# Analyse de la dynamique des modèles biologiques par programmation logique

Léo-Paul Delsaux

Stage effectué au laboratoire CRIStAL de Villeneuve-d'Ascq

29 août 2022

# Introduction

Mots-clés :

- Bio-informatique
- Answer Set Programming (ASP)
- Réseau d'automates asynchrone (AAN)
- État local/global, transition locale/globale, chemin, cycle, automate produit, attracteur

# Hitori

| | | | | |
|---|---|---|---|---|
| 2 | 2 | 1 | 5 | 3 |
| 2 | 3 | 1 | 4 | 5 |
| 1 | 1 | 1 | 3 | 5 |
| 1 | 3 | 5 | 4 | 2 |
| 5 | 4 | 3 | 2 | 1 |

FIGURE – Grille de Hitori (https://fr.wikipedia.org/wiki/Hitori)

# Hitori - