4th International Workshop on Interactions between Computer Science and Biology

Under-approximation of Reachability in Multivalued Asynchronous Networks

Maxime FOLSCHETTE
MeForBio / IRCCyN / École Centrale de Nantes (Nantes, France)
maxime.folschette@irccyn.ec-nantes.fr
http://www.irccyn.ec-nantes.fr/~folschet/

Joint work with:
Loïc PAULEVÉ, Morgan MAGNIN, Olivier ROUX
**MeForBio team:**
Algebraic modelling to study complex dynamical biological systems
MeForBio team:
Algebraic modelling to study complex dynamical biological systems

1) Asynchronous Discrete Networks (ADN)
   Convenient to model biological systems

2) Process Hitting (PH)
   Cannot accurately describe ADNs

3) Enhancing PH with priorities
   To efficiently compute reachability in ADNs
The Asynchronous Discrete Networks (ADN)

[De Jong in *Journal of Computational Biology*, 2002]

- A set of components  \( N = \{a, b, z\} \)
The Asynchronous Discrete Networks (ADN)
[De Jong in *Journal of Computational Biology*, 2002]

- A set of components $N = \{a, b, z\}$
- A set of expression levels for each component $z \in F^z = \mathbb{K}$
- The set of global states $F = F^a \times F^b \times F^z$
The Asynchronous Discrete Networks (ADN)

[De Jong in *Journal of Computational Biology*, 2002]

- A set of components \( N = \{a, b, z\} \)
- A set of expression levels for each component \( z \in F^z = [0; 2] \)
- The set of global states \( F = F^a \times F^b \times F^z \)
- An evolution function for each component \( f^z : F \rightarrow F^z \)

\[
\begin{array}{c|c}
    b & f^a(b) \\
    \hline
    0 & 1 \\
    1 & 0 \\
\end{array}
\quad
\begin{array}{c|c|c}
    a & b & f^b(a, b) \\
    \hline
    0 & 0 & 1 \\
    0 & 1 & 1 \\
    1 & 0 & 0 \\
    1 & 1 & 1 \\
\end{array}
\quad
\begin{array}{c|c|c}
    a & b & f^z(a, b) \\
    \hline
    0 & 0 & 0 \\
    0 & 1 & 1 \\
    1 & 0 & 1 \\
    1 & 1 & 2 \\
\end{array}
\]
Asynchronous Discrete Networks (ADN)

State Graph: $G = (\mathcal{F}, \mathcal{E})$, where one component evolves at a time given its function $f^a$

$$(x, y) \in \mathcal{E} \iff \exists a \in \mathbb{N}, y^a = f^a(x) \land \forall b \neq a, y^b = x^b$$
Under-approximation of Reachability in Multivalued Asynchronous Networks

Asynchronous Discrete Networks (ADN)

The Asynchronous Discrete Networks (ADN)

State Graph: \( G = (F, E) \), where one component evolves at a time given its function \( f^a \)

\[(x, y) \in E \iff \exists a \in N, y^a = f^a(x) \land \forall b \neq a, y^b = x^b\]

Size of the State Graph:

\[|F| = \prod_{a \in N} |F^a| \geq 2^{|N|}\]

\( \rightarrow \text{Exponential in the number } |N| \text{ of components} \)
The Asynchronous Discrete Networks (ADN)

State Graph: $G = (F, E)$, where one component evolves at a time given its function $f^a$

$$(x, y) \in E \iff \exists a \in N, y^a = f^a(x) \land \forall b \neq a, y^b = x^b$$

Size of the State Graph: $|F| = \prod_{a \in N} |F^a| \geq 2^{|N|}$

$\rightarrow$ **Exponential** in the number $|N|$ of components

Some works give a link between the structure and the behaviour of an ADN

- **Thomas’ conjecture** (condition for multiple fixed points or attractive cycle)
  - Boolean: [Remy, Ruet, Thieffry in *Advances in Applied Mathematics*, 2008]
  - Multivalued: [Richard, Comet in *Discrete Applied Mathematics*, 2007]
The Asynchronous Discrete Networks (ADN)

State Graph: $G = (\mathcal{F}, \mathcal{E})$, where one component evolves at a time given its function $f^a$

$$(x, y) \in \mathcal{E} \iff \exists a \in N, y^a = f^a(x) \land \forall b \neq a, y^b = x^b$$

