Journée MDSC

Analysing Biological Networks with Exhaustive and Abstract Methods

Analyse des réseaux biologiques par des méthodes exhaustives et d'approximation

Maxime FOLSCHETTE

MDSC team / Bioinfo project / I3S laboratory / University of Nice Sophia Antipolis
`maxime.folschette@i3s.unice.fr`
`http://maxime.folschette.name/`

2016/05/24

Overview of This Presentation

Frameworks: the models we will talk about

- **Thomas modeling** (historically widespread)
- **Asynchronous Automata Networks** (generalization)

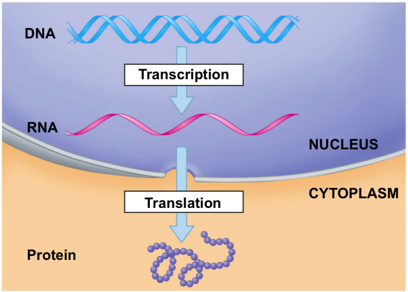
Exhaustive analyses: classical model-checking approaches with a high complexity

- Modal logic with an explicit fixed point: **μ-calculus**
- Logic programming: **Answer Set Programming**

Static analyses: approximations of the dynamics for lower complexity

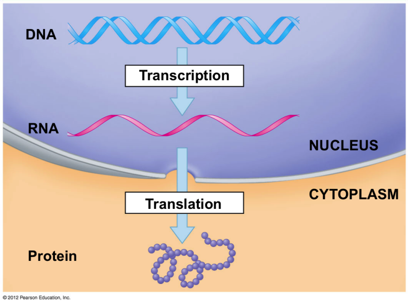
- **Classical results** of static analysis
- **Abstract interpretation**: a finer approach

Abstractions of the Representation



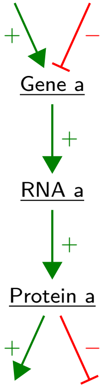
© 2012 Pearson Education, Inc.

Abstractions of the Representation

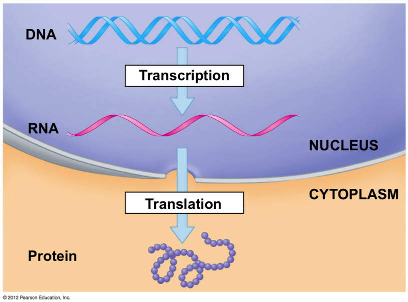


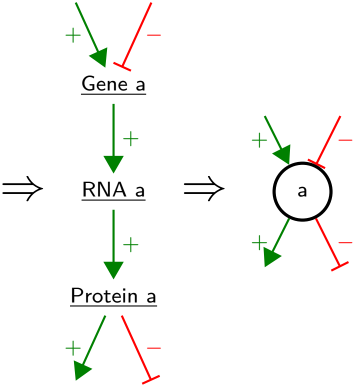
© 2012 Pearson Education, Inc.

$\Rightarrow$



Abstractions of the Representation

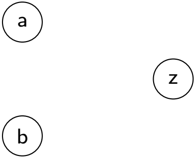




Discrete Networks / Thomas Modeling

[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

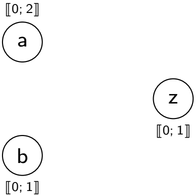
- A set of components $N = \{a, b, z\}$



Discrete Networks / Thomas Modeling

[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

- A set of components $N = \{a, b, z\}$
- A set of discrete expression levels for each component $a \in \mathbb{F}^a = [\![0; 2]\!]$
- The set of global states $\mathbb{F} = \mathbb{F}^a \times \mathbb{F}^b \times \mathbb{F}^z$



Discrete Networks / Thomas Modeling

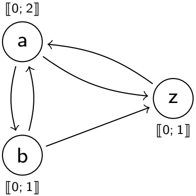
[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

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

a	$f^b(a)$
0	**0**
1	**1**
2	**1**

z	b	$f^a(z, b)$
0	0	**1**
0	1	**0**
1	0	**1**
1	1	**2**

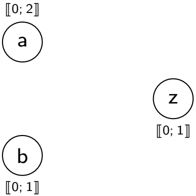
a	b	$f^z(a, b)$
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**0**
2	0	**0**
2	1	**1**



Discrete Networks / Thomas Modeling

[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

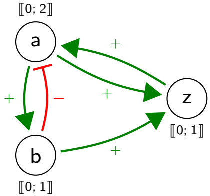
- A set of components $N = \{a, b, z\}$
- A set of discrete expression levels for each component $a \in \mathbb{F}^a = [\![0; 2]\!]$
- The set of global states $\mathbb{F} = \mathbb{F}^a \times \mathbb{F}^b \times \mathbb{F}^z$



Discrete Networks / Thomas Modeling

[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

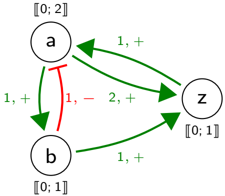
- A set of components $N = \{a, b, z\}$
- A set of discrete expression levels for each component $a \in \mathbb{F}^a = [\![0;2]\!]$
- The set of global states $\mathbb{F} = \mathbb{F}^a \times \mathbb{F}^b \times \mathbb{F}^z$
- Signs on the edges $a \xrightarrow{+} z$



Discrete Networks / Thomas Modeling

[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

- A set of components $N = \{a, b, z\}$
- A set of discrete expression levels for each component $a \in \mathbb{F}^a = [\![0;2]\!]$
- The set of global states $\mathbb{F} = \mathbb{F}^a \times \mathbb{F}^b \times \mathbb{F}^z$
- Signs on the edges $a \xrightarrow{+} z$ or signs + thresholds $a \xrightarrow{2,+} z$



Discrete Networks / Thomas Modeling

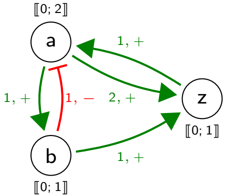
[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

- A set of components $N = \{a, b, z\}$
- A set of discrete expression levels for each component $a \in \mathbb{F}^a = [\![0;2]\!]$
- The set of global states $\mathbb{F} = \mathbb{F}^a \times \mathbb{F}^b \times \mathbb{F}^z$
- Signs on the edges $a \xrightarrow{+} z$ or signs + thresholds $a \xrightarrow{2,+} z$
- Discrete parameters / evolution functions $f^a : \mathbb{F} \to \mathbb{F}^a$

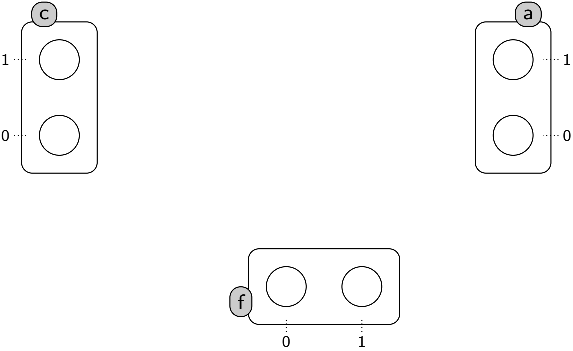
a	$f^b(a)$
0	**0**
1	**1**
2	**1**

z	b	$f^a(z,b)$
0	0	**1**
0	1	**0**
1	0	**1**
1	1	**2**

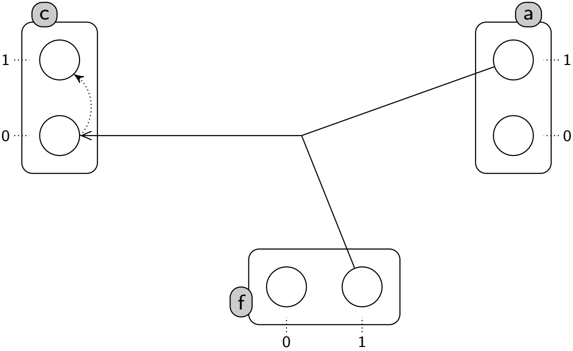
a	b	$f^z(a,b)$
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**0**
2	0	**0**
2	1	**1**



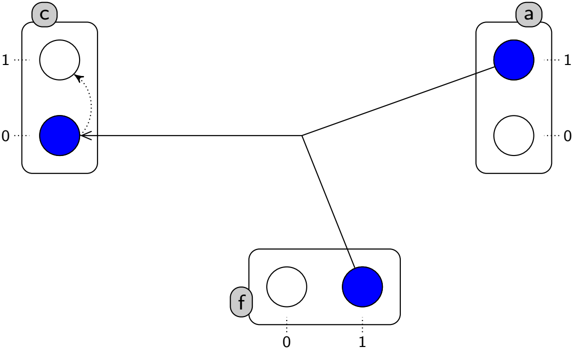
Asynchronous Automata Networks (AAN)
Enriched Process Hitting (PH)



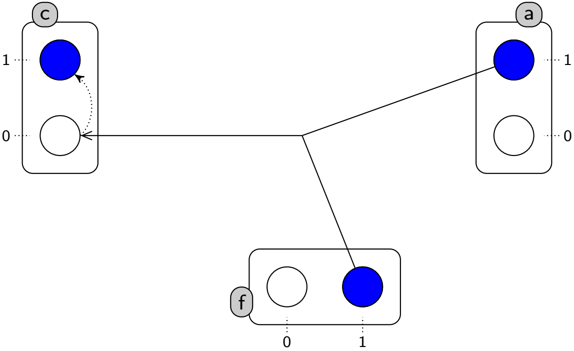
Asynchronous Automata Networks (AAN)
Enriched Process Hitting (PH)



Asynchronous Automata Networks (AAN)
Enriched Process Hitting (PH)

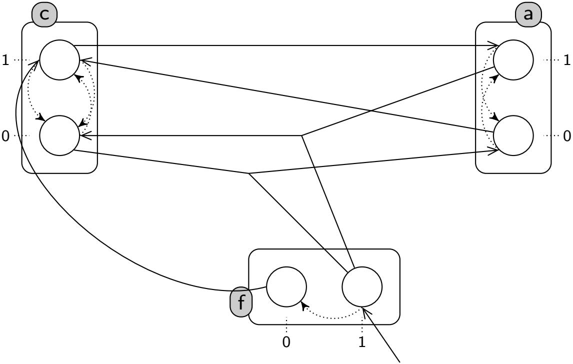


Asynchronous Automata Networks (AAN)
Enriched Process Hitting (PH)



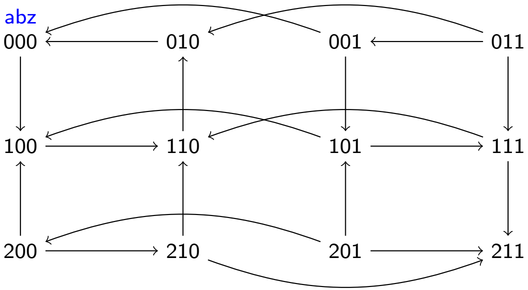
Asynchronous Automata Networks (AAN)
Enriched Process Hitting (PH)

Model from [François *et al.* in Molecular Systems Biology, 2007]



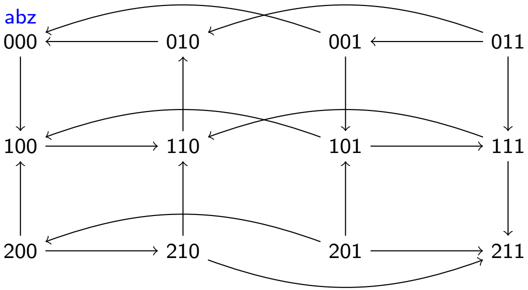
State-graph

The state-graph depicts the whole dynamics
Computation: **exponential** in the size of the model



State-graph

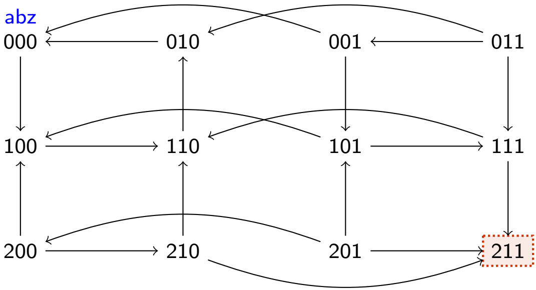
The state-graph depicts the whole dynamics
Computation: **exponential** in the size of the model



Attractor = minimal set of states from which the dynamics cannot escape

State-graph

The state-graph depicts the whole dynamics
Computation: **exponential** in the size of the model

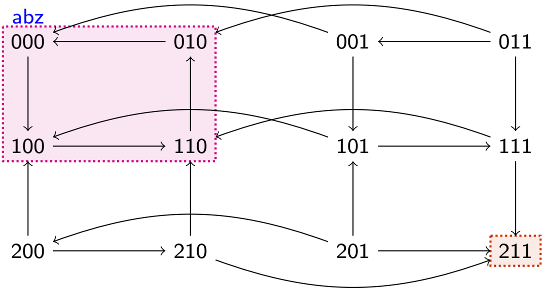


Attractor = minimal set of states from which the dynamics cannot escape

- **Stable state** (state with no successors)

State-graph

The state-graph depicts the whole dynamics
Computation: **exponential** in the size of the model

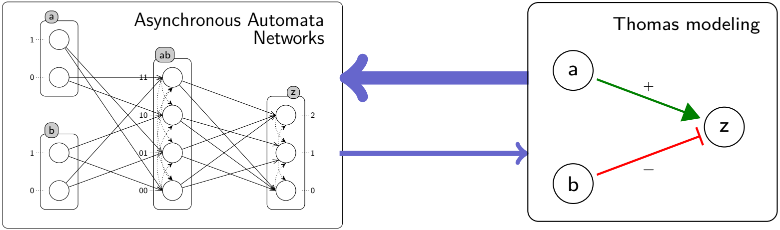


Attractor = minimal set of states from which the dynamics cannot escape

- **Stable state** (state with no successors)
- **Complex attractor** (loop or composition of loops)

Translations Between AAN and Thomas Modeling

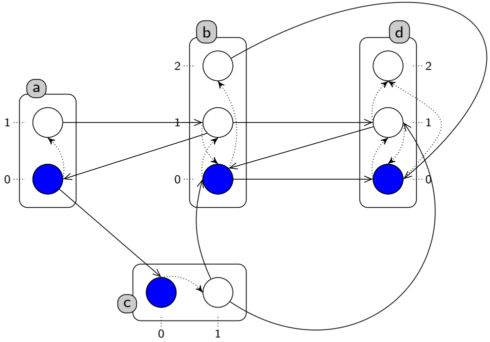
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]
[Folschette *et al.*, *CS2Bio'13*, 2013]



