_egg}(\text{tweety}).
\end{align*}
\]

Solution:

\[
\begin{align*}
\text{bird}(\text{tweety}) & \leftarrow \text{True}. \\
\text{mammal}(\text{tweety}) & \leftarrow \text{unknown}. \\
\text{fly}(\text{tweety}) & \leftarrow \text{bird}(\text{tweety}), \text{not mammal}(\text{tweety}).
\end{align*}
\]
Answer Set Programming

Example:

\[
\begin{align*}
bird(X) & \leftarrow \text{lays\_egg}(X). \\
mammal(X) & \leftarrow \text{engender}(X). \\
fly(X) & \leftarrow bird(X), \textbf{not} mammal(X). \\
lays\_egg(\text{tweety}).
\end{align*}
\]

Solution:

\[
\begin{align*}
bird(\text{tweety}) & \leftarrow \text{True}. \\
mammal(\text{tweety}) & \leftarrow \text{unknown}. \\
fly(\text{tweety}) & \leftarrow bird(\text{tweety}), \textbf{not} mammal(\text{tweety}). \\
fly(\text{tweety}) & \leftarrow \text{True}, \textbf{not} \text{ unknown}.
\end{align*}
\]
Answer Set Programming

Example:

\[
\begin{align*}
\text{bird}(X) & \leftarrow \text{lays\_egg}(X). \\
\text{mammal}(X) & \leftarrow \text{engender}(X). \\
\text{fly}(X) & \leftarrow \text{bird}(X), \textbf{not} \ \text{mammal}(X). \\
\text{lays\_egg}(\text{tweety}).
\end{align*}
\]

Solution:

\[
\begin{align*}
\text{bird}(\text{tweety}) & \leftarrow \text{True}. \\
\text{mammal}(\text{tweety}) & \leftarrow \text{unknown}. \\
\text{fly}(\text{tweety}) & \leftarrow \text{bird}(\text{tweety}), \textbf{not} \ \text{mammal}(\text{tweety}). \\
\text{fly}(\text{tweety}) & \leftarrow \text{True, not unknown}. \\
\text{fly}(\text{tweety}) & \leftarrow \text{True, True}.
\end{align*}
\]
Answer Set Programming

Example:

\(bird(X) \leftarrow lays\_egg(X).\)
\(mammal(X) \leftarrow engender(X).\)
\(fly(X) \leftarrow bird(X), \text{not mammal}(X).\)
\(lays\_egg(tweety).\)

Solution:

\(bird(tweety) \leftarrow \text{True}.\)
\(mammal(tweety) \leftarrow \text{unknown}.\)
\(fly(tweety) \leftarrow bird(tweety), \text{not mammal}(tweety).\)
\(fly(tweety) \leftarrow \text{True, not unknown}.\)
\(fly(tweety) \leftarrow \text{True, True}.\)
\(fly(tweety) \leftarrow \text{True}.\)
Answer Set Programming

Example:

\[ \text{bird}(X) \leftarrow \text{lays\_egg}(X). \]
\[ \text{mammal}(X) \leftarrow \text{engender}(X). \]
\[ \text{fly}(X) \leftarrow \text{bird}(X), \text{not mammal}(X). \]
\[ \text{lays\_egg}(\text{tweety}). \]

Solution:

\[ \text{bird}(\text{tweety}) \leftarrow \text{True}. \]
\[ \text{mammal}(\text{tweety}) \leftarrow \text{unknown}. \]
\[ \text{fly}(\text{tweety}) \leftarrow \text{bird}(\text{tweety}), \text{not mammal}(\text{tweety}). \]
\[ \text{fly}(\text{tweety}) \leftarrow \text{True}, \text{not unknown}. \]
\[ \text{fly}(\text{tweety}) \leftarrow \text{True}, \text{True}. \]
\[ \text{fly}(\text{tweety}) \leftarrow \text{True}. \]

Answer: \text{fly(\text{tweety}), bird(\text{tweety}).}
The Process Hitting modeling