Size of the State Graph: $|\mathcal{F}| = \prod_{a \in N} |\mathcal{F}^a| \geq 2^{|N|}$

→ **Exponential** in the number $|N|$ of components

Some works give a link between the structure and the behaviour of an ADN

- **Thomas’ conjecture** (condition for multiple fixed points or attractive cycle)
  - Boolean: [Remy, Ruet, Thieffry in *Advances in Applied Mathematics*, 2008]
  - Multivalued: [Richard, Comet in *Discrete Applied Mathematics*, 2007]

But methods related to reachability rely on the State Graph

e.g.: Starting from $(a, b, z) = (0, 0, 0)$, can the system reach $z = 2$?

- **Temporal logics**
  - CTL: [Bernot, Comet, Richard, Guespin in *Journal of Theoretical Biology*, 2004]
  - LTL: [Ito, Izumi, Hagihara, Yonezaki in *BioInformatics and BioEngineering*, 2010]
The Process Hitting modeling

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

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

- **Processes:** local states / levels of expression \( z_0, z_1, z_2 \)
- **States:** sets of active processes
- **Actions:** dynamics

\[ b \rightarrow z_0 \uparrow z_1, a_0 \rightarrow a_0 \uparrow a_1, a_1 \rightarrow z_1 \uparrow z_2 \]
The Process Hitting framework (PH)

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

**Sorting**: components $a, b, z$

**Processes**: local states / levels of expression $z_0, z_1, z_2$
The Process Hitting modeling

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

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

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

States: sets of active processes \( \langle a_0, b_1, z_0 \rangle \)
The Process Hitting framework (PH)

The Process Hitting modeling

[Paulevé, Magnin, Roux in Transactions on Computational Systems Biology, 2011]

**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 \( b_1 \rightarrow z_0 \uparrow z_1, a_0 \rightarrow a_0 \uparrow a_1, a_1 \rightarrow z_1 \uparrow z_2 \)
The Process Hitting modeling

[Paulevé, Magnin, Roux in Transactions on Computational Systems Biology, 2011]

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 $b_1 \rightarrow z_0 \uparrow z_1, a_0 \rightarrow a_0 \uparrow a_1, a_1 \rightarrow z_1 \uparrow z_2$
The Process Hitting modeling

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology, 2011*]

**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  \( b_1 \rightarrow z_0 \uparrow z_1 \), \( a_0 \rightarrow a_0 \uparrow a_1 \), \( a_1 \rightarrow z_1 \uparrow z_2 \)
The Process Hitting modeling

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

**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 $b_1 \rightarrow z_0 \uparrow z_1$, $a_0 \rightarrow a_0 \uparrow a_1$, $a_1 \rightarrow z_1 \uparrow z_2$
**The Process Hitting modeling**

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology, 2011*]

**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 $b_1 \rightarrow z_0 \uparrow z_1$, $a_0 \rightarrow a_0 \uparrow a_1$, $a_1 \rightarrow z_1 \uparrow z_2$
Static analysis: successive reachability of processes

Static analysis: successive reachability of processes


• Initial context

\[ \langle a_1, \{b_0, b_1\}, c_0, d_0 \rangle \]
Static analysis: successive reachability of processes


- **Initial context**
  \(\langle a_1, \{b_0, b_1\}, c_0, d_0\rangle\)

- **Objectives**
  \([\overset{\mathcal{R}}{d_1} :: \overset{\mathcal{R}}{b_1} :: \overset{\mathcal{R}}{d_2} ]\)
Static analysis: successive reachability of processes


- Initial context
  \[ \langle a_1, \{ b_0, b_1 \}, c_0, d_0 \rangle \]

- Objectives
  \[ \langle \vdash d_1 :: \vdash b_1 :: \vdash d_2 \rangle \]
  \[ \langle \vdash d_2 \rangle \]
Static analysis: successive reachability of processes


- Initial context
  \( \langle a_1, \{ b_0, b_1 \}, c_0, d_0 \rangle \)

- Objectives
  \[
  [ \uparrow d_1 :: \uparrow b_1 :: \uparrow d_2 ]
  \]
  \[
  [ \uparrow d_2 ]
  \]

→ Concretization of the objective = scenario

\[
\begin{align*}
  a_0 & \rightarrow c_0 \uparrow c_1 :: b_0 \rightarrow d_0 \uparrow d_1 :: c_1 \rightarrow b_0 \uparrow b_1 :: b_1 \rightarrow d_1 \uparrow d_2
\end{align*}
\]
Static analysis: successive reachability of processes


- **Initial context**
  \[
  \langle a_1, \{b_0, b_1\}, c_0, d_0 \rangle
  \]

- **Objectives**
  \[
  [ \overset{d_1}{} \overset{b_1}{} \overset{d_2}{} ]
  \]

\[
\rightarrow \text{Concretization of the objective } = \text{ scenario}
\]

\[
a_0 \rightarrow c_0 \overset{c_1}{} \overset{b_0}{} \overset{d_0}{} \overset{d_1}{} \overset{c_1}{} \rightarrow b_0 \overset{b_1}{} \overset{d_1}{} \overset{b_1}{} \overset{d_2}{}
\]
Static analysis: successive reachability of processes


- Initial context
  \( \langle a_1, \{b_0, b_1\}, c_0, d_0 \rangle \)

- Objectives
  \[ \vdash d_1 :: \vdash b_1 :: \vdash d_2 \]
  \[ \vdash d_2 \]

→ Concretization of the objective = scenario

\[ a_0 \rightarrow c_0 \vdash c_1 :: b_0 \rightarrow d_0 \vdash d_1 :: c_1 \rightarrow b_0 \vdash b_1 :: b_1 \rightarrow d_1 \vdash d_2 \]
Static analysis: successive reachability of processes

[Paulevé, Magnin, Roux in Mathematical Structures in Computer Science, 2012]

- Initial context
\[ \langle a_1, \{ b_0, b_1 \}, c_0, d_0 \rangle \]

- Objectives
\[ [ \overset{\top} d_1 :: \overset{\top} b_1 :: \overset{\top} d_2 ] \]
\[ [ \overset{\top} d_2 ] \]

→ Concretization of the objective = scenario
\[ a_0 \rightarrow c_0 \overset{\top} c_1 :: b_0 \rightarrow d_0 \overset{\top} d_1 :: c_1 \rightarrow b_0 \overset{\top} b_1 :: \overset{\top} b_1 \rightarrow d_1 \overset{\top} d_2 \]
Static analysis: successive reachability of processes
[Paulevé, Magnin, Roux in Mathematical Structures in Computer Science, 2012]

- Initial context
  \[ \langle a_1, \{ b_0, b_1 \}, c_0, d_0 \rangle \]

- Objectives
  \[ [ \overset{\triangleright}{d_1} :: \overset{\triangleright}{b_1} :: \overset{\triangleright}{d_2} ] \]
  \[ [ \overset{\triangleright}{d_2} ] \]

→ Concretization of the objective = scenario
\[ a_0 \rightarrow c_0 \overset{\triangleright}{c_1} :: b_0 \rightarrow d_0 \overset{\triangleright}{d_1} :: c_1 \rightarrow b_0 \overset{\triangleright}{b_1} :: b_1 \rightarrow d_1 \overset{\triangleright}{d_2} \]
Over- and Under-approximations


Static analysis by abstractions:

→ Directly checking an objective sequence $R$ is hard (*State Graph*)
→ Rather check the approximations $P$ and $Q$, where $P \Rightarrow R \Rightarrow Q$:

```
+-------------------+-------------------+
|       Exact solution       |
+-------------------+-------------------+
|                R                |
+-------------------+-------------------+
```
Over- and Under-approximations


Static analysis by abstractions:

→ Directly checking an objective sequence $R$ is hard (**State Graph**)
→ Rather check the approximations $P$ and $Q$, where $P \Rightarrow R \Rightarrow Q$:

- Over-Approximation
- Exact solution
- $R$
- $\neg Q$
Over- and Under-approximations


Static analysis by abstractions:

→ Directly checking an objective sequence $R$ is hard (**State Graph**)
→ Rather check the approximations $P$ and $Q$, where $P \Rightarrow R \Rightarrow Q$:

![Diagram showing over-approximation and exact solution](image-url)
Over- and Under-approximations

[Paulevé, Magnin, Roux in Mathematical Structures in Computer Science, 2012]

Static analysis by abstractions:

→ Directly checking an objective sequence \( R \) is hard (State Graph)
→ Rather check the approximations \( P \) and \( Q \), where \( P \Rightarrow R \Rightarrow Q \):

\[
\begin{align*}
\text{Over-Approximation} & \quad \neg Q \\
\text{Exact solution} & \quad \text{Under-Approximation}
\end{align*}
\]
Over- and Under-approximations

[Paulevé, Magnin, Roux in Mathematical Structures in Computer Science, 2012]

Static analysis by abstractions:

→ Directly checking an objective sequence $R$ is hard (State Graph)
→ Rather check the approximations $P$ and $Q$, where $P \Rightarrow R \Rightarrow Q$:

Over-Approximation

Under-Approximation

Exact solution
Over- and Under-approximations

[Paulevé, Magnin, Roux in Mathematical Structures in Computer Science, 2012]

Static analysis by abstractions:

→ Directly checking an objective sequence $R$ is hard (State Graph)

→ Rather check the approximations $P$ and $Q$, where $P \Rightarrow R \Rightarrow Q$:
Over- and Under-approximations


Static analysis by abstractions:

→ Directly checking an objective sequence $R$ is hard (*State Graph*)
→ Rather check the approximations $P$ and $Q$, where $P \Rightarrow R \Rightarrow Q$:

![Diagram](image)
Over- and Under-approximations

[Paulevé, Magnin, Roux in Mathematical Structures in Computer Science, 2012]

Static analysis by abstractions:

→ Directly checking an objective sequence $R$ is hard (State Graph)
→ Rather check the approximations $P$ and $Q$, where $P \Rightarrow R \Rightarrow Q$:

Computing $P$ or $Q$ is polynomial in the number of sorts and exponential in the number of processes in each sort
→ Efficient for big models with few levels of expression
Under-approximation of Reachability in Multivalued Asynchronous Networks

- **The Process Hitting framework (PH)**

**Under-approximation**

- Sufficient condition:
  - no cycle
  - each objective has a solution

- **Required process**
- **Objective**
- **Solution to an objective**
Under-approximation of Reachability in Multivalued Asynchronous Networks

The Process Hitting framework (PH)

Sufficient condition:
- no cycle
- each objective has a solution

**Required process**

- $d_2$

**Objective**

- $d_0 \rightarrow^* d_2$

**Solution to an objective**

- $d_0 \rightarrow^* d_2$

- $b_0 \rightarrow^* b_0 \rightarrow^* a_1 \rightarrow^* a_1$

- $b_1 \rightarrow^* b_1 \rightarrow^* c_1 \rightarrow^* c_1$
Under-approximation of Reachability in Multivalued Asynchronous Networks

The Process Hitting framework (PH)

Under-approximation

Sufficient condition:
- no cycle
- each objective has a solution

$R$ is true

Required process

Objective

Solution to an objective
Under-approximation of Reachability in Multivalued Asynchronous Networks

The Process Hitting framework (PH)

Under-approximation

Sufficient condition:
- no cycle
- each objective has a solution
Under-approximation of Reachability in Multivalued Asynchronous Networks

The Process Hitting framework (PH)

Under-approximation

Sufficient condition:
- no cycle
- each objective has a solution

Inconclusive

Under-approximation

Maxime FOLSCHETTE
The Process Hitting framework (PH)

Implementation in PINT

Existing free OCaml library: PINT

→ Compiler + tools for Process Hitting models
→ Documentation & examples: http://processhitting.wordpress.com/
Implementation in PINT

Existing free OCaml library: PINT

→ Compiler + tools for Process Hitting models
→ Documentation & examples: http://processhitting.wordpress.com/

Computation time for various reachability analyses:

<table>
<thead>
<tr>
<th>Model</th>
<th>Sorts</th>
<th>Procs</th>
<th>Actions</th>
<th>States</th>
<th>Biocham(^1)</th>
<th>libddd(^2)</th>
<th>PINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>egfr20</td>
<td>35</td>
<td>196</td>
<td>670</td>
<td>2(^{64})</td>
<td>[3s – ∞]</td>
<td>[1s – 150s]</td>
<td>0.007s</td>
</tr>
<tr>
<td>tcrsig40</td>
<td>54</td>
<td>156</td>
<td>301</td>
<td>2(^{73})</td>
<td>[1s – ∞]</td>
<td>[0.6s – ∞]</td>
<td>0.004s</td>
</tr>
<tr>
<td>tcrsig94</td>
<td>133</td>
<td>448</td>
<td>1124</td>
<td>2(^{194})</td>
<td>∞</td>
<td>∞</td>
<td>0.030s</td>
</tr>
<tr>
<td>egfr104</td>
<td>193</td>
<td>748</td>
<td>2356</td>
<td>2(^{320})</td>
<td>∞</td>
<td>∞</td>
<td>0.050s</td>
</tr>
</tbody>
</table>

\(^1\) Inria Paris-Rocquencourt/Contraintes
\(^2\) LIP6/Move

egrf20: [Epidermal Growth Factor Receptor, by Özgür Sahin et al.]
egrf104: [Epidermal Growth Factor Receptor, by Regina Samaga et al.]
tcrsig40: [T-Cell Receptor Signaling, by Steffen Klamt et al.]
tcrsig94: [T-Cell Receptor Signaling, by Julio Saez-Rodriguez et al.]
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology, 2011*]

**Cooperation** between $a_1$ and $b_1$: \[ a_1 \land b_1 \rightarrow z_0 \uparrow z_1 \]
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

**Cooperation** between \( a_1 \) and \( b_1 \):

\[
 a_1 \land b_1 \rightarrow z_0 \uparrow z_1
\]
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology, 2011*]

**Cooperation** between $a_1$ and $b_1$: $a_1 \land b_1 \rightarrow z_0 \uparrow z_1$
Adding cooperations

[Paulevé, Magnin, Roux in Transactions on Computational Systems Biology, 2011]

**Cooperation** between $a_1$ and $b_1$: $a_1 \land b_1 \rightarrow z_0 \uparrow z_1$
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology, 2011*]

**Cooperation** between $a_1$ and $b_1$: $a_1 \land b_1 \rightarrow z_0 \uparrow z_1$

Solution: a **cooperative sort** $ab$ to express $a_1 \land b_1$
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology, 2011*]

**Cooperation** between \( a_1 \) and \( b_1 \): \( a_1 \land b_1 \rightarrow z_0 \uparrow z_1 \)

Solution: a *cooperative sort* \( ab \) to express \( a_1 \land b_1 \)
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology, 2011*]

**Cooperation** between $a_1$ and $b_1$: $a_1 \land b_1 \rightarrow z_0 \uparrow z_1$

Solution: a *cooperative sort* $ab$ to express $a_1 \land b_1$
Adding cooperations

[Paulevé, Magnin, Roux in Transactions on Computational Systems Biology, 2011]

**Cooperation** between \( a_1 \) and \( b_1 \): \( a_1 \land b_1 \rightarrow z_0 \uparrow z_1 \)

Solution: a **cooperative sort** \( ab \) to express \( a_1 \land b_1 \)
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

Cooperation between $a_1$ and $b_1$: $a_1 \land b_1 \rightarrow z_0 \uparrow z_1$

Solution: a cooperative sort $ab$ to express $a_1 \land b_1$

Constraint: each configuration is represented by one process $a_1 \land b_1 \Rightarrow ab_{11}$
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology, 2011*]

**Cooperation** between \( a_1 \) and \( b_1 \):

\[
a_1 \land b_1 \to z_0 \uparrow z_1
\]

Solution: a **cooperative sort** \( ab \) to express \( a_1 \land b_1 \)

Constraint: each configuration is represented by one process

\[
a_1 \land b_1 \Rightarrow ab_{11}
\]
Adding cooperations

[Paulevé, Magnin, Roux in Transactions on Computational Systems Biology, 2011]

**Cooperation** between $a_1$ and $b_1$: $a_1 \land b_1 \rightarrow z_0 \uparrow z_1$

Solution: a **cooperative sort** $ab$ to express $a_1 \land b_1$

Constraint: each configuration is represented by one process $a_1 \land b_1 \Rightarrow ab_{11}$
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology, 2011*]

**Cooperation** between $a_1$ and $b_1$: $a_1 \land b_1 \rightarrow z_0 \uparrow z_1$

Solution: a **cooperative sort** $ab$ to express $a_1 \land b_1$

Constraint: each configuration is represented by one process $a_1 \land b_1 \Rightarrow ab_{11}$
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