- Asynchronous Automata Networks encompass Thomas modeling
- Mutual translations developed
- Results are also mutually applicable

The Reachability Problem

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]

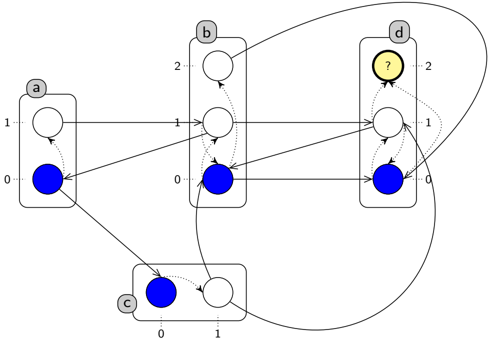


- Initial state

$\langle a_1, b_0, c_0, d_0 \rangle$

The Reachability Problem

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]



- Initial state

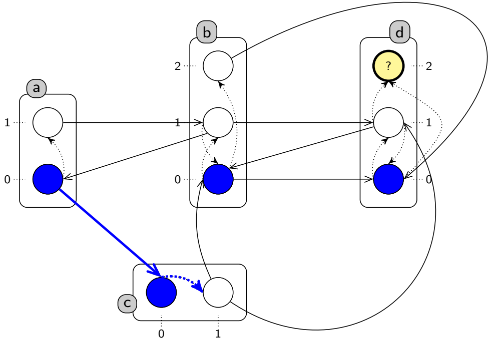
$\langle a_1, b_0, c_0, d_0 \rangle$

- Objective

$[\ d_2\ ]$

The Reachability Problem

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]



- Initial state

$\langle a_1, b_0, c_0, d_0 \rangle$

- Objective

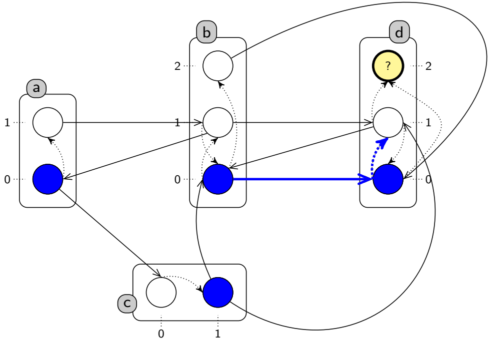
$[\ d_2\ ]$

→ Concretization of the objective = scenario

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

The Reachability Problem

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]



- Initial state

$\langle a_1, b_0, c_0, d_0 \rangle$

- Objective

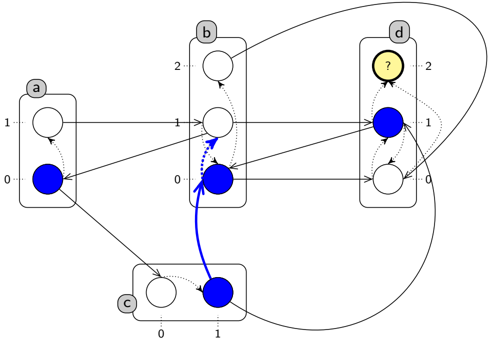
$[\ d_2\ ]$

→ Concretization of the objective = scenario

$$a_0 \to c_0 \upharpoonright c_1 :: \underline{b_0 \to d_0 \upharpoonright d_1} :: c_1 \to b_0 \upharpoonright b_1 :: b_1 \to d_1 \upharpoonright d_2$$

The Reachability Problem

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]



- Initial state

$\langle a_1, b_0, c_0, d_0 \rangle$

- Objective

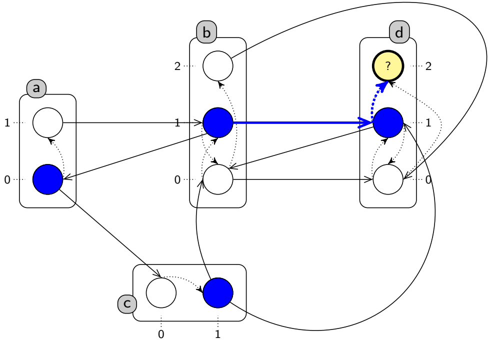
$[\ d_2\ ]$

$\rightarrow$ Concretization of the objective = scenario

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

The Reachability Problem

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]



- Initial state

$\langle a_1, b_0, c_0, d_0 \rangle$

- Objective

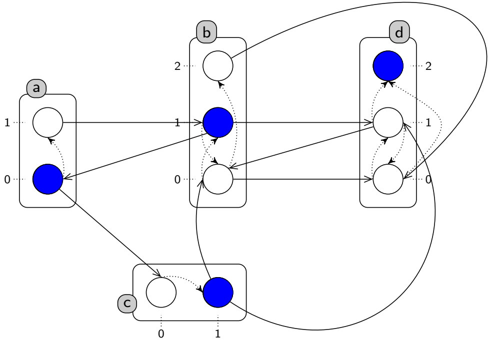
$[\ d_2\ ]$

$\rightarrow$ Concretization of the objective = scenario

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

The Reachability Problem

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]



- Initial state

$\langle a_1, b_0, c_0, d_0 \rangle$

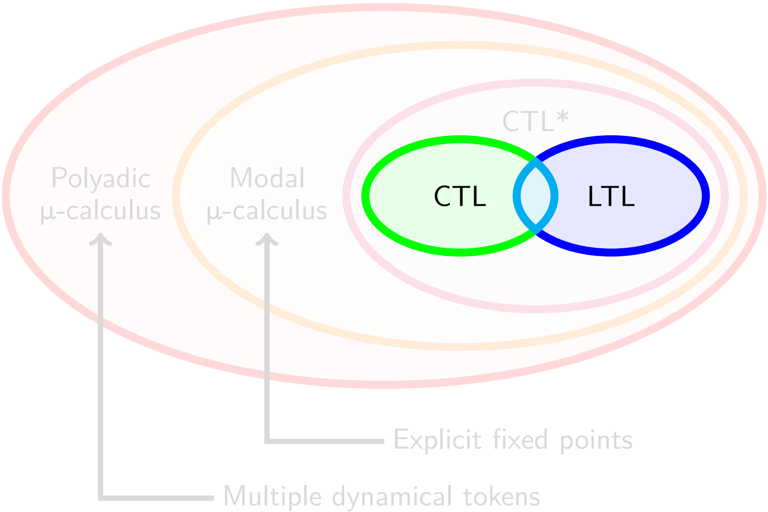
- Objective

$[\ d_2\ ]$

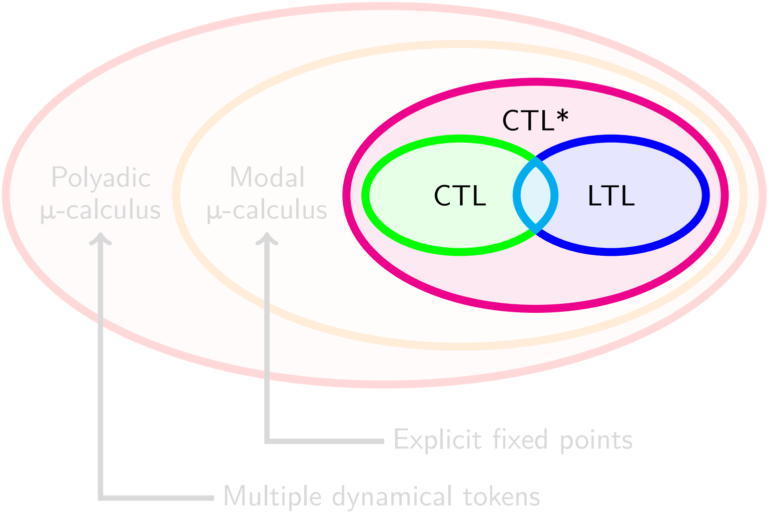
→ Concretization of the objective = scenario

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

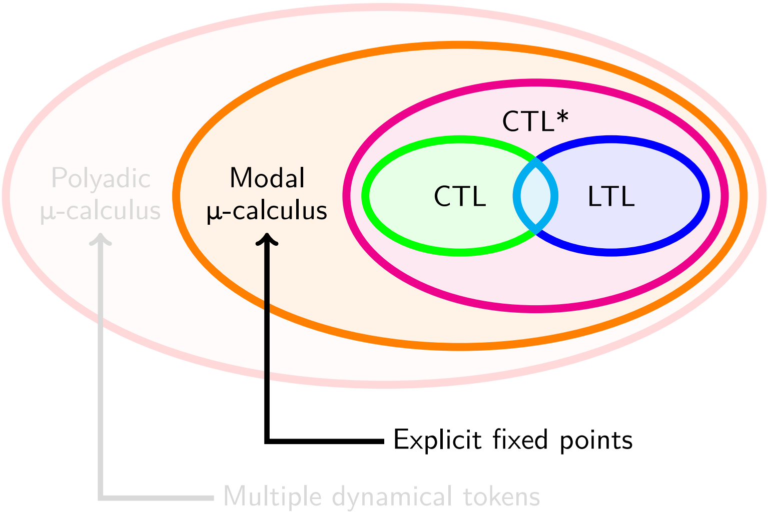
The Polyadic µ-caculus



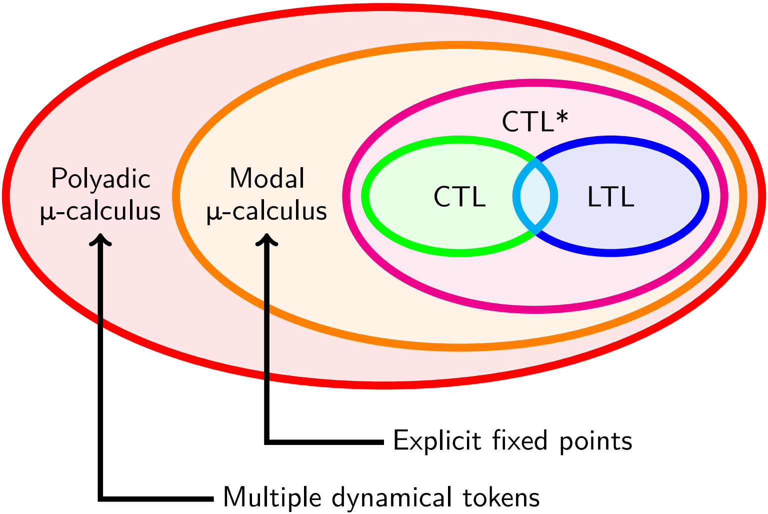
The Polyadic µ-caculus



The Polyadic µ-caculus



The Polyadic μ-caculus



CTL*

CTL

LTL

Polyadic
μ-calculus

Modal
μ-calculus

Explicit fixed points

Multiple dynamical tokens

The Modal μ-calculus

LTL: Implicit fixed point of the "Until" operator
$p\ U\ q \equiv$ "Either q, or p and the next state also verifies $p\ U\ q$"

(Modal) μ-calculus makes such fixed points explicit

$$\varphi = p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid X$$

- Basic property: p ("p is verified in this node")
- Modal operators: $\Box$ ("for all successors"), $\Diamond$ ("there exists a successor")
- Fixed points: μ (least fixed point), ν (greatest fixed point)

Polyadic (modal) μ-calculus allows to manipulate several tokens in parallel

$$\varphi = p_i \mid i \leftarrow j \mid i = j \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond_i\varphi \mid \Box_i\varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid X$$

Token manipulations:
- $i = j$ ("make tokens i and j point to the same node")
- $i \leftarrow j$ ("move token i to the position of token j")

The Modal μ-calculus

LTL: Implicit fixed point of the "Until" operator
$p\ U\ q \equiv$ "Either q, or p and the next state also verifies $p\ U\ q$"

(Modal) μ-calculus makes such fixed points explicit

$$\varphi = p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid X$$

- Basic property: p ("p is verified in this node")
- Modal operators: $\Box$ ("for all successors"), $\Diamond$ ("there exists a successor")
- Fixed points: μ (least fixed point), ν (greatest fixed point)

Polyadic (modal) μ-calculus allows to manipulate several tokens in parallel

$$\varphi = p_i \mid i \leftarrow j \mid i = j \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond_i\varphi \mid \Box_i\varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid X$$

Token manipulations:
- $i = j$ ("make tokens i and j point to the same node")
- $i \leftarrow j$ ("move token i to the position of token j")

The Modal µ-calculus

LTL: Implicit fixed point of the "Until" operator
$p\ U\ q \equiv$ "Either q, or p and the next state also verifies $p\ U\ q$"

(Modal) µ-calculus makes such fixed points explicit

$$\varphi = p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid X$$

- Basic property: p ("p is verified in this node")
- Modal operators: $\Box$ ("for all successors"), $\Diamond$ ("there exists a successor")
- Fixed points: μ (least fixed point), ν (greatest fixed point)

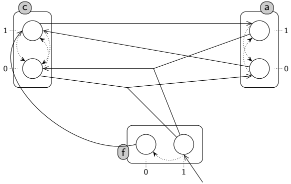
Polyadic (modal) µ-calculus allows to manipulate several tokens in parallel

$$\varphi = p_i \mid i \leftarrow j \mid i = j \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond_i\varphi \mid \Box_i\varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid X$$

Token manipulations:

- $i = j$ ("make tokens i and j point to the same node")
- $i \leftarrow j$ ("move token i to the position of token j")

Applications of the Polyadic μ-calculus



Objective: Unify formulas for many dynamical problems

Not always possible with classical temporal logics (LTL, CTL, CTL*):

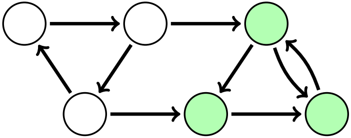
1) From the initial state $(a, b, z) = (0, 0, 0)$, is it possible to reach $z = 2$?
$(a = 0 \wedge b = 0 \wedge z = 0) \Rightarrow \mathrm{EF}(z = 2)$

2) Does $(0, 0, 0)$ belong to an attractor?
$(a = 0 \wedge b = 0 \wedge z = 0) \Rightarrow \mathrm{N}\bot \vee \mathrm{AG}(\mathrm{EF}(a = 0 \wedge b = 0 \wedge z = 0))$

3) What is the set of attractors of the model?
??? — Requires a quantification on the set of all states

Idea: Use polyadic μ-calculus with one token per automata

Search for Attractors with Polyadic µ-calculus

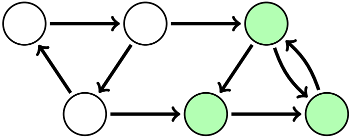


🟢 = belongs to an attractor

$$\varphi_{\text{att}} = \{\boldsymbol{y} \leftarrow \boldsymbol{x}\}\nu W. \underbrace{\underbrace{(\mu Z.\,(\boldsymbol{x} = \boldsymbol{y}) \vee (\Diamond_{\boldsymbol{x}} Z))}_{\varphi_{\text{reach}}} \wedge (\Box_{\boldsymbol{x}} W)}_{\varphi_{\text{explore}}}$$

- $[\![\varphi_{\text{reach}}]\!] = \{(s;t) \mid s \to^* t\}$
  $\varphi_{\text{reach}} \equiv$ "There exists a path from $\boldsymbol{x}$ to $\boldsymbol{y}$"
- $[\![\varphi_{\text{explore}}]\!] = \{(s;t) \mid \forall s', s \to^* s' \Rightarrow s' \to^* t\}$
  $\varphi_{\text{explore}} \equiv$ "All successors of $\boldsymbol{x}$ can reach $\boldsymbol{y}$"
- $[\![\varphi_{\text{att}}]\!] = \{(s;s) \mid \forall s', s \to^* s' \Rightarrow s' \to^* s\}$
  $\varphi_{\text{att}} \equiv$ "$\boldsymbol{x}$ belongs to an attractor"

Search for Attractors with Polyadic µ-calculus

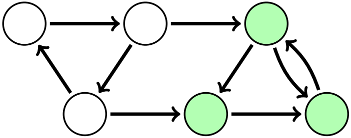


🟢 = belongs to an attractor

$$\varphi_{\text{att}} = \{\boldsymbol{y} \leftarrow \boldsymbol{x}\}\nu W.\underbrace{\underbrace{(\mu Z.\,(\boldsymbol{x} = \boldsymbol{y}) \vee (\Diamond_{\boldsymbol{x}} Z))}_{\varphi_{\text{reach}}} \wedge (\Box_{\boldsymbol{x}} W)}_{\varphi_{\text{explore}}}$$

- $[\![\varphi_{\text{reach}}]\!] = \{(s;t) \mid s \rightarrow^* t\}$
  $\varphi_{\text{reach}} \equiv$ "There exists a path from $\boldsymbol{x}$ to $\boldsymbol{y}$"
- $[\![\varphi_{\text{explore}}]\!] = \{(s;t) \mid \forall s', s \rightarrow^* s' \Rightarrow s' \rightarrow^* t\}$
  $\varphi_{\text{explore}} \equiv$ "All successors of $\boldsymbol{x}$ can reach $\boldsymbol{y}$"
- $[\![\varphi_{\text{att}}]\!] = \{(s;s) \mid \forall s', s \rightarrow^* s' \Rightarrow s' \rightarrow^* s\}$
  $\varphi_{\text{att}} \equiv$ "$\boldsymbol{x}$ belongs to an attractor"

Search for Attractors with Polyadic μ-calculus

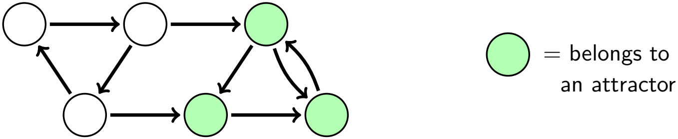


🟢 = belongs to an attractor

$$\varphi_{\text{att}} = \{\boldsymbol{y} \leftarrow \boldsymbol{x}\}\nu W.\underbrace{\underbrace{(\mu Z.\,(\boldsymbol{x} = \boldsymbol{y}) \vee (\Diamond_{\boldsymbol{x}} Z))}_{\varphi_{\text{reach}}} \wedge (\Box_{\boldsymbol{x}} W)}_{\varphi_{\text{explore}}}$$

- $[\![\varphi_{\text{reach}}]\!] = \{(s; t) \mid s \to^{*} t\}$
  $\varphi_{\text{reach}} \equiv$ “There exists a path from $\boldsymbol{x}$ to $\boldsymbol{y}$”
- $[\![\varphi_{\text{explore}}]\!] = \{(s; t) \mid \forall s', s \to^{*} s' \Rightarrow s' \to^{*} t\}$
  $\varphi_{\text{explore}} \equiv$ “All successors of $\boldsymbol{x}$ can reach $\boldsymbol{y}$”
- $[\![\varphi_{\text{att}}]\!] = \{(s; s) \mid \forall s', s \to^{*} s' \Rightarrow s' \to^{*} s\}$
  $\varphi_{\text{att}} \equiv$ “$\boldsymbol{x}$ belongs to an attractor”

Search for Attractors with Polyadic µ-calculus



$$\varphi_{\text{att}} = \{\boldsymbol{y} \leftarrow \boldsymbol{x}\}\nu W.\underbrace{\underbrace{(\mu Z.\,(\boldsymbol{x} = \boldsymbol{y}) \vee (\Diamond_{\boldsymbol{x}} Z))}_{\varphi_{\text{reach}}} \wedge (\Box_{\boldsymbol{x}} W)}_{\varphi_{\text{explore}}}$$

- $[\![\varphi_{\text{reach}}]\!] = \{(s;t) \mid s \to^{*} t\}$
  $\varphi_{\text{reach}} \equiv$ "There exists a path from $\boldsymbol{x}$ to $\boldsymbol{y}$"
- $[\![\varphi_{\text{explore}}]\!] = \{(s;t) \mid \forall s', s \to^{*} s' \Rightarrow s' \to^{*} t\}$
  $\varphi_{\text{explore}} \equiv$ "All successors of $\boldsymbol{x}$ can reach $\boldsymbol{y}$"
- $[\![\varphi_{\text{att}}]\!] = \{(s;s) \mid \forall s', s \to^{*} s' \Rightarrow s' \to^{*} s\}$
  $\varphi_{\text{att}} \equiv$ "$\boldsymbol{x}$ belongs to an attractor"

Conclusion on Polyadic µ-calculus

Properties expressed so far:

- Enumeration of attractors
- Enumeration of switches
- Bisimulation between two models (regarding a set of observables)
- Highlighting Zeno behaviors

Aim: Unification of properties without quantifiers

Complexity: Exponential (equivalent to building the state graph)

Outlooks:

- New formulas
- Implementation
- Generate µ-calculus formulas? (More readable interface)

Answer Set Programming

Answer Set Programming (ASP): Declarative & logic programming

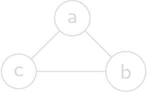
Rule: $head \leftarrow body.$
"If *body* is true, then *head* must be true (usual logical consequence)"

Fact: $head.$
"*head* is always true"

Constraint: $\bot \leftarrow body.$
"If *body* is true, it invalidates the whole answer set"

Example:

$node(a).\ node(b).\ node(c).$
$edge(a,b).\ edge(b,c).\ edge(a,c).$
$edge(X,Y) \leftarrow edge(Y,X).$



Solving: Finding the minimal set of atoms satisfying the problem

```
node(a) node(c) node(b)
edge(a,b) edge(b,c) edge(a,c)
edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming

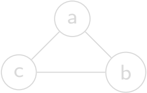
Answer Set Programming (ASP): Declarative & logic programming

Rule: $head \leftarrow A_1, ..., A_n, \neg A_{n+1}, ..., \neg A_m.$
"If *body* is true, then *head* must be true (usual logical consequence)"

Fact: $head.$
"*head* is always true"

Constraint: $\perp \leftarrow body.$
"If *body* is true, it invalidates the whole answer set"

Example:
$node(a).\ node(b).\ node(c).$
$edge(a, b).\ edge(b, c).\ edge(a, c).$
$edge(X, Y) \leftarrow edge(Y, X).$



Solving: Finding the minimal set of atoms satisfying the problem

```
node(a) node(c) node(b)
edge(a,b) edge(b,c) edge(a,c)
edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming

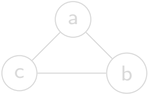
Answer Set Programming (ASP): Declarative & logic programming

Rule: $head \leftarrow A_1, \ldots, A_n, \neg A_{n+1}, \ldots, \neg A_m.$
“If *body* is true, then *head* must be true (usual logical consequence)”

Fact: $head \leftarrow \top.$
“*head* is always true”

Constraint: $\bot \leftarrow body.$
“If *body* is true, it invalidates the whole answer set”

Example:
$node(a).\ node(b).\ node(c).$
$edge(a, b).\ edge(b, c).\ edge(a, c).$
$edge(X, Y) \leftarrow edge(Y, X).$



Solving: Finding the minimal set of atoms satisfying the problem

```
node(a) node(c) node(b)
edge(a,b) edge(b,c) edge(a,c)
edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming

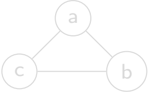
Answer Set Programming (ASP): Declarative & logic programming

Rule: $head \leftarrow A_1, ..., A_n, \neg A_{n+1}, ..., \neg A_m.$
“If *body* is true, then *head* must be true (usual logical consequence)”

Fact: $head.$
“*head* is always true”

Constraint: $\bot \leftarrow body.$
“If *body* is true, it invalidates the whole answer set”

Example:
$node(a).\ node(b).\ node(c).$
$edge(a, b).\ edge(b, c).\ edge(a, c).$
$edge(X, Y) \leftarrow edge(Y, X).$



Solving: Finding the minimal set of atoms satisfying the problem

```
node(a) node(c) node(b)
edge(a,b) edge(b,c) edge(a,c)
edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming

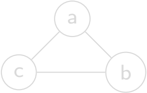
Answer Set Programming (ASP): Declarative & logic programming

Rule: $head \leftarrow A_1, ..., A_n, \neg A_{n+1}, ..., \neg A_m.$
"If *body* is true, then *head* must be true (usual logical consequence)"

Fact: $head.$
"*head* is always true"

Constraint: $\bot \leftarrow body.$
"If *body* is true, it invalidates the whole answer set"

Example:
$node(a). \; node(b). \; node(c).$
$edge(a, b). \; edge(b, c). \; edge(a, c).$
$edge(X, Y) \leftarrow edge(Y, X).$



Solving: Finding the minimal set of atoms satisfying the problem

```
node(a) node(c) node(b)
edge(a,b) edge(b,c) edge(a,c)
edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming

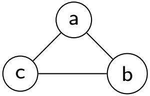
Answer Set Programming (ASP): Declarative & logic programming

Rule: $head \leftarrow A_1, \ldots, A_n, \neg A_{n+1}, \ldots, \neg A_m.$
"If *body* is true, then *head* must be true (usual logical consequence)"

Fact: $head.$
"*head* is always true"

Constraint: $\bot \leftarrow body.$
"If *body* is true, it invalidates the whole answer set"

Example:
$node(a).\ node(b).\ node(c).$
$edge(a, b).\ edge(b, c).\ edge(a, c).$
$edge(X, Y) \leftarrow edge(Y, X).$



Solving: Finding the minimal set of atoms satisfying the problem

```
node(a) node(c) node(b)
edge(a,b) edge(b,c) edge(a,c)
edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming

Answer Set Programming (ASP): Declarative & logic programming

Rule: $head \leftarrow A_1, ..., A_n, \neg A_{n+1}, ..., \neg A_m.$

"If *body* is true, then *head* must be true (usual logical consequence)"

Fact: $head.$

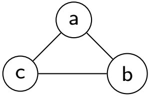
"*head* is always true"

Constraint: $\bot \leftarrow body.$

"If *body* is true, it invalidates the whole answer set"

Example:

$node(a).\ node(b).\ node(c).$
$edge(a, b).\ edge(b, c).\ edge(a, c).$
$edge(X, Y) \leftarrow edge(Y, X).$



Solving: Finding the minimal set of atoms satisfying the problem

```
node(a) node(c) node(b)
edge(a,b) edge(b,c) edge(a,c)
edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming

Cardinalities: $min \ \{ \ atom : enum \ \} \ max \leftarrow body.$

- Enumerates all atoms of the form *atom* according to the variables of *enum*
- Keep between *min* and *max* possibilities
- Creates as many answer sets as there are combinations

General method:

1) **Enumerate** of all candidate combinations using cardinalities

$color(red). \ color(green). \ color(blue).$

$1 \ \{ \ attrib(X, C) : color(C) \ \} \ 1 \leftarrow node(X).$

```
Answer set 1: attrib(b,red) attrib(c,red) attrib(a,red)
Answer set 2: attrib(b,red) attrib(c,red) attrib(a,blue)
Answer set 3: attrib(b,red) attrib(c,green) attrib(a,blue)
   ⋮ (27 answer sets)
```

2) **Filter out** the undesired candidates using constraints

$\bot \leftarrow attrib(X, C), attrib(Y, C), edge(X, Y).$

```
Answer set 1: attrib(b,green) attrib(c,blue) attrib(a,red)
Answer set 2: attrib(b,green) attrib(c,red) attrib(a,blue)
Answer set 3: attrib(b,blue) attrib(c,green) attrib(a,red)
   ⋮ (6 answer sets)
```

Answer Set Programming

Cardinalities: $min \; \{ \; atom : enum \; \} \; max \leftarrow body.$

- Enumerates all atoms of the form *atom* according to the variables of *enum*
- Keep between *min* and *max* possibilities
- Creates as many answer sets as there are combinations

General method:

1) **Enumerate** of all candidate combinations using cardinalities

$color(red). \; color(green). \; color(blue).$

$1 \; \{ \; attrib(X, C) : color(C) \; \} \; 1 \leftarrow node(X).$

```
Answer set 1: attrib(b,red) attrib(c,red) attrib(a,red)
Answer set 2: attrib(b,red) attrib(c,red) attrib(a,blue)
Answer set 3: attrib(b,red) attrib(c,green) attrib(a,blue)
   ⋮ (27 answer sets)
```

2) **Filter out** the undesired candidates using constraints

$\bot \leftarrow attrib(X, C), attrib(Y, C), edge(X, Y).$

```
Answer set 1: attrib(b,green) attrib(c,blue) attrib(a,red)
Answer set 2: attrib(b,green) attrib(c,red) attrib(a,blue)
Answer set 3: attrib(b,blue) attrib(c,green) attrib(a,red)
   ⋮ (6 answer sets)
```

Answer Set Programming

Cardinalities: $min \ \{ \ atom : enum \ \} \ max \leftarrow body.$

- Enumerates all atoms of the form *atom* according to the variables of *enum*
- Keep between *min* and *max* possibilities
- Creates as many answer sets as there are combinations

General method:

1) **Enumerate** of all candidate combinations using cardinalities

$color(red). \ color(green). \ color(blue).$

$1 \ \{ \ attrib(X, C) : color(C) \ \} \ 1 \leftarrow node(X).$

```
Answer set 1: attrib(b,red) attrib(c,red) attrib(a,red)
Answer set 2: attrib(b,red) attrib(c,red) attrib(a,blue)
Answer set 3: attrib(b,red) attrib(c,green) attrib(a,blue)
   ⋮ (27 answer sets)
```

2) **Filter out** the undesired candidates using constraints

$\bot \leftarrow attrib(X, C), attrib(Y, C), edge(X, Y).$

```
Answer set 1: attrib(b,green) attrib(c,blue) attrib(a,red)
Answer set 2: attrib(b,green) attrib(c,red) attrib(a,blue)
Answer set 3: attrib(b,blue) attrib(c,green) attrib(a,red)
   ⋮ (6 answer sets)
```

ASP for Model-checking

[Ben Abdallah, Folschette, Roux, Magnin, *BIBM'15*, 2015]

Usage: Describe the problem instead of its resolution

Stable states enumeration

1) Describe the model with facts (automata, actions)
2) Describe what a playable action is with rules
3) Enumerate all states with cardinalities
4) Filter out states with a playable action

Reachability analysis (reaching a given state)

1) Describe the model with facts (automata, actions, initial & target states)
2) Create the dynamics:
    - describe playability with rules
    - enumerate potential futures with cardinalities and constraints
3) Filter out paths that don't end in the target state

Conclusion on ASP for Model-checking

[Ben Abdallah, Folschette, Roux, Magnin, *BIBM'15*, 2015]

Complexity: Exponential (exhaustive computation of the dynamics)
But strong heuristics that give good results

Models		Stable states	Reachability analysis		
Name	States	ASP	libddd[1]	GINsim[2]	ASP
egfr20	2^{64}	**0.017s**	1min 55s	2min 32s	**12s**
tcrsig40	2^{73}	**0.021s**	∞	∞	**4min 28s**

[1] LIP6/Move [Couvreur *et al.*, *Lecture Notes in Computer Science*, 2002]
[2] TAGC/IGC [Chaouiya, Naldi, Thieffry, *Methods in Molecular Biology*, 2012]

egfr20 : Epithelial Growth Factor Receptor (20 components) [Sahin *et al.*, 2009]
egfr104 : Epithelial Growth Factor Receptor (104 components) [Samaga *et al.*, 2009]

Pros: Very flexible (programming language),
The complexity taken care of by the solver

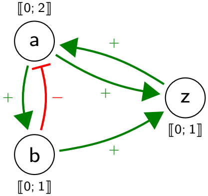
Outlooks:

- New properties to check (reverse reachability, Eden gardens)
- Optimizations (exclude cycles)

Static Analysis of Thomas Modeling

[Thomas, *Numerical Methods in the Study of Critical Phenomena*, 1981]

Conjectures of René Thomas:

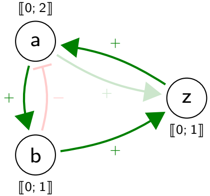


Static Analysis of Thomas Modeling

[Thomas, *Numerical Methods in the Study of Critical Phenomena*, 1981]

Conjectures of René Thomas:

- Multiple **stable states** ⇒ positive cycle in the graph



110

111

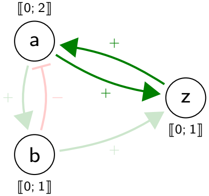
000

Static Analysis of Thomas Modeling

[Thomas, *Numerical Methods in the Study of Critical Phenomena*, 1981]

Conjectures of René Thomas:

- Multiple **stable states** ⇒ positive cycle in the graph



110

111

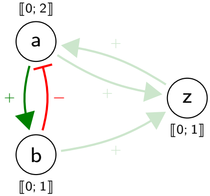
000

Static Analysis of Thomas Modeling

[Thomas, *Numerical Methods in the Study of Critical Phenomena*, 1981]

Conjectures of René Thomas:

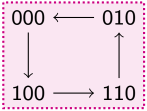
- Multiple **stable states** ⇒ positive cycle in the graph
- Sustained oscillations (**complex attractor**) ⇒ negative cycle in the graph



110

111

000

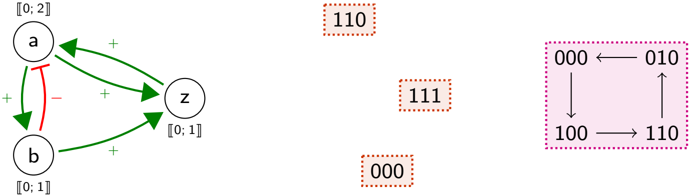


Static Analysis of Thomas Modeling

[Thomas, *Numerical Methods in the Study of Critical Phenomena*, 1981]

Conjectures of René Thomas:

- Multiple **stable states** ⇒ positive cycle in the graph
- Sustained oscillations (**complex attractor**) ⇒ negative cycle in the graph



Proofs: [Remy, Ruet & Thieffry, *Advances in Applied Mathematics*, 2008]
[Richard, *Advances in Applied Mathematics*, 2010]
[Richard & Comet, *Discrete Applied Mathematics*, 2007]

Static Analysis of Thomas Modeling

[Thomas, *Numerical Methods in the Study of Critical Phenomena*, 1981]
[Paulevé & Richard, *Electronic Notes in Theoretical Computer Science*, 2012]

Contrapositives:

- No positive cycle in the graph ⇒ The **stable state** (if any) is unique
- No negative cycle in the graph ⇒ No **complex attractor** (only stable states)

Other results:

- Lower & upper bounds of the number of attractors
- Functionality of the cycles
- Sufficient condition for no stable state
- Topological stable states

Complexity: Usually very low (searching for all cycles)

Limitations: Very broad results on the dynamics
(cannot predict the evolution of one particular component)

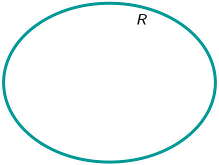
→ **Need for more precise methods**

Over- and Under-approximations

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]

→ Directly checking R is hard (**exponential**)

→ Rather check **approximations** P and Q so that: $P \Rightarrow R \Rightarrow Q$
Computing P or Q is much simpler (roughly **polynomial**)



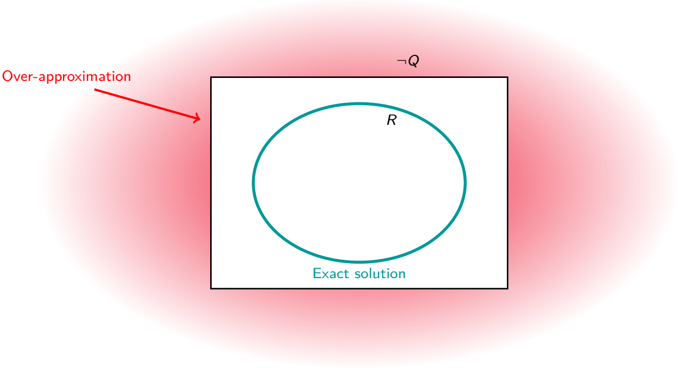
Exact solution

Over- and Under-approximations

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]

→ Directly checking R is hard (**exponential**)

→ Rather check **approximations** P and Q so that: $P \Rightarrow R \Rightarrow Q$
Computing P or Q is much simpler (roughly **polynomial**)

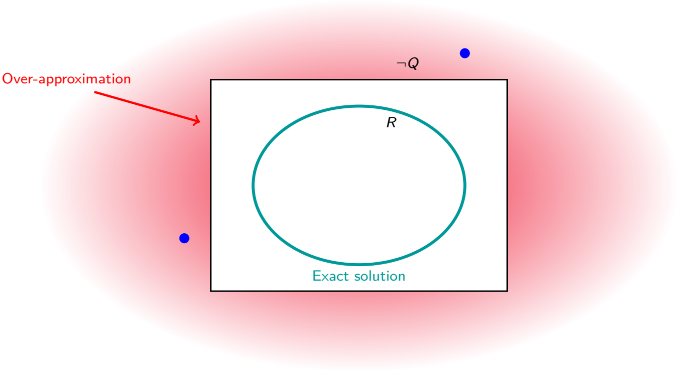


Over- and Under-approximations

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]

→ Directly checking R is hard (**exponential**)

→ Rather check **approximations** P and Q so that: $P \Rightarrow R \Rightarrow Q$
Computing P or Q is much simpler (roughly **polynomial**)

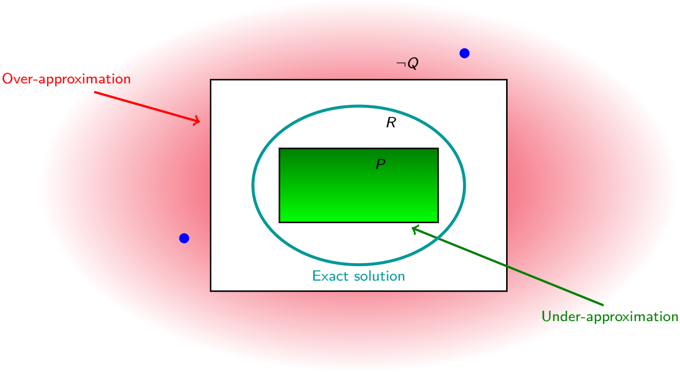


Over- and Under-approximations

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]

→ Directly checking R is hard (**exponential**)

→ Rather check **approximations** P and Q so that: $P \Rightarrow R \Rightarrow Q$
Computing P or Q is much simpler (roughly **polynomial**)

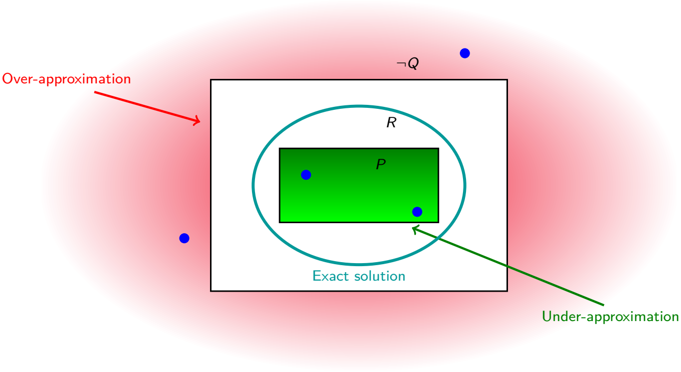


Over- and Under-approximations

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]

→ Directly checking R is hard (**exponential**)