Sorts: components  $a, b, z$
Exhaustive analysis of the dynamics of Process Hitting through ASP

The Process Hitting modeling

**Sorts**: components \( a, b, z \)

**Processes**: local states / levels of expression \( z_0, z_1, z_2 \)
Exhaustive analysis of the dynamics of Process Hitting through ASP

The Process Hitting modeling

**Sorts:** components  \( a, b, z \)

**Processes:** local states / levels of expression  \( z_0, z_1, z_2 \)

**States:** sets of active processes  \( \langle a_0, b_1, z_0 \rangle \)
Exhaustive analysis of the dynamics of Process Hitting through ASP

The Process Hitting modeling

Sorts: 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 \( b_1 \rightarrow z_0 \overset{\cdot}{\rightarrow} z_1, a_0 \rightarrow a_0 \overset{\cdot}{\rightarrow} a_1, a_1 \rightarrow z_1 \overset{\cdot}{\rightarrow} z_2 \)

\( b_1 \rightarrow z_0 \overset{\cdot}{\rightarrow} z_1, a_0 \rightarrow a_0 \overset{\cdot}{\rightarrow} a_1, a_1 \rightarrow z_1 \overset{\cdot}{\rightarrow} z_2 \)
Exhaustive analysis of the dynamics of Process Hitting through ASP

**The Process Hitting modeling**

**Sorts:** 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

\[
\begin{align*}
&b_1 \rightarrow z_0 \uparrow z_1, \\
&a_0 \rightarrow a_0 \uparrow a_1, \\
&a_1 \rightarrow z_1 \uparrow z_2 \\
&b_1 \rightarrow z_0 \uparrow z_1, \\
&a_0 \rightarrow a_0 \uparrow a_1, \\
&a_1 \rightarrow z_1 \uparrow z_2
\end{align*}
\]
Exhaustive analysis of the dynamics of Process Hitting through ASP

The Process Hitting modeling

**Sorts:** 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 $b_1 \rightarrow z_0 \uparrow z_1$, $a_0 \rightarrow a_0 \uparrow a_1$, $a_1 \rightarrow z_1 \uparrow z_2$

$b_1 \rightarrow z_0 \uparrow z_1$, $a_0 \rightarrow a_0 \uparrow a_1$, $a_1 \rightarrow z_1 \uparrow z_2$
Exhaustive analysis of the dynamics of Process Hitting through ASP

The Process Hitting modeling

**Sorts:** 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  

- \( b_1 \rightarrow z_0 \uparrow z_1, a_0 \rightarrow a_0 \uparrow a_1, a_1 \rightarrow z_1 \uparrow z_2 \)
- \( b_1 \rightarrow z_0 \uparrow z_1, a_0 \rightarrow a_0 \uparrow a_1, a_1 \rightarrow z_1 \uparrow z_2 \)
Network traduction:

- **Sort**: sort(A).
- **Process**: process(A,I).
- **Action** \( a_i \rightarrow b_j \rightarrow b_k \) : action(A,I,B,J,K).

Example:

\[
\begin{align*}
&\text{sort}(\text{"a"}). \\
&\text{sort}(\text{"b"}). \\
&\text{process}(\text{"a"}, 0..1). \\
&\text{process}(\text{"b"}, 0..2). \\
&\text{action}(\text{"b"},0,\text{"a"},0,1). \\
&\text{action}(\text{"b"},1,\text{"b"},1,0).
\end{align*}
\]
Network traduction:

- **Sort**: sort(A).
- **Process**: process(A,I).
- **Action** \( a_i \rightarrow b_j \rightarrow b_k \): action(A,I,B,J,K).

Example:

![Diagram with nodes labeled a and b, and connections between them representing actions and processes.](image-url)
Network traduction:

- **Sort**: sort(A).
- **Process**: process(A,I).
- **Action** $a_i \rightarrow b_j \rightarrow b_k$: action(A,I,B,J,K).

Example:

```
sort("a"). sort("b").
```

```
2
1
0
```

```
1
0
```

```
```
Exhaustive analysis of the dynamics of Process Hitting through ASP

**PH through ASP**

**Network traduction:**

- **Sort:** sort(A).
- **Process:** process(A,I).
- **Action** $a_i \rightarrow b_j \Rightarrow b_k : \text{action}(A,I,B,J,K)$.

**Example:**

```
sort("a"). sort("b").
process("a", 0..1).
process("b", 0..2).
```
Network traduction:

- **Sort**: sort(A).
- **Process**: process(A,I).
- **Action** $a_i \rightarrow b_j \mapsto b_k$ : action(A,I,B,J,K).

Example:

```
sort("a"). sort("b").
process("a", 0..1).
process("b", 0..2).
action("b",0,"a",0,1).
action("b",1,"b",1,0).
```
Fixed point = state where no action can be played
Exhaustive analysis of the dynamics of Process Hitting through ASP

Fixed point

Fixed point \(=\) state where no action can be played

\[\rightarrow\] Hitless Graph
Fixed point = state where no action can be played

→ Hitless Graph → $n$-clics = fixed points
Fixed point = state where no action can be played
  → Hitless Graph → n-clics = fixed points
Implementation of the algorithm (N-Cliques)

Definition of hitless graph:

\[\text{noAction}(A,I,B,J) :- \text{not hit}(A,I,B,J), \text{not hit}(B,J,A,I), A\neq B,\]
\[\text{shownProcess}(A,I), \text{shownProcess}(B,J).\]
Exhaustive analysis of the dynamics of Process Hitting through ASP ◦ Fixed point ◦ ASP

Implementation of the algorithm (N-Cliques)

**Definition of hitless graph:**


[Diagram of a graph showing the relationships between nodes labeled 'a', 'b', and 'z'.]
Definition of hitless graph:
\[
\text{noAction}(A,I,B,J) :- \neg \text{hit}(A,I,B,J), \neg \text{hit}(B,J,A,I), A \neq B,
\]
\[
\text{shownProcess}(A,I), \text{shownProcess}(B,J).
\]

Select processes:
\[
1 \{ \text{selectProcess}(A,I) : \text{shownProcess}(A,I) \} 1 :- \text{sort}(A).
\]
Exhaustive analysis of the dynamics of Process Hitting through ASP ○ Fixed point ○ ASP

**Implementation of the algorithm (N-Cliques)**

**Definition of hitless graph:**

\[
\text{noAction}(A,I,B,J) :- \neg \text{hit}(A,I,B,J), \neg \text{hit}(B,J,A,I), A \neq B, \\
\text{shownProcess}(A,I), \text{shownProcess}(B,J).
\]

**Select processes:**

\[
1 \{ \text{selectProcess}(A,I) : \text{shownProcess}(A,I) \} \ 1 :- \text{sort}(A).
\]

**Find Fixed points:**

\[
\text{noHit}(A,I,B,J) :- \text{noAction}(A,I,B,J), \\
\text{selectProcess}(A,I), \text{selectProcess}(B,J).
\]

\[
\text{noExistFixPoint} :- 0 \{ \text{noHit}(A,I,B,J) \} 0, \text{selectProcess}(A,I), \\
\text{selectProcess}(B,J).
\]
 Definition of hitless graph:

\[
\text{noAction}(A,I,B,J) :- \text{not hit}(A,I,B,J), \text{not hit}(B,J,A,I), A!=B, \\
\quad \text{shownProcess}(A,I), \text{shownProcess}(B,J).
\]

Select processes:

\[
1 \{\text{selectProcess}(A,I) : \text{shownProcess}(A,I) \} 1 :- \text{sort}(A).
\]

Find Fixed points:

\[
\text{noHit}(A,I,B,J) :- \text{noAction}(A,I,B,J), \\
\quad \text{selectProcess}(A,I), \text{selectProcess}(B,J).
\]

\[
\text{noExistFixPoint} :- 0 \{\text{noHit}(A,I,B,J)\} 0, \text{selectProcess}(A,I), \\
\quad \text{selectProcess}(B,J).
\]
ASPs program result:

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

1 \{ \text{selectProcess}(A,I) : \text{showProcess}(A,I) \} \ 1 \leftarrow \text{sort}(A).
\text{noHit}(A,I,B,J) \leftarrow \text{noAction}(A,I,B,J), \text{selectProcess}(A,I),
\text{selectProcess}(B,J), A\neq B.
\neg \text{noExistFixPoint} : 0 \{ \text{noHit}(A,I,B,J) \} 0, \text{selectProcess}(A,I),
\text{selectProcess}(B,J).
\leftarrow \neg \text{noExistFixPoint}.
\text{fixProcess}(A,I) \leftarrow \text{selectProcess}(A,I).

\begin{center}
\begin{tikzpicture}
  \node (z) at (0,0) [circle,fill,inner sep=0.5pt] {Z};
  \node (a) at (3,1) [circle,fill,inner sep=0.5pt] {a};
  \node (b) at (1,2) [circle,fill,inner sep=0.5pt] {b};

  \draw [fill=white] (0,3) circle (0.5pt);
  \draw [fill=white] (0,2) circle (0.5pt);
  \draw [fill=white] (0,1) circle (0.5pt);
  \draw [fill=white] (0,0) circle (0.5pt);

  \draw [fill=white] (1,3) circle (0.5pt);
  \draw [fill=white] (1,2) circle (0.5pt);
  \draw [fill=white] (1,1) circle (0.5pt);
  \draw [fill=white] (1,0) circle (0.5pt);

  \draw [fill=white] (2,3) circle (0.5pt);
  \draw [fill=white] (2,2) circle (0.5pt);
  \draw [fill=white] (2,1) circle (0.5pt);
  \draw [fill=white] (2,0) circle (0.5pt);

  \draw (z) -- (a);
  \draw (z) -- (b);
  \draw (a) -- (b);
  \draw (b) -- (z);
\end{tikzpicture}
\end{center}
Optimization:

\[ 1 \ \{ \text{selectProcess}(A,I) : \text{shownProcess}(A,I) \} \ 1 : - \ \text{sort}(A). \]

\[ : - \ \text{hit}(A,I,B,J), \ \text{selectProcess}(A,I), \ \text{selectProcess}(B,J), \ A!=B. \]
Exhaustive analysis of the dynamics of Process Hitting through ASP o Fixed point o Comparaison

Static analysis

Fixed Point

Comparison

<table>
<thead>
<tr>
<th>Model</th>
<th>#sorts</th>
<th>#states</th>
<th>#fix-point</th>
<th>mthd1</th>
<th>mthd2</th>
<th>PINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>mvbrn</td>
<td>3</td>
<td>12</td>
<td>1</td>
<td>0.000s</td>
<td>0.000s</td>
<td>0.006s</td>
</tr>
<tr>
<td>ERBB</td>
<td>42</td>
<td>$2^{70}$</td>
<td>3</td>
<td>0.220s</td>
<td>0.000s</td>
<td>0.017s</td>
</tr>
<tr>
<td>tcrsig40</td>
<td>54</td>
<td>$2^{73}$</td>
<td>1</td>
<td>0.220s</td>
<td>0.020s</td>
<td>0.021s</td>
</tr>
<tr>
<td>tcrsig94</td>
<td>133</td>
<td>$2^{194}$</td>
<td>0</td>
<td>2.540s</td>
<td>0.060s</td>
<td>0.027s</td>
</tr>
<tr>
<td>egfr104</td>
<td>193</td>
<td>$2^{320}$</td>
<td>0</td>
<td>8.220s</td>
<td>0.140s</td>
<td>0.074s</td>
</tr>
</tbody>
</table>

Figure: Excecuting time of ASP methods 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:
Exhaustive analysis of the dynamics of Process Hitting through ASP

Dynamic analysis

Reachability

Reachability of processes:

• Initial context

\[ \langle a_0, b_0, c_0, z_0 \rangle \]
Exhaustive analysis of the dynamics of Process Hitting through ASP

Dynamic analysis

Reachability

Reachability of processes:

- Initial context
  \[ \langle a_0, b_0, c_0, z_0 \rangle \]
- Objectives
  \[ \overset{d_2}{\rightarrow} \]
Exhaustive analysis of the dynamics of Process Hitting through ASP

Reachability

Dynamic analysis

Reachability

Reachability of processes:

- Initial context

\[ \langle a_0, b_0, c_0, z_0 \rangle \]

- Objectives

\[ [ \uparrow d_2 ] \]

→ Concretization of the objective = scenario

\[ a_0 \rightarrow c_0 \uparrow c_1 :: b_0 \rightarrow d_0 \uparrow d_1 :: c_1 \rightarrow b_0 \uparrow b_1 :: b_1 \rightarrow d_1 \uparrow d_2 \]
Dynamic analysis

Reachability

Reachability of processes:

- Initial context
  \[ \langle a_0, b_0, c_0, z_0 \rangle \]

- Objectives
  \[ [ \overset{\rightharpoonup}{\rightarrow} d_2 ] \]

\[ a_0 \overset{\rightarrow} \rightarrow c_0 \overset{\rightharpoonup}{\rightarrow} c_1 :: b_0 \overset{\rightarrow} \rightarrow d_0 \overset{\rightharpoonup}{\rightarrow} d_1 :: c_1 \overset{\rightharpoonup}{\rightarrow} b_0 \overset{\rightharpoonup}{\rightarrow} b_1 :: b_1 \overset{\rightharpoonup}{\rightarrow} d_1 \overset{\rightharpoonup}{\rightarrow} d_2 \]
Exhaustive analysis of the dynamics of Process Hitting through ASP ◦ Reachability

Dynamic analysis
Reachability

Reachability of processes:

- Initial context
  \[ \langle a_0, b_0, c_0, z_0 \rangle \]
- Objectives

\[ \uparrow \rightarrow \]
Concretization of the objective = scenario

\[ a_0 \rightarrow c_0 \uparrow c_1 :: b_0 \rightarrow d_0 \uparrow d_1 :: c_1 \rightarrow b_0 \uparrow b_1 :: b_1 \rightarrow d_1 \uparrow d_2 \]
Exhaustive analysis of the dynamics of Process Hitting through ASP

Dynamic analysis
Reachability

Reachability of processes:

- Initial context
  \[ \langle a_0, b_0, c_0, z_0 \rangle \]
- Objectives
  \[ [ \uparrow d_2 ] \]

→ Concretization of the objective = scenario
\[
\begin{align*}
a_0 & \rightarrow c_0 \uparrow c_1 :: b_0 \rightarrow d_0 \uparrow d_1 :: c_1 \rightarrow b_0 \uparrow b_1 :: b_1 \rightarrow d_1 \uparrow d_2
\end{align*}
\]
Exhaustive analysis of the dynamics of Process Hitting through ASP

Dynamic analysis

Reachability

Reachability of processes:

- Initial context
  \[ \langle a_0, b_0, c_0, z_0 \rangle \]
- Objectives
  \[ [ \Downarrow d_2 ] \]

→ Concretization of the objective = scenario

\[
a_0 \rightarrow c_0 \Downarrow c_1 :: b_0 \rightarrow d_0 \Downarrow d_1 :: c_1 \rightarrow b_0 \Downarrow b_1 :: b_1 \rightarrow d_1 \Downarrow d_2
\]
Exhaustive analysis of the dynamics of Process Hitting through ASP ○ Reachability

Dynamic analysis

Reachability

Reachability of processes:

- Initial context
  \[ \langle a_0, b_0, c_0, z_0 \rangle \]
- Objectives
  \[ [ \uparrow d_2 ] \]