**Cooperation** between $a_1$ and $b_1$: $a_1 \land b_1 \rightarrow z_0 \uparrow z_1$

Solution: a **cooperative sort** $ab$ to express $a_1 \land b_1$

Constraint: each configuration is represented by one process $a_1 \land b_1 \Rightarrow ab_{11}$
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology, 2011*]

Cooperation between $a_1$ and $b_1$: $a_1 \land b_1 \rightarrow z_0 \uparrow z_1$

Solution: a cooperative sort $ab$ to express $a_1 \land b_1$

Constraint: each configuration is represented by one process $a_1 \land b_1 \Rightarrow ab_{11}$
Adapting the expressivity of PH

**Drawback:** Cooperations are too “loose” to be as expressive as ADN.
Adapting the expressivity of PH

Drawback: Cooperations are too “loose” to be as expressive as ADN.
Adapting the expressivity of PH

**Drawback:** Cooperations are too “loose” to be as expressive as ADN.
Adapting the expressivity of PH

**Drawback**: Cooperations are too “loose” to be as expressive as ADN.
Drawback: Cooperations are too “loose” to be as expressive as ADN.

\[ \langle a_0, b_0, ab_{00}, z_0 \rangle \]
**Drawback:** Cooperations are too “loose” to be as expressive as ADN.

\[ \langle a_0, b_0, ab_{00}, z_0 \rangle \]
**Drawback:** Cooperations are too “loose” to be as expressive as ADN.

\[ \langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \]
Drawback: Cooperations are too “loose” to be as expressive as ADN.

\[
\langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle
\]
Drawback: Cooperations are too “loose” to be as expressive as ADN.

\[
\langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0 \rangle
\]
Drawback: Cooperations are too “loose” to be as expressive as ADN.

\[ \langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_1, ab_{10}, z_0 \rangle \]
Drawback: Cooperations are too “loose” to be as expressive as ADN.

\[
\langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0 \rangle \\
\rightarrow \langle a_0, b_1, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_1, ab_{11}, z_0 \rangle
\]
Drawback: Cooperations are too “loose” to be as expressive as ADN.

\[ \langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_1, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_1, ab_{11}, z_0 \rangle \rightarrow \langle a_0, b_1, ab_{11}, z_1 \rangle \]

The cooperativity should be: \[ a_1 \land b_1 \text{ simultaneously} \] i.e. “in the same state”

but the model behaves like: \[ P(a_1) \land P(b_1) \] with \( P \) = “previously”
Adapting the expressivity of PH

- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)
Adapting the expressivity of PH

- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

⇒ Whenever a regular action is played, all cooperative sorts are already updated
Adapting the expressivity of PH

- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

⇒ Whenever a regular action is played, all cooperative sorts are already updated

\[ \langle a_0, b_0, ab_{00}, z_0 \rangle \]
Adapting the expressivity of PH

- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

⇒ Whenever a regular action is played, all cooperative sorts are already updated

\[ \langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \]
Adapting the expressivity of PH

- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

⇒ Whenever a regular action is played, all cooperative sorts are already updated

\[ \langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle \]
Adapting the expressivity of PH

- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

⇒ Whenever a regular action is played, all cooperative sorts are already updated

\[ \langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0 \rangle \]
Adapting the expressivity of PH

- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

⇒ Whenever a regular action is played, all cooperative sorts are already updated

\[
\langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0 \rangle \\
\rightarrow \langle a_0, b_0, ab_{00}, z_0 \rangle
\]
Prioritise actions updating cooperative sorts (non-biological actions)
All other actions remain unprioritised (evolutions with delays)

⇒ Whenever a regular action is played, all cooperative sorts are already updated

\[
\langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0 \rangle \\
\rightarrow \langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_0, b_1, ab_{00}, z_0 \rangle
\]
Under-approximation of Reachability in Multivalued Asynchronous Networks

Adapting the expressivity of PH

- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

⇒ Whenever a regular action is played, all cooperative sorts are already updated

\[
\begin{align*}
\langle a_0, b_0, ab_{00}, z_0 \rangle &\rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \\
&\rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle \\
&\rightarrow \langle a_0, b_0, ab_{00}, z_0 \rangle \\
&\rightarrow \langle a_0, b_1, ab_{00}, z_0 \rangle \\
&\rightarrow \langle a_0, b_1, ab_{01}, z_0 \rangle
\end{align*}
\]
Static analysis with prioritised actions

**Sufficient condition:**
- no cycle
- each objective has a solution
- coherent edges

```plaintext
\[ z_1 \rightarrow z_0 \overset{\ast}{\rightarrow} z_1 \rightarrow \overrightarrow{ab_{11}} \]
```

```
\[
\begin{align*}
  a_1 & \overset{\ast}{\rightarrow} a_1 & \rightarrow & \circ \\
  a_1 & \rightarrow & a_0 & \overset{\ast}{\rightarrow} a_1 & \rightarrow & \circ \\
  b_0 & \rightarrow & b_0 & \overset{\ast}{\rightarrow} b_0 & \rightarrow & \circ \\
  b_1 & \rightarrow & b_0 & \overset{\ast}{\rightarrow} b_1 & \rightarrow & \circ \\
  b_1 & \rightarrow & b_1 & \overset{\ast}{\rightarrow} b_0 & \rightarrow & \circ \\
  a_0 & \rightarrow & a_0 & \overset{\ast}{\rightarrow} a_0 & \rightarrow & \circ \\
  a_1 & \overset{\ast}{\rightarrow} a_0 & \rightarrow & \circ \\
\end{align*}
\]
```
Static analysis with prioritised actions

**Sufficient condition:**

- no cycle
- each objective has a solution
- coherent edges
Static analysis with prioritised actions

**Sufficient condition:**

- no cycle
- each objective has a solution
- coherent edges

![Diagram](image)

- **Required process**
- **Objective**
- **Solution to an objective**
- **Solution to a prioritised cooperative sort process**
Static analysis with prioritised actions

**Sufficient condition:**

- no cycle
- each objective has a solution
- coherent edges

\[
\begin{align*}
  \text{Required process} & \quad \text{Objective} \\
  \text{Solution to an objective} & \quad \text{Solution to a prioritised cooperative sort process}
\end{align*}
\]
**Static analysis with prioritised actions**

**Sufficient condition:**

- no cycle
- each objective has a solution
- coherent edges

\[
\begin{align*}
z_1 & \rightarrow z_0 \overset{?}{\rightarrow} z_1 \\
\rightarrow & \quad ab_{11}
\end{align*}
\]

**Inconclusive**

- \( a_1 \overset{?}{\rightarrow} a_1 \rightarrow \bullet \)
- \( b_1 \overset{?}{\rightarrow} b_0 \rightarrow \bullet \)
- \( a_0 \overset{?}{\rightarrow} a_0 \rightarrow \bullet \)
- \( b_0 \overset{?}{\rightarrow} b_0 \rightarrow \bullet \)
- \( b_1 \overset{?}{\rightarrow} b_1 \rightarrow \bullet \)
- \( a_1 \overset{?}{\rightarrow} a_0 \rightarrow \bullet \)

**Required process**
- \( z_1 \)

**Objective**
- \( z_0 \overset{?}{\rightarrow} z_1 \)

**Solution to an objective**
- \( \bullet \)

**Solution to a prioritised cooperative sort process**
- \( ab_{11} \)
Implementation

Complexity:

- **Building the graph:**
  - Polynomial in the number of sorts
  - Exponential in the number of processes in each sort

- **Analysing the graph:**
  - Polynomial in the size of the graph
Complexity:

- **Building the graph:**
  - Polynomial in the number of sorts
  - Exponential in the number of processes in each sort

- **Analysing the graph:**
  - Polynomial in the size of the graph

<table>
<thead>
<tr>
<th>Model</th>
<th>Sorts</th>
<th>Procs</th>
<th>Actions</th>
<th>States</th>
<th>libddd(^1)</th>
<th>GINsim(^2)</th>
<th>PINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>egfr20</td>
<td>35</td>
<td>196</td>
<td>670</td>
<td>(2^{64})</td>
<td>&lt;1s</td>
<td>0.35s</td>
<td></td>
</tr>
<tr>
<td>tcrsig40</td>
<td>54</td>
<td>156</td>
<td>301</td>
<td>(2^{73})</td>
<td>∞</td>
<td>0.2s</td>
<td></td>
</tr>
<tr>
<td>tcrsig94</td>
<td>133</td>
<td>448</td>
<td>1124</td>
<td>(2^{194})</td>
<td>[13min – ∞]</td>
<td>0.8s</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) LIP6/Move  
\(^2\) TAGC/IGC

**egfr20**: [Epidermal Growth Factor Receptor, by Özgür Sahin et al.]

**tcrsig40**: [T-Cell Receptor Signaling, by Steffen Klamt et al.]

**tcrsig94**: [T-Cell Receptor Signaling, by Julio Saez-Rodriguez et al.]
Summary

- The Process Hitting framework
  - Restricted concurrent actions
  - Efficient static analysis on biological models (few expression levels)

- But raw Process Hitting is insufficient to models ADNs
  - How to represent cooperations?
  - Cooperative sorts only represent a combination of past states

- Solution: prioritised actions
  - Accurate cooperative sorts
  - Expressivity of ADN is reached
Conclusion

- **Achieved:**
  - Rise the expressivity of PH
  - Efficient reachability analysis in ADNs

- **Value:**
  - Model a whole class of ADNs in one PH model
  - Efficiently analyse reachability for the whole class
  - Refine the PH model to match desired behaviour
  - Infer the underlying class of ADNs

  [Folschette, Paulevé, Inoue, Magnin, Roux in *Computational Methods in Systems Biology, 2012*]
Conclusion

- Achieved:
  - Rise the expressivity of PH
  - Efficient reachability analysis in ADNs

- Value:
  - Model a whole class of ADNs in one PH model
  - Efficiently analyse reachability for the whole class
  - Refine the PH model to match desired behaviour
  - Infer the underlying class of ADNs


Outlook

- Allow prioritised actions even for biological evolutions
- Allow $n > 2$ classes of priority
  - Model actions with delays by using priorities
Bibliography


Thank you
Under-approximation of Reachability in Multivalued Asynchronous Networks

Annex: Graphs of local causality

Under-approximation

- Sufficient condition:
  - no cycle
  - each objective has a solution

- Required process
- Objective
- Solution to an objective

Maxime FOLSCHETTE
CS2Bio'13 — 2013/06/06
Sufficient condition:
- no cycle
- each objective has a solution
Under-approximation of Reachability in Multivalued Asynchronous Networks

Annex: Graphs of local causality

Under-approximation

Sufficient condition:
- no cycle
- each objective has a solution

\( R \) is true
Under-approximation of Reachability in Multivalued Asynchronous Networks

Annex: Graphs of local causality

Under-approximation

Sufficient condition:
- no cycle
- each objective has a solution
Sufficient condition:

- no cycle
- each objective has a solution

Inconclusive
Over-approximation

Necessary condition:

\[ a_0 \xrightarrow{\perp} a_1 \]

\[ b_0 \xrightarrow{\hat{\theta}} b_2 \]

\[ c_0 \xrightarrow{\hat{\theta}} c_1 \]

\[ d_1 \xrightarrow{\hat{\theta}} d_1 \]

\[ d_2 \]

\[ b_1 \xrightarrow{\hat{\theta}} b_1 \]
Over-approximation

**Necessary condition:**
There exists a traversal with no cycle
- objective $\rightarrow$ follow **one** solution
- solution $\rightarrow$ follow **all** processes
- process $\rightarrow$ follow **all** objectives
Over-approximation

Necessary condition:
There exists a traversal with no cycle
- objective → follow one solution
- solution → follow all processes
- process → follow all objectives
Over-approximation

**Necessary condition:**
There exists a traversal with no cycle
- objective → follow **one** solution
- solution → follow **all** processes
- process → follow **all** objectives

$R$ is false
Over-approximation

**Necessary condition:**
There exists a traversal with no cycle
- objective → follow **one** solution
- solution → follow **all** processes
- process → follow **all** objectives
Necessary condition:
There exists a traversal with no cycle
- objective → follow one solution
- solution → follow all processes
- process → follow all objectives

Inconclusive
Over-approximation

Necessary condition:
There exists a traversal with no cycle
- objective $\rightarrow$ follow one solution
- solution $\rightarrow$ follow all processes
- process $\rightarrow$ follow all objectives

Inconclusive