→ Rather check **approximations** P and Q so that: $P \Rightarrow R \Rightarrow Q$
Computing P or Q is much simpler (roughly **polynomial**)

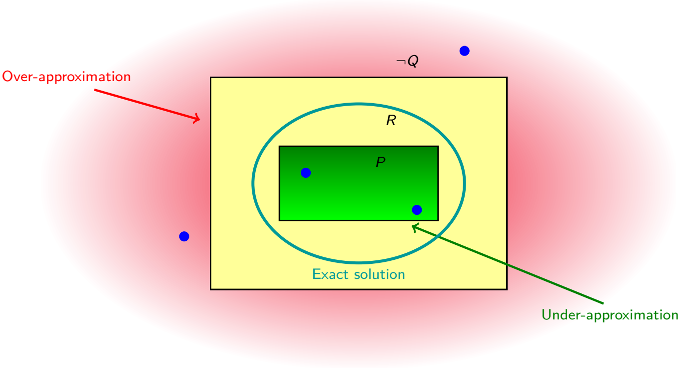


Over- and Under-approximations

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]

→ Directly checking R is hard (**exponential**)

→ Rather check **approximations** P and Q so that: $P \Rightarrow R \Rightarrow Q$
Computing P or Q is much simpler (roughly **polynomial**)

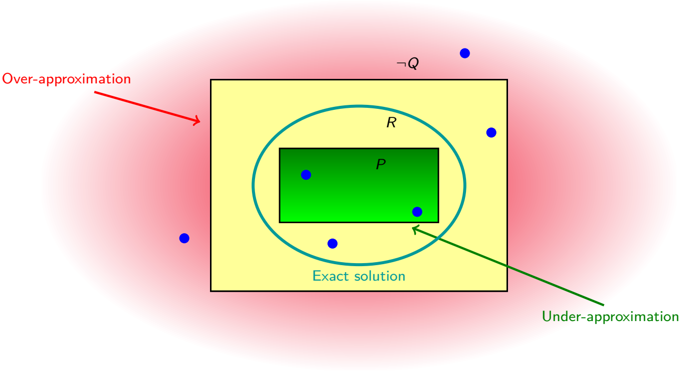


Over- and Under-approximations

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]

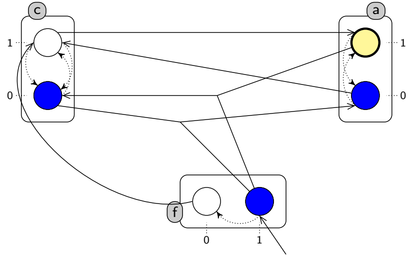
→ Directly checking R is hard (**exponential**)

→ Rather check **approximations** P and Q so that: $P \Rightarrow R \Rightarrow Q$
Computing P or Q is much simpler (roughly **polynomial**)

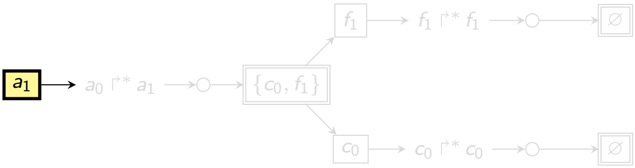


Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- All leaves are $\boxed{\boxed{\varnothing}}$



$\boxed{a_1}$

$a_0 \Rsh^* a_1$

$\{c_0, f_1\}$

f_1

$f_1 \Rsh^* f_1$

$\varnothing$

c_0

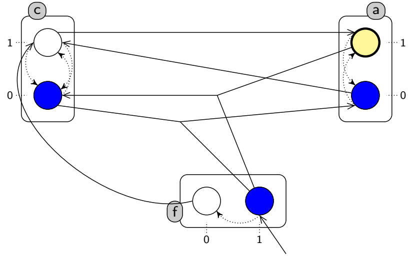
$c_0 \Rsh^* c_0$

$\varnothing$

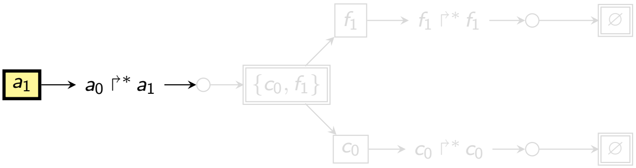
$$\{c_0, f_1\} \rightarrow a_0 \Rsh a_1$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- All leaves are $\boxed{\boxed{\varnothing}}$



$\boxed{a_1} \longrightarrow a_0 \Rsh^* a_1 \longrightarrow \circ \longrightarrow \boxed{\boxed{\{c_0, f_1\}}}$

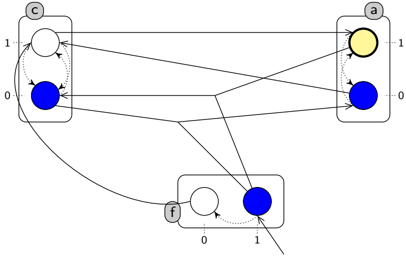
$\boxed{f_1} \longrightarrow f_1 \Rsh^* f_1 \longrightarrow \circ \longrightarrow \boxed{\boxed{\varnothing}}$

$\boxed{c_0} \longrightarrow c_0 \Rsh^* c_0 \longrightarrow \circ \longrightarrow \boxed{\boxed{\varnothing}}$

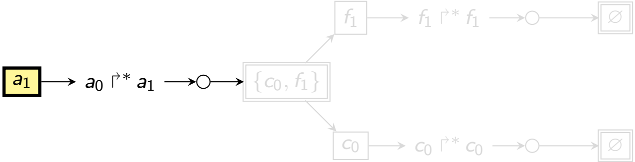
$$\{c_0, f_1\} \rightarrow a_0 \Rsh a_1$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- All leaves are $\boxed{\emptyset}$



$\boxed{a_1} \longrightarrow a_0 \Rsh^* a_1 \longrightarrow \bigcirc \longrightarrow \boxed{\{c_0, f_1\}}$

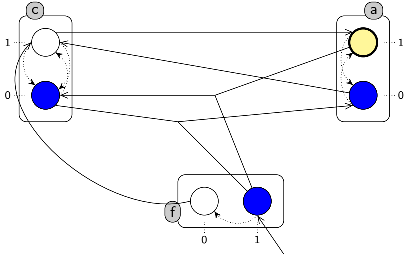
$\boxed{f_1} \longrightarrow f_1 \Rsh^* f_1 \longrightarrow \bigcirc \longrightarrow \boxed{\emptyset}$

$\boxed{c_0} \longrightarrow c_0 \Rsh^* c_0 \longrightarrow \bigcirc \longrightarrow \boxed{\emptyset}$

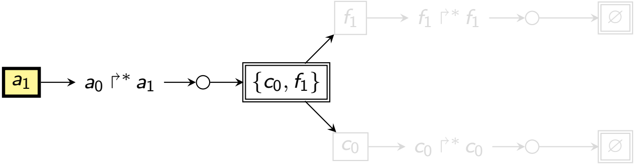
$$\{c_0, f_1\} \to a_0 \Rsh a_1$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- All leaves are $\boxed{\varnothing}$



$a_1 \longrightarrow a_0 \upharpoonright^* a_1 \longrightarrow \circ \longrightarrow \boxed{\{c_0, f_1\}}$

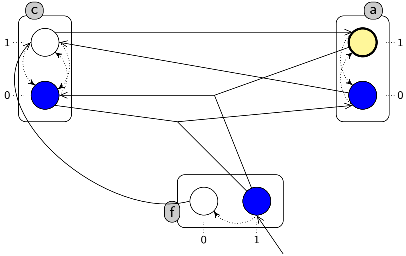
$f_1 \longrightarrow f_1 \upharpoonright^* f_1 \longrightarrow \circ \longrightarrow \boxed{\varnothing}$

$c_0 \longrightarrow c_0 \upharpoonright^* c_0 \longrightarrow \circ \longrightarrow \boxed{\varnothing}$

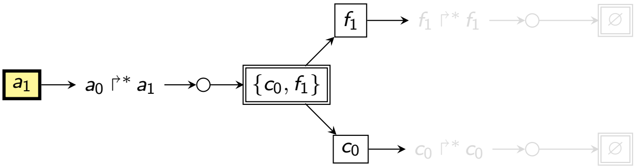
$$\{c_0, f_1\} \rightarrow a_0 \upharpoonright a_1$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- All leaves are $\boxed{\boxed{\varnothing}}$



$\boxed{a_1} \longrightarrow a_0 \Rsh^* a_1 \longrightarrow \circ \longrightarrow \boxed{\boxed{\{c_0, f_1\}}}$

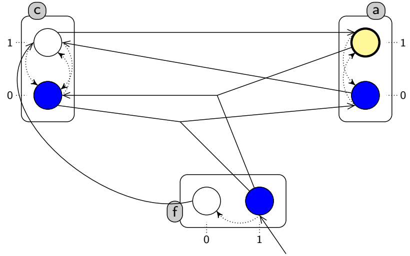
$\boxed{f_1} \longrightarrow f_1 \Rsh^* f_1 \longrightarrow \circ \longrightarrow \boxed{\boxed{\varnothing}}$

$\boxed{c_0} \longrightarrow c_0 \Rsh^* c_0 \longrightarrow \circ \longrightarrow \boxed{\boxed{\varnothing}}$

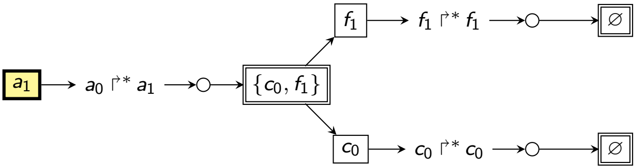
$$\{c_0, f_1\} \rightarrow a_0 \Rsh a_1$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- All leaves are $\boxed{\varnothing}$



$\boxed{a_1}$ → $a_0 \Rsh^* a_1$ → ○ → $\boxed{\{c_0, f_1\}}$

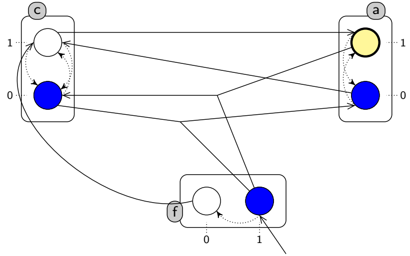
$\boxed{\{c_0, f_1\}}$ → $\boxed{f_1}$ → $f_1 \Rsh^* f_1$ → ○ → $\boxed{\varnothing}$

$\boxed{\{c_0, f_1\}}$ → $\boxed{c_0}$ → $c_0 \Rsh^* c_0$ → ○ → $\boxed{\varnothing}$

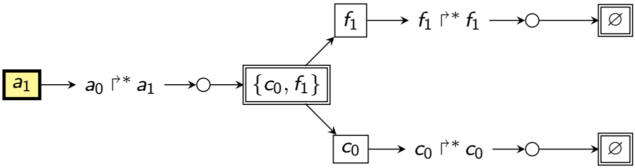
$$\{c_0, f_1\} \rightarrow a_0 \Rsh a_1$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- All leaves are $\boxed{\varnothing}$



$a_1 \rightarrow a_0 \Rsh^* a_1 \rightarrow \circ \rightarrow \{c_0, f_1\}$

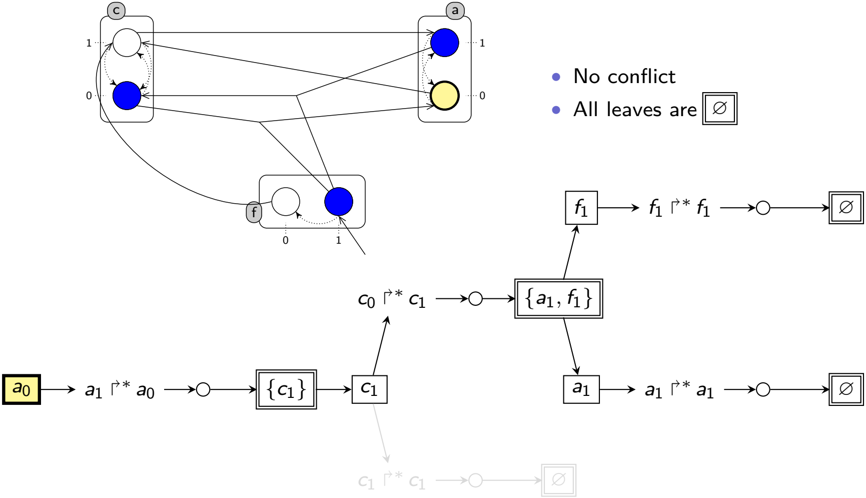
$\{c_0, f_1\} \rightarrow f_1 \rightarrow f_1 \Rsh^* f_1 \rightarrow \circ \rightarrow \varnothing$

$\{c_0, f_1\} \rightarrow c_0 \rightarrow c_0 \Rsh^* c_0 \rightarrow \circ \rightarrow \varnothing$

$$\{c_0, f_1\} \rightarrow a_0 \Rsh a_1$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- All leaves are $\boxed{\boxed{\varnothing}}$

$\boxed{a_0} \rightarrow a_1 \Rsh^* a_0 \rightarrow \circ \rightarrow \boxed{\boxed{\{c_1\}}} \rightarrow \boxed{c_1}$

$\boxed{c_1} \rightarrow c_0 \Rsh^* c_1 \rightarrow \circ \rightarrow \boxed{\boxed{\{a_1, f_1\}}}$

$\boxed{\boxed{\{a_1, f_1\}}} \rightarrow \boxed{f_1} \rightarrow f_1 \Rsh^* f_1 \rightarrow \circ \rightarrow \boxed{\boxed{\varnothing}}$

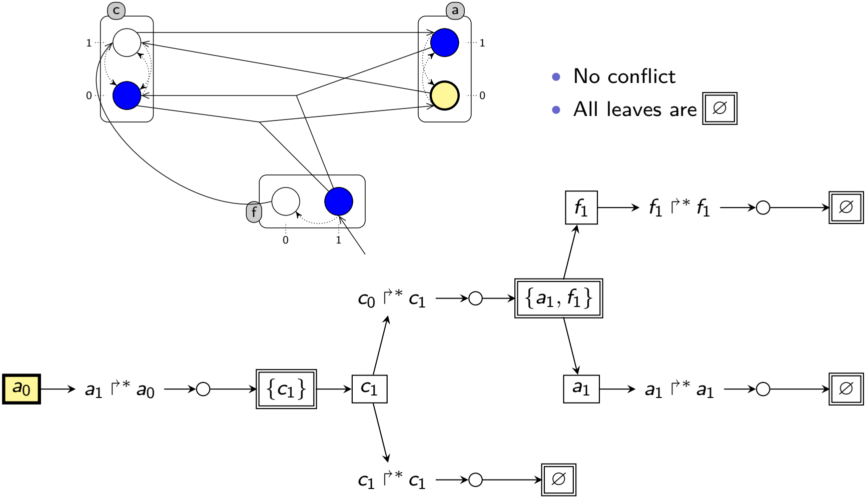
$\boxed{\boxed{\{a_1, f_1\}}} \rightarrow \boxed{a_1} \rightarrow a_1 \Rsh^* a_1 \rightarrow \circ \rightarrow \boxed{\boxed{\varnothing}}$

$\boxed{c_1} \rightarrow c_1 \Rsh^* c_1 \rightarrow \circ \rightarrow \boxed{\boxed{\varnothing}}$

$$\{a_1, f_1\} \rightarrow c_0 \Rsh c_1 \;::\; \{c_1\} \rightarrow a_1 \Rsh a_0$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- All leaves are $\boxed{\varnothing}$

$\boxed{a_0} \rightarrow a_1 \Rsh^* a_0 \rightarrow \circ \rightarrow \boxed{\{c_1\}} \rightarrow \boxed{c_1}$

$\boxed{c_1} \rightarrow c_0 \Rsh^* c_1 \rightarrow \circ \rightarrow \boxed{\{a_1, f_1\}}$

$\boxed{\{a_1, f_1\}} \rightarrow \boxed{f_1} \rightarrow f_1 \Rsh^* f_1 \rightarrow \circ \rightarrow \boxed{\varnothing}$

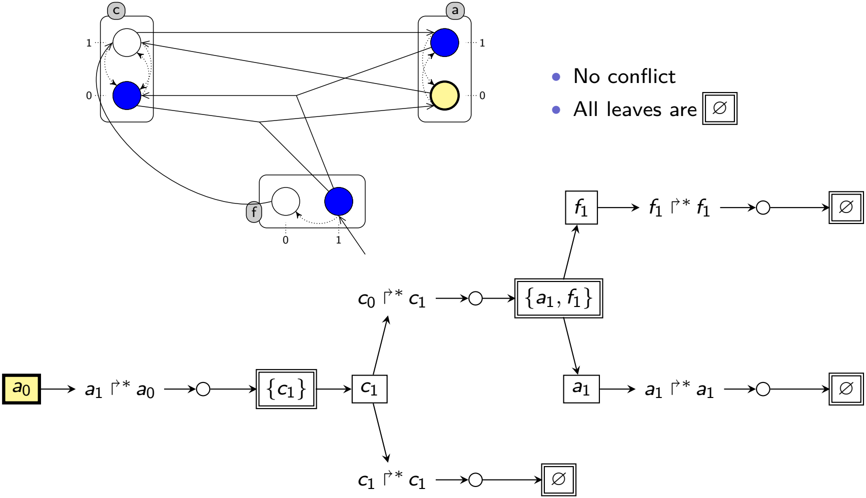
$\boxed{\{a_1, f_1\}} \rightarrow \boxed{a_1} \rightarrow a_1 \Rsh^* a_1 \rightarrow \circ \rightarrow \boxed{\varnothing}$

$\boxed{c_1} \rightarrow c_1 \Rsh^* c_1 \rightarrow \circ \rightarrow \boxed{\varnothing}$

$$\{a_1, f_1\} \rightarrow c_0 \Rsh c_1 \quad :: \quad \{c_1\} \rightarrow a_1 \Rsh a_0$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- All leaves are $\boxed{\varnothing}$

$\boxed{a_0} \rightarrow a_1 \upharpoonright^* a_0 \rightarrow \circ \rightarrow \boxed{\{c_1\}} \rightarrow \boxed{c_1}$

$\boxed{c_1} \rightarrow c_0 \upharpoonright^* c_1 \rightarrow \circ \rightarrow \boxed{\{a_1, f_1\}}$

$\boxed{c_1} \rightarrow c_1 \upharpoonright^* c_1 \rightarrow \circ \rightarrow \boxed{\varnothing}$

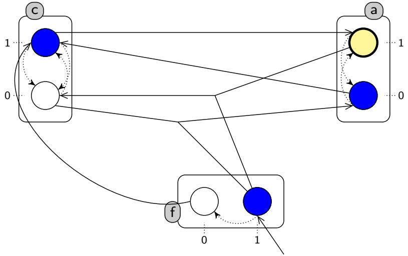
$\boxed{\{a_1, f_1\}} \rightarrow \boxed{f_1} \rightarrow f_1 \upharpoonright^* f_1 \rightarrow \circ \rightarrow \boxed{\varnothing}$

$\boxed{\{a_1, f_1\}} \rightarrow \boxed{a_1} \rightarrow a_1 \upharpoonright^* a_1 \rightarrow \circ \rightarrow \boxed{\varnothing}$

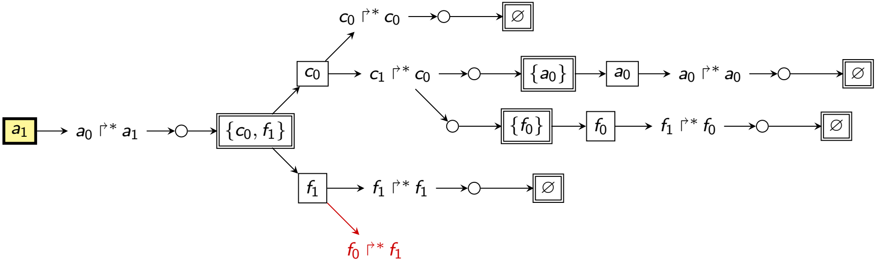
$$\{a_1, f_1\} \rightarrow c_0 \upharpoonright c_1 \;::\; \{c_1\} \rightarrow a_1 \upharpoonright a_0$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- ~~All leaves are $\boxed{\emptyset}$~~



a_1

$a_0 \Rsh^* a_1$

$\{c_0, f_1\}$

c_0

$c_0 \Rsh^* c_0$

$\emptyset$

$c_1 \Rsh^* c_0$

$\{a_0\}$

a_0

$a_0 \Rsh^* a_0$

$\emptyset$

$\{f_0\}$

f_0

$f_1 \Rsh^* f_0$

$\emptyset$

f_1

$f_1 \Rsh^* f_1$

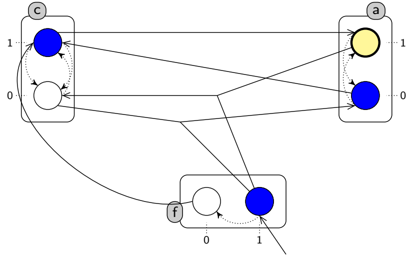
$\emptyset$

$f_0 \Rsh^* f_1$

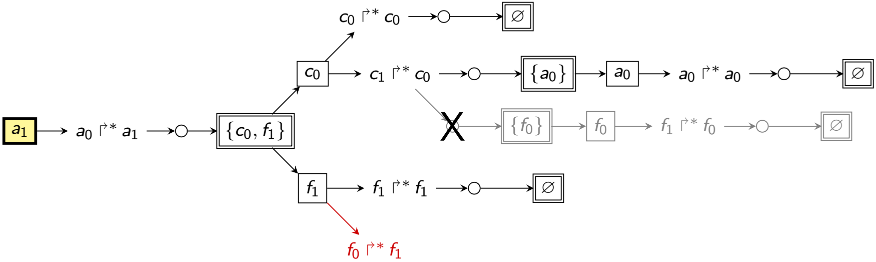
$$\{a_0\} \rightarrow c_1 \Rsh c_0 \;::\; \{c_0, f_1\} \rightarrow a_0 \Rsh a_1$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- ~~All leaves are $\boxed{\varnothing}$~~



$\boxed{a_1} \rightarrow a_0 \Rsh^* a_1 \rightarrow \circ \rightarrow \boxed{\{c_0, f_1\}}$

$\boxed{c_0} \rightarrow c_0 \Rsh^* c_0 \rightarrow \circ \rightarrow \boxed{\varnothing}$

$\boxed{c_0} \rightarrow c_1 \Rsh^* c_0 \rightarrow \circ \rightarrow \boxed{\{a_0\}} \rightarrow \boxed{a_0} \rightarrow a_0 \Rsh^* a_0 \rightarrow \circ \rightarrow \boxed{\varnothing}$

$c_1 \Rsh^* c_0 \rightarrow$ X $\rightarrow \boxed{\{f_0\}} \rightarrow \boxed{f_0} \rightarrow f_1 \Rsh^* f_0 \rightarrow \circ \rightarrow \boxed{\varnothing}$

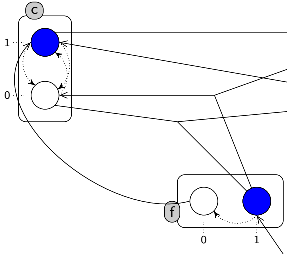
$\boxed{f_1} \rightarrow f_1 \Rsh^* f_1 \rightarrow \circ \rightarrow \boxed{\varnothing}$

$\boxed{f_1} \rightarrow f_0 \Rsh^* f_1$

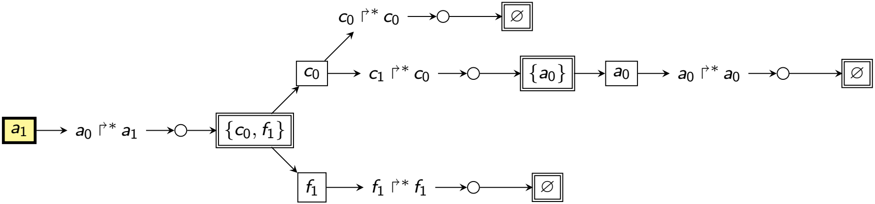
$$\{a_0\} \rightarrow c_1 \Rsh c_0 \;::\; \{c_0, f_1\} \rightarrow a_0 \Rsh a_1$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



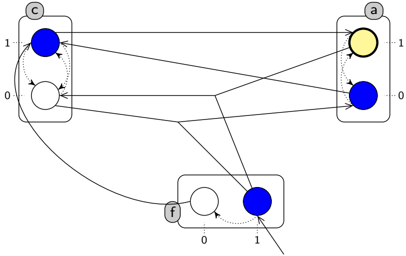
- No conflict
- All leaves are $\boxed{\varnothing}$



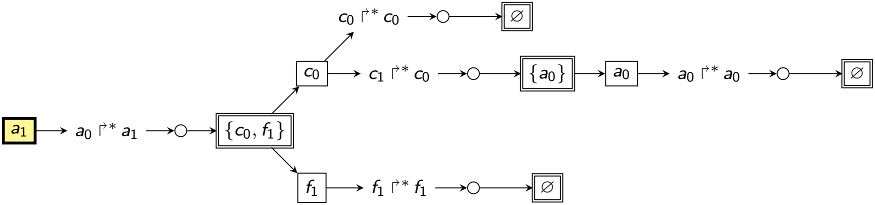
$$\{a_0\} \rightarrow c_1 \Rsh c_0 \;::\; \{c_0, f_1\} \rightarrow a_0 \Rsh a_1$$

Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



- No conflict
- All leaves are $\boxed{\boxed{\varnothing}}$



$$\{a_0\} \rightarrow c_1 \upharpoonright c_0 \;::\; \{c_0, f_1\} \rightarrow a_0 \upharpoonright a_1$$

Implementation of the Abstract Interpretation

Complexity:

- Computation of the local causality graph:
    - Polynomial in the number of automata
    - Exponential in the number of local states of each automata (usually very low, max. 4)
- Analysis of the graph (sufficient condition):
    - Polynomial in the size of the abstract graph
- Enumeration of the subsets of solutions (if needed):
    - Exponential in the size of the abstract graph

→ Very efficient on biological networks: **many components with few local states**

Model	Automata	Actions	States	libddd[1]	GINsim[2]	PINT[3]
egfr20	35	670	2^{64}		$<$1s	**0.02s**
tcrsig40	54	301	2^{73}		∞	**0.02s**
tcrsig94	133	1124	2^{194}	[$>$1min – ∞]		**0.03s**
egfr104	193	2356	2^{320}	[$>$1min – ∞]		**0.16s**

[1] LIP6/Move [Couvreur *et al.*, *Lecture Notes in Computer Science*, 2002]

[2] TAGC/IGC [Chaouiya, Naldi, Thieffry, *Methods in Molecular Biology*, 2012]

[3] Loïc Paulevé [`http://loicpauleve.name/pint/`]

egfr20 : Epithelial Growth Factor Receptor (20 components) [Sahin *et al.*, 2009]
egfr104 : Epithelial Growth Factor Receptor (104 components) [Samaga *et al.*, 2009]
tcrsig40 : T-Cell Receptor (40 components) [Klamt *et al.*, 2006]
tcrsig94 : T-Cell Receptor (94 components) [Saez-Rodriguez *et al.*, 2007]

Implementation of the Abstract Interpretation

Complexity:

- Computation of the local causality graph:
  - Polynomial in the number of automata
  - Exponential in the number of local states of each automata (usually very low, max. 4)
- Analysis of the graph (sufficient condition):
  - Polynomial in the size of the abstract graph
- Enumeration of the subsets of solutions (if needed):
  - Exponential in the size of the abstract graph

→ Very efficient on biological networks: **many components with few local states**

Model	Automata	Actions	States	libddd[1]	GINsim[2]	PINT[3]
egfr20	35	670	2^{64}		<1s	**0.02s**
tcrsig40	54	301	2^{73}		∞	**0.02s**
tcrsig94	133	1124	2^{194}	[>1min – ∞]		**0.03s**
egfr104	193	2356	2^{320}	[>1min – ∞]		**0.16s**

[1] LIP6/Move [Couvreur *et al.*, *Lecture Notes in Computer Science*, 2002]

[2] TAGC/IGC [Chaouiya, Naldi, Thieffry, *Methods in Molecular Biology*, 2012]

[3] Loïc Paulevé [`http://loicpauleve.name/pint/`]

egfr20 : Epithelial Growth Factor Receptor (20 components) [Sahin *et al.*, 2009]
egfr104 : Epithelial Growth Factor Receptor (104 components) [Samaga *et al.*, 2009]
tcrsig40 : T-Cell Receptor (40 components) [Klamt *et al.*, 2006]
tcrsig94 : T-Cell Receptor (94 components) [Saez-Rodriguez *et al.*, 2007]

Summary & Conclusion

- Discrete modeling = coherent abstraction of real biochemical phenomena
  - → Discrete Networks / Thomas modeling
  - → Asynchronous Automata Networks
  - → ...And other extensions related to Asynchronous Automata Networks
- Polyadic μ-calculus
  - → More generic than CTL*
  - → Examples: enumeration of attractors, switches, bisimulation...
  - → Ongoing implementation
- Answer Set Programming
  - → Logic programming works!
  - → Powerful heuristics giving good performances
- Static analysis by abstract interpretation
  - → (Only) reachability properties (CTL operator EF)
  - → Very efficient (polynomial complexity)
  - → Broad range of models (translations of Thomas models)

Summary & Conclusion

- Discrete modeling = coherent abstraction of real biochemical phenomena
  - → Discrete Networks / Thomas modeling
  - → Asynchronous Automata Networks
  - → ...And other extensions related to Asynchronous Automata Networks
- Polyadic μ-calculus
  - → More generic than CTL*
  - → Examples: enumeration of attractors, switches, bisimulation...
  - → Ongoing implementation
- Answer Set Programming
  - → Logic programming works!
  - → Powerful heuristics giving good performances
- Static analysis by abstract interpretation
  - → (Only) reachability properties (CTL operator EF)
  - → Very efficient (polynomial complexity)
  - → Broad range of models (translations of Thomas models)

Summary & Conclusion

- Discrete modeling = coherent abstraction of real biochemical phenomena
  - → Discrete Networks / Thomas modeling
  - → Asynchronous Automata Networks
  - → …And other extensions related to Asynchronous Automata Networks
- Polyadic μ-calculus
  - → More generic than CTL*
  - → Examples: enumeration of attractors, switches, bisimulation…
  - → Ongoing implementation
- Answer Set Programming
  - → Logic programming works!
  - → Powerful heuristics giving good performances
- Static analysis by abstract interpretation
  - → (Only) reachability properties (CTL operator EF)
  - → Very efficient (polynomial complexity)
  - → Broad range of models (translations of Thomas models)

Summary & Conclusion

- Discrete modeling = coherent abstraction of real biochemical phenomena
  - → Discrete Networks / Thomas modeling
  - → Asynchronous Automata Networks
  - → ...And other extensions related to Asynchronous Automata Networks

- Polyadic μ-calculus
  - → More generic than CTL*
  - → Examples: enumeration of attractors, switches, bisimulation...
  - → Ongoing implementation

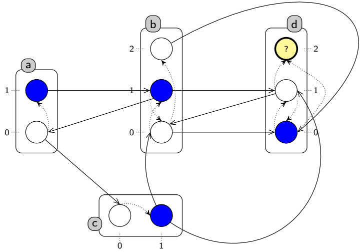
- Answer Set Programming
  - → Logic programming works!
  - → Powerful heuristics giving good performances

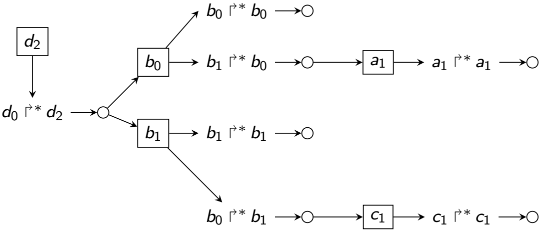
- Static analysis by abstract interpretation
  - → (Only) reachability properties (CTL operator EF)
  - → Very efficient (polynomial complexity)
  - → Broad range of models (translations of Thomas models)

Bibliography

- René Thomas. On the Relation Between the Logical Structure of Systems and Their Ability to Generate Multiple Steady States or Sustained Oscillations. In Jean Della Dora, Jacques Demongeot and Bernard Lacolle, editors: *Numerical Methods in the Study of Critical Phenomena*, Synergies 9, 180–193. Springer Berlin Heidelberg, 1981.
- 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*, Lecture Notes in Computer Science, 171–191. Springer Berlin Heidelberg, 2011.
- Loïc Paulevé, Morgan Magnin, Olivier Roux. Static analysis of biological regulatory networks dynamics using abstract interpretation. *Mathematical Structures in Computer Science*. 2012.
- Loïc Paulevé, Adrien Richard. Static Analysis of Boolean Networks Based on Interaction Graphs: A Survey, *Electronic Notes in Theoretical Computer Science* 284, 93–104. Elsevier, 2012.
- Adrien Richard and Jean-Paul Comet. Necessary conditions for multistationarity in discrete dynamical systems. *Discrete Applied Mathematics* 155(18), 2403–2413. 2007.
- Adrien Richard. Negative circuits and sustained oscillations in asynchronous automata networks, *Advances in Applied Mathematics* 44(4), 378–392. Elsevier, 2010.
- Élisabeth Remy, Paul Ruet and Denis Thieffry. Graphic requirements for multistability and attractive cycles in a boolean dynamical framework, *Advances in Applied Mathematics* 41(3), 335–350. Elsevier, 2008.
- Maxime Folschette, Loïc Paulevé, Kastumi Inoue, Morgan Magnin and Olivier Roux. Identification of Biological Regulatory Networks from Process Hitting models, *Theoretical Computer Science* 568, 49–71. Elsevier, 2015a.
- Maxime Folschette, Loïc Paulevé, Morgan Magnin and Olivier Roux. Sufficient Conditions for Reachability in Automata Networks with Priorities, *Theoretical Computer Science*. Elsevier, 2015b.
- Maxime Folschette, Loïc Paulevé, Morgan Magnin, Olivier Roux. Under-approximation of Reachability in Multivalued Asynchronous Networks. In E. Merelli and A. Troina, editors, *4th International Workshop on Interactions between Computer Science and Biology (CS2Bio'13)*, Electronic Notes in Theoretical Computer Science, Volume 299, 33–51. June 2013.
- Emna Ben Abdallah, Maxime Folschette, Olivier Roux, Morgan Magnin. Exhaustive analysis of dynamical properties of Biological Regulatory Networks with Answer Set Programming. *IEEE International Conference on Bioinformatics and Biomedicine (BIBM'15)*, 281–285, IEEE. November 2015.

Under-approximation





$\boxed{d_2}$

$d_0 \Rsh^* d_2$

$\boxed{b_0}$

$\boxed{b_1}$

$b_0 \Rsh^* b_0$

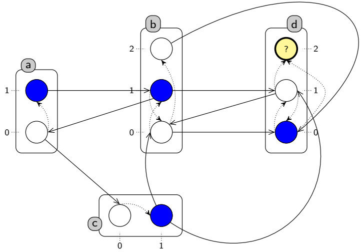
$b_1 \Rsh^* b_0$ → $\boxed{a_1}$ → $a_1 \Rsh^* a_1$

$b_1 \Rsh^* b_1$

$b_0 \Rsh^* b_1$ → $\boxed{c_1}$ → $c_1 \Rsh^* c_1$

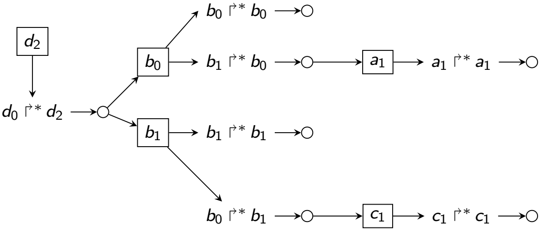
$\boxed{d_2}$	Required process
$d_0 \Rsh^* d_2$	Objective
○	Solution to an objective

Under-approximation



Sufficient condition:

- no cycle
- each objective has a solution



$\boxed{d_2}$

$d_0 \Rsh^* d_2$

$\boxed{b_0}$

$\boxed{b_1}$

$b_0 \Rsh^* b_0$

$b_1 \Rsh^* b_0$

$b_1 \Rsh^* b_1$

$b_0 \Rsh^* b_1$

$\boxed{a_1}$

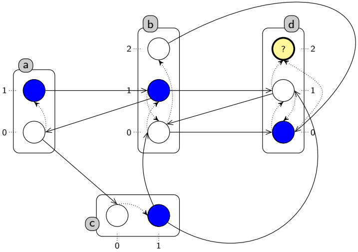
$a_1 \Rsh^* a_1$

$\boxed{c_1}$

$c_1 \Rsh^* c_1$

$\boxed{d_2}$	Required process
$d_0 \Rsh^* d_2$	Objective
○	Solution to an objective

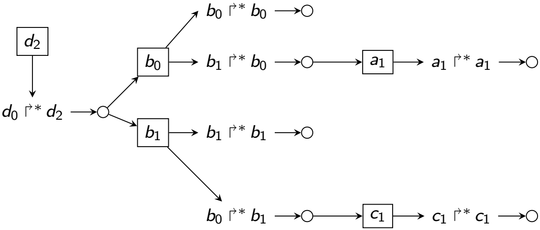
Under-approximation



Sufficient condition:

- no cycle
- each objective has a solution

P is **true** $\Rightarrow$ R is **true**



$\boxed{d_2}$

$d_0 \Rsh^{*} d_2 \longrightarrow \bigcirc$

$\boxed{b_0}$ $\longrightarrow$ $b_0 \Rsh^{*} b_0 \longrightarrow \bigcirc$

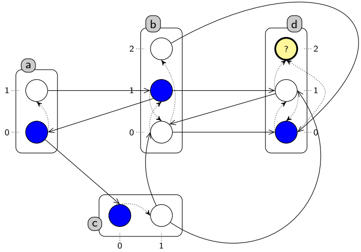
$\boxed{b_0}$ $\longrightarrow$ $b_1 \Rsh^{*} b_0 \longrightarrow \bigcirc \longrightarrow \boxed{a_1} \longrightarrow a_1 \Rsh^{*} a_1 \longrightarrow \bigcirc$

$\boxed{b_1}$ $\longrightarrow$ $b_1 \Rsh^{*} b_1 \longrightarrow \bigcirc$

$\boxed{b_1}$ $\longrightarrow$ $b_0 \Rsh^{*} b_1 \longrightarrow \bigcirc \longrightarrow \boxed{c_1} \longrightarrow c_1 \Rsh^{*} c_1 \longrightarrow \bigcirc$

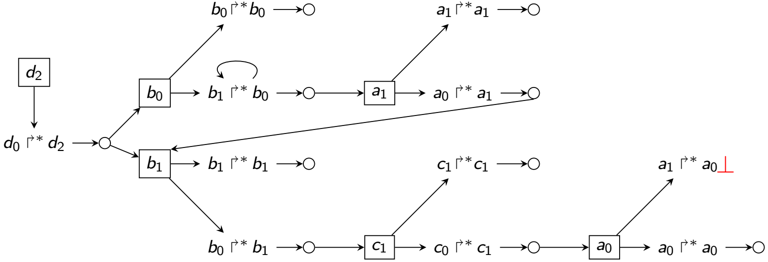
$\boxed{d_2}$	Required process
$d_0 \Rsh^{*} d_2$	Objective
○	Solution to an objective

Under-approximation



Sufficient condition:

- no cycle
- ~~each objective has a solution~~



$b_0 \Rsh^* b_0$

$a_1 \Rsh^* a_1$

d_2

b_0

$b_1 \Rsh^* b_0$

a_1

$a_0 \Rsh^* a_1$

$d_0 \Rsh^* d_2$

b_1

$b_1 \Rsh^* b_1$

$c_1 \Rsh^* c_1$

$a_1 \Rsh^* a_0 \bot$

$b_0 \Rsh^* b_1$

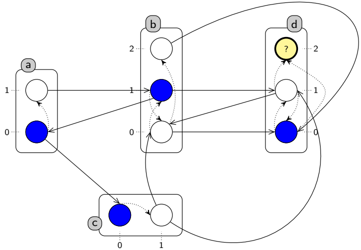
c_1

$c_0 \Rsh^* c_1$

a_0

$a_0 \Rsh^* a_0$

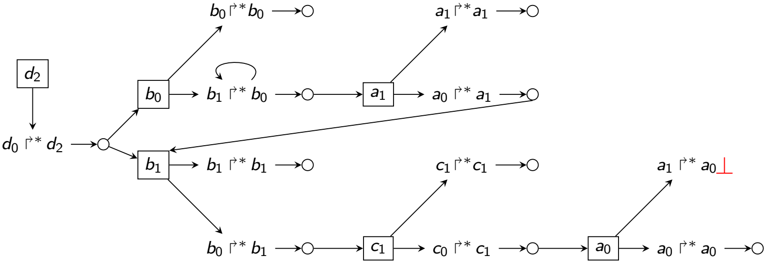
Under-approximation



Sufficient condition:

- no cycle
- ~~each objective has a solution~~

P is **false** ⇒ **Inconclusive**



d_2

$d_0 \Rsh^* d_2$

b_0

b_1

$b_0 \Rsh^* b_0$

$b_1 \Rsh^* b_0$

$b_1 \Rsh^* b_1$

$b_0 \Rsh^* b_1$

a_1

$a_1 \Rsh^* a_1$

$a_0 \Rsh^* a_1$

c_1

$c_1 \Rsh^* c_1$

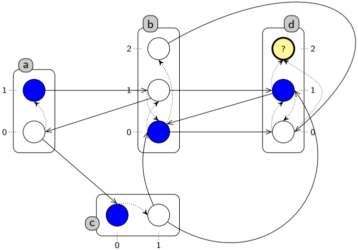
$c_0 \Rsh^* c_1$

a_0

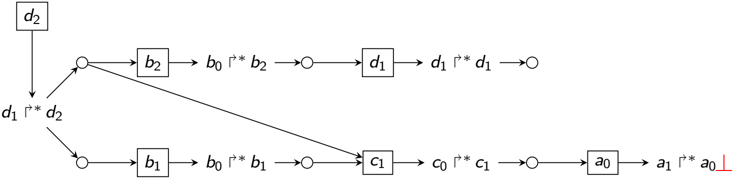
$a_1 \Rsh^* a_0 \bot$

$a_0 \Rsh^* a_0$

Over-approximation



Necessary condition:



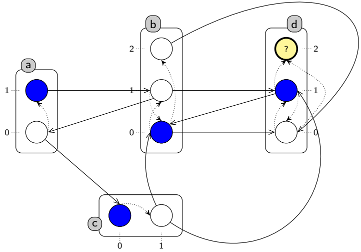
$\boxed{d_2}$

$d_1 \Rsh^{*} d_2$

$\boxed{b_2}$ $b_0 \Rsh^{*} b_2$ $\boxed{d_1}$ $d_1 \Rsh^{*} d_1$

$\boxed{b_1}$ $b_0 \Rsh^{*} b_1$ $\boxed{c_1}$ $c_0 \Rsh^{*} c_1$ $\boxed{a_0}$ $a_1 \Rsh^{*} a_0 \bot$

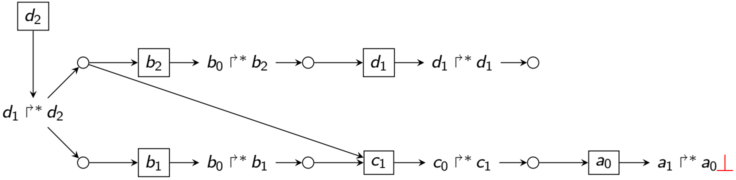
Over-approximation



Necessary condition:

There exists a traversal with no cycle

- objective → follow **one** solution
- solution → follow **all** processes
- process → follow **all** objectives



d_2

$d_1 \Rsh^* d_2$

b_2

$b_0 \Rsh^* b_2$

d_1

$d_1 \Rsh^* d_1$

b_1