→ Concretization of the objective = scenario
  \[ a_0 \rightarrow c_0 \uparrow c_1 :: b_0 \rightarrow d_0 \uparrow d_1 :: c_1 \rightarrow b_0 \uparrow b_1 :: b_1 \rightarrow d_1 \uparrow d_2 \]
Dynamic analysis
Evolution through ASP

Network evolution through ASP

Exhaustive analysis of the dynamics of Process Hitting through ASP ○ Reachability ○ ASP 1st method
Dynamic analysis
Evolution through ASP

Network evolution through ASP

Initializing:

\[
\text{init(}\text{activeProcess("a",0))}.\quad \text{avec } a: \text{ sorte, } 0: \text{ indice du processus}
\]
Dynamic analysis
Evolution through ASP

Network evolution through ASP

Playable actions at step T:

\[
\text{playableAction}(A,I,B,J,K,T) :- \text{action}(A,I,B,J,K), \notag \\
\text{instate}(\text{activeProcess}(A,I),T), \notag \\
\text{instate}(\text{activeProcess}(B,J),T), \text{time}(T). \notag 
\]
Dynamic analysis
Evolution through ASP

Network evolution through ASP

Change active processes:

\[
\{\text{activeFromTo}(B,J,K,T)\} \leftarrow \text{playableAction}(A,I,B,J,K,T),
J \neq K, \text{time}(T).
\]
\[
\leftarrow 2\{\text{activeFromTo}(B,J,K,T)\}, \text{time}(T).
\]
Dynamic analysis
Evolution through ASP

Network evolution through ASP

Active processes at next step (T+1):

\[
\text{instate}(\text{activeProcess}(B,K),T+1) \leftarrow \text{activeFromTo}(B,J,K,T), \text{time}(T).
\]
\[
\text{instate}(\text{activeProcess}(A,I),T+1) \leftarrow \text{instate}(\text{activeProcess}(A,I),T),\text{activeFromTo}(B,J,K,T), A \neq B, \text{time}(T).
\]
Network evolution through ASP

time(0..N).

Results (N = 3):

Answer 1: activeFromTo("d",0,1,0) activeFromTo("c",0,1,1) activeFromTo("b",0,1,2).
Answer 2: activeFromTo("d",0,1,0) activeFromTo("b",0,2,1)
Answer 3: activeFromTo("c",0,1,0) activeFromTo("d",0,1,1) activeFromTo("d",1,0,2) activeFromTo("b",0,1,3)
...
Answer 29: activeFromTo("c",0,1,0) activeFromTo("b",0,1,1) activeFromTo("a",0,1,2)
Success reachability through ASP:

goal(activeProcess("d",2)).
Exhaustive analysis of the dynamics of Process Hitting through ASP ○ Reachability ○ ASP 1\textsuperscript{st} method

Dynamic analysis
Reachability through ASP

Success reachability through ASP:

goal(activeProcess("d",2)).
satisfaible(F,T) :- goal(F), instate(F,T).
:- not satisfaibleTot.
**Dynamic analysis**

Reachability through ASP

**Success reachability through ASP:**

```prolog
goal(activeProcess("d",2)).
satisfaible(F,T) :- goal(F), instate(F,T).
:- not satisfaibleTot.
time(0..N).
```

![Diagram showing reachability through ASP](attachment:image_url)
Success reachability through ASP:

\[
\text{goal}(\text{activeProcess}("d",2)).
\text{satisfaible}(F,T) :- \text{goal}(F), \text{instate}(F,T).
\text{:- not satisfaibleTot.}
\text{time}(0..N).
\text{predict N} \rightarrow \text{Inconvenient}
\]
Dynamic analysis
Reachability through ASP