$b_0 \Rsh^* b_1$

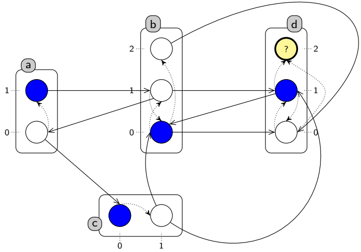
c_1

$c_0 \Rsh^* c_1$

a_0

$a_1 \Rsh^* a_0 \bot$

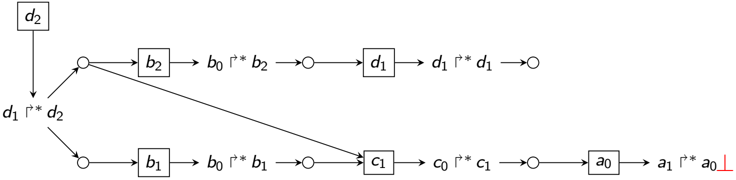
Over-approximation



Necessary condition:

~~There exists a traversal~~ with no cycle

- ~~objective → follow **one** solution~~
- solution → follow **all** processes
- process → follow **all** objectives

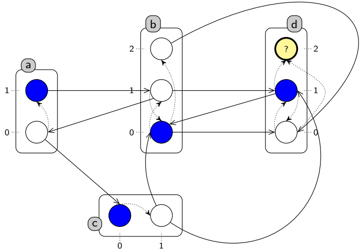

d_2

$d_1 \Rsh^* d_2$

b_2 $b_0 \Rsh^* b_2$ d_1 $d_1 \Rsh^* d_1$

b_1 $b_0 \Rsh^* b_1$ c_1 $c_0 \Rsh^* c_1$ a_0 $a_1 \Rsh^* a_0 \bot$

Over-approximation

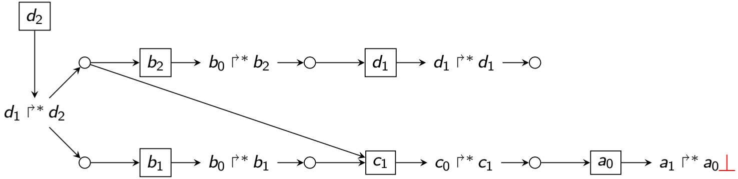


Necessary condition:

~~There exists a traversal~~ with no cycle

- ~~objective → follow **one** solution~~
- solution → follow **all** processes
- process → follow **all** objectives

Q is **false** ⇒ R is **false**



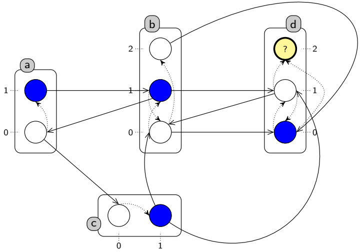
d_2

$d_1 \Rsh^* d_2$

b_2 $b_0 \Rsh^* b_2$ d_1 $d_1 \Rsh^* d_1$

b_1 $b_0 \Rsh^* b_1$ c_1 $c_0 \Rsh^* c_1$ a_0 $a_1 \Rsh^* a_0 \bot$

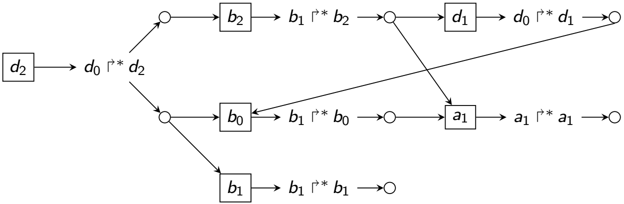
Over-approximation



Necessary condition:

There exists a traversal with no cycle

- objective → follow **one** solution
- solution → follow **all** processes
- process → follow **all** objectives



d_2 → $d_0 \Rsh^* d_2$

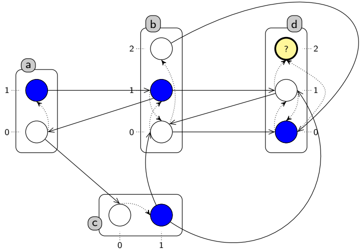
b_2 → $b_1 \Rsh^* b_2$

d_1 → $d_0 \Rsh^* d_1$

b_0 → $b_1 \Rsh^* b_0$

a_1 → $a_1 \Rsh^* a_1$

b_1 → $b_1 \Rsh^* b_1$



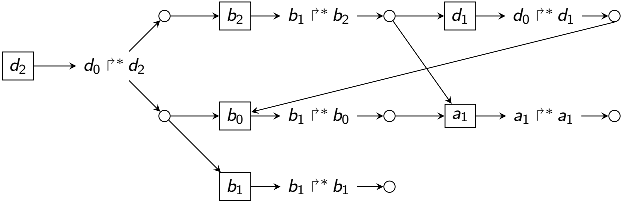
Over-approximation

Necessary condition:

There exists a traversal with no cycle

- objective → follow **one** solution
- solution → follow **all** processes
- process → follow **all** objectives

R **is true** ⇒ **Inconclusive**



$\boxed{d_2}$

$d_0 \Rsh^* d_2$

$\boxed{b_2}$

$b_1 \Rsh^* b_2$

$\boxed{d_1}$

$d_0 \Rsh^* d_1$

$\boxed{b_0}$

$b_1 \Rsh^* b_0$

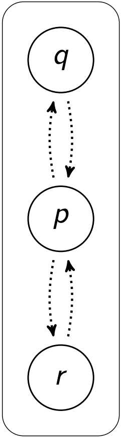
$\boxed{a_1}$

$a_1 \Rsh^* a_1$

$\boxed{b_1}$

$b_1 \Rsh^* b_1$

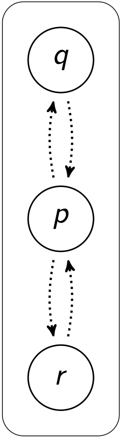
Examples with Modal µ-calculus



No tokens: only one evolution is studied

Atomic property (p**,** q**,** r**)**
$[\![p]\!] = \{p\}$

Examples with Modal µ-calculus



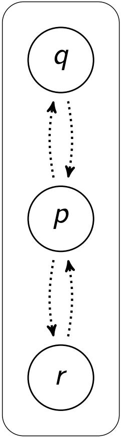
No tokens: only one evolution is studied

Atomic property (*p*, *q*, *r*)

$[\![p]\!] = \{p\}$

$[\![q \vee r]\!] = \{q; r\}$

Examples with Modal μ-calculus



No tokens: only one evolution is studied

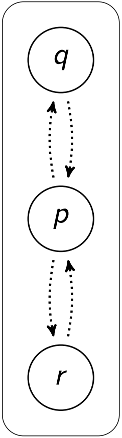
Atomic property (p, q, r)

$[\![p]\!] = \{p\}$
$[\![q \vee r]\!] = \{q; r\}$

Possible future (“may”)

$[\![\Diamond\ q]\!] = \{p\}$

Examples with Modal μ-calculus



No tokens: only one evolution is studied

Atomic property (*p*, *q*, *r*)

$[\![p]\!] = \{p\}$

$[\![q \vee r]\!] = \{q; r\}$

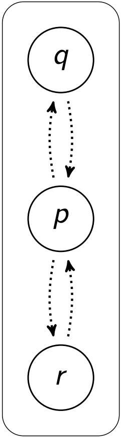
Possible future ("may")

$[\![\lozenge\ q]\!] = \{p\}$

Necessary future ("must")

$[\![\square\ q]\!] = \varnothing$

Examples with Modal μ-calculus



No tokens: only one evolution is studied

Atomic property (*p*, *q*, *r*)

$[\![p]\!] = \{p\}$

$[\![q \vee r]\!] = \{q; r\}$

Possible future ("may")

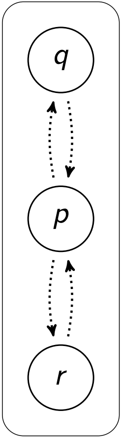
$[\![\Diamond\ q]\!] = \{p\}$

Necessary future ("must")

$[\![\Box\ q]\!] = \varnothing$

$[\![\Box\ p]\!] = \{q; r\}$

Examples with Polyadic µ-calculus

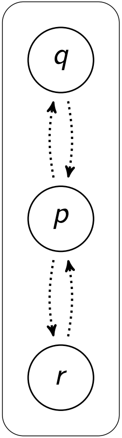


Atomic property (p, q, r)

$[\![p_1 \wedge r_2]\!] = \{(p, r)\}$

$[\![p_1]\!] = \{(p, p); (p, q); (p, r)\}$

Examples with Polyadic μ-calculus



Atomic property (p, q, r)

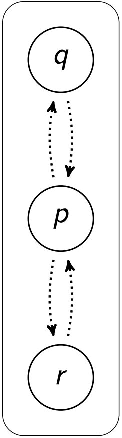
$[\![p_1 \wedge r_2]\!] = \{(p, r)\}$

$[\![p_1]\!] = \{(p, p); (p, q); (p, r)\}$

Token affectation ($i \leftarrow j$)

$[\![\{2 \leftarrow 1\}\ p_1 \wedge p_2]\!] = \{(p, p); (p, q); (p, r)\}$

Examples with Polyadic µ-calculus



Atomic property (p, q, r)

$[\![p_1 \wedge r_2]\!] = \{(p, r)\}$

$[\![p_1]\!] = \{(p, p); (p, q); (p, r)\}$

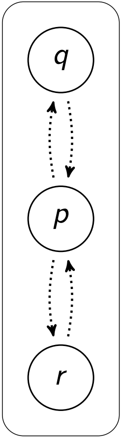
Token affectation ($i \leftarrow j$)

$[\![\{2 \leftarrow 1\}\ p_1 \wedge p_2]\!] = \{(p, p); (p, q); (p, r)\}$

Token comparison ($i = j$)

$[\![1 = 2]\!] = \{(p, p); (q, q); (r, r)\}$

Examples with Polyadic μ-calculus



Atomic property (p, q, r)

$[\![p_1 \wedge r_2]\!] = \{(p, r)\}$

$[\![p_1]\!] = \{(p, p); (p, q); (p, r)\}$

Token affectation ($i \leftarrow j$)

$[\![\{2 \leftarrow 1\}\ p_1 \wedge p_2]\!] = \{(p, p); (p, q); (p, r)\}$

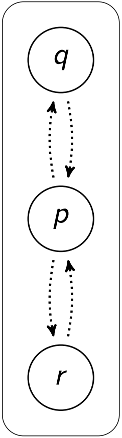
Token comparison ($i = j$)

$[\![1 = 2]\!] = \{(p, p); (q, q); (r, r)\}$

Possible future (“may”)

$[\![\Diamond_1\ q]\!] = \{(p, p); (p, q); (p, r)\}$

Examples with Polyadic μ-calculus



Atomic property (p, q, r)

$[\![p_1 \wedge r_2]\!] = \{(p, r)\}$

$[\![p_1]\!] = \{(p, p); (p, q); (p, r)\}$

Token affectation ($i \leftarrow j$)

$[\![\{2 \leftarrow 1\}\ p_1 \wedge p_2]\!] = \{(p, p); (p, q); (p, r)\}$

Token comparison ($i = j$)

$[\![1 = 2]\!] = \{(p, p); (q, q); (r, r)\}$

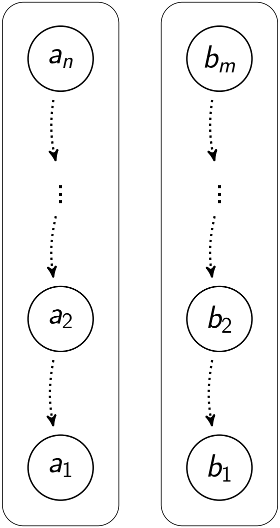
Possible future (“may”)

$[\![\Diamond_1\ q]\!] = \{(p, p); (p, q); (p, r)\}$

Necessary future (“must”)

$[\![\Box_1\ q]\!] = \varnothing$

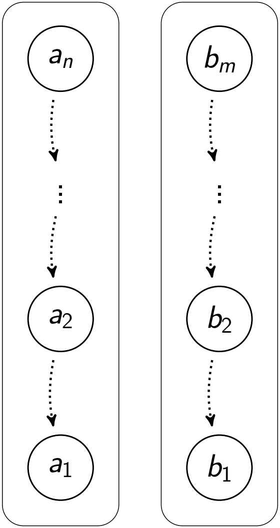
Examples with Polyadic µ-calculus



Least fixed point (μ)

$$\phi = \mu X.(\Box_1 \bot \wedge \Box_2 \bot) \vee \Diamond_1 \Diamond_2 X$$

Examples with Polyadic μ-calculus



Least fixed point (μ)

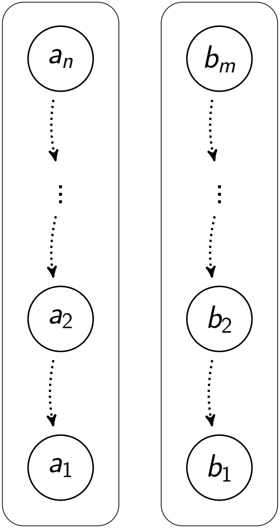
$$\phi = \mu X.(\Box_1\bot \wedge \Box_2\bot) \vee \Diamond_1\Diamond_2 X$$

Iterations:

$$[\![\phi]\!]_0 = \varnothing$$
$$[\![\phi]\!]_1 = \{(a_1, b_1)\}$$
$$[\![\phi]\!]_2 = \{(a_1, b_1); (a_2, b_2)\}$$
$$[\![\phi]\!]_3 = \{(a_1, b_1); (a_2, b_2); (a_3, b_3)\}$$

⋮

Examples with Polyadic µ-calculus



Least fixed point (μ)

$$\phi = \mu X.(\Box_1 \bot \wedge \Box_2 \bot) \vee \Diamond_1 \Diamond_2 X$$

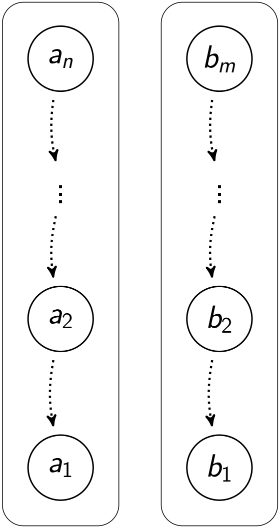
Iterations:

$$\llbracket\phi\rrbracket_0 = \varnothing$$
$$\llbracket\phi\rrbracket_1 = \{(a_1, b_1)\}$$
$$\llbracket\phi\rrbracket_2 = \{(a_1, b_1); (a_2, b_2)\}$$
$$\llbracket\phi\rrbracket_3 = \{(a_1, b_1); (a_2, b_2); (a_3, b_3)\}$$
$$\vdots$$

Generalization:

$$\llbracket\phi\rrbracket = \{(a_i, b_i) \mid i \in [1; \min(m, n)]\}$$

Examples with Polyadic µ-calculus



Least fixed point (μ)

$$\phi = \mu X.(\Box_1 \bot \wedge \Box_2 \bot) \vee \Diamond_1 \Diamond_2 X$$

Iterations:

$$[\![\phi]\!]_0 = \varnothing$$
$$[\![\phi]\!]_1 = \{(a_1, b_1)\}$$
$$[\![\phi]\!]_2 = \{(a_1, b_1); (a_2, b_2)\}$$
$$[\![\phi]\!]_3 = \{(a_1, b_1); (a_2, b_2); (a_3, b_3)\}$$
$$\vdots$$

Generalization:

$$[\![\phi]\!] = \{(a_i, b_i) \mid i \in [1; \min(m, n)]\}$$

Idea: use one (or n) token per automata