Results for \((N = 2)\):
UNSATISFIABLE

Results for \((N = 3)\):
Answer 1: \texttt{activeFromTo(c,0,1,0), activeFromTo(d,0,1,1), activeFromTo(b,0,1,2), activeFromTo(d,1,2,3)}.
Answer 2: \texttt{activeFromTo("d",0,1,0) activeFromTo("c",0,1,1) activeFromTo("b",0,1,2) activeFromTo("d",1,2,3)}
**Success reachability through ASP iterative:**

\[
\text{goal(activeProcess("d","2").}
\]

#base

\[
\text{instate(F,0) :- init(F).}
\]

#cumulative t

\[
\text{playableAction(A, I, B, J, K,t), activeFromTo(B, J, K,t),}
\]
\[
\text{instate(activeProcess(A, I),t + 1)...}
\]

#volatile t

\[
\text{notSatisfaible(t) :- goal(F), not instate(F,t).}
\]
\[
\text{:- notSatisfaible(t).}
\]
Dynamic analysis
Reachability through ASP

Success reachability through ASP iterative:

Results:

Answer 1:  activeFromTo(c,0,1,0), activeFromTo(d,0,1,1),
activeFromTo(b,0,1,2), activeFromTo(d,1,2,3).
Answer 2:  activeFromTo("d",0,1,0) activeFromTo("c",0,1,1)
activeFromTo("b",0,1,2) activeFromTo("d",1,2,3)
Dynamic analysis
Reachability through ASP

Comparison:

Initializing biological models components and the objectives.

<table>
<thead>
<tr>
<th>Model</th>
<th>#sorts</th>
<th>#states</th>
<th>#steps</th>
<th>ASP</th>
<th>ASPi</th>
<th>PINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exemple</td>
<td>4</td>
<td>36</td>
<td>4</td>
<td>0.000s</td>
<td>0.000s</td>
<td>0.000s</td>
</tr>
<tr>
<td>ERBB</td>
<td>42</td>
<td>$2^{70}$</td>
<td>18</td>
<td>10.620s</td>
<td>5.020s</td>
<td>0.022s</td>
</tr>
<tr>
<td>tcrsig40</td>
<td>54</td>
<td>$2^{73}$</td>
<td>26</td>
<td>156.500s</td>
<td>127.250s</td>
<td>0.012s</td>
</tr>
</tbody>
</table>

Figure: Execution time of ASP methods (CLINGO et ICLINGO) and PINT applied for biological networks with a desktop computer (core i5 and 4GB RAM)
Dynamic analysis
Reachability through ASP

Comparison:

Method of Rocca et al.:

- ASP
- CTL properties with model checking (AF, EF, AG...)
- Transitions graph
Comparison:

Method of Rocca et al.:
- ASP
- CTL properties with model checking (AF, EF, AG...)
- Transitions graph

Comparison of the property EF

\[ \text{prop} = \text{EF} (l_0, goal) \]
Comparaison:

Example: Tail resorption of tadpole:
12 sorts, 42 process, 139 actions and 524,288 states.

\[ \text{prop} = EF(l_0, \text{goal}) \]
Comparaison:

Example: Tail resorption of tadpole:
12 sorts, 42 processes, 139 actions and 524,288 states.

\[ \text{prop} = \text{EF}(l_0, \text{goal}) \]

Network traduction:
- Transition graph: 3\text{min}6s
- Process Hitting: 0.346s
Comparaison:

Example: Tail resorption of tadpole:
12 sorts, 42 process, 139 actions and 524,288 states.

\[ \text{prop} = \text{EF}(l_0, \text{goal}) \]

Network traduction:
- Transition graph: 3min6s
- Process Hitting: 0.346s

Property verification:
- Rocca et al. method: 7min17s
- Our iterative method: 1.9s
Conclusion & Prospects

• New dynamic analysis of Process Hitting models:
  – Fixed point
  – Network evolution
  – Reachability

• Prospects:
  – Adaptation on other models (PN, model of Thomas…)
  – Eliminating cycles
  – Search attractors
  – Reverse reachability \((\text{goal} \rightarrow I_0?)\)
Bibliography


Thanks for your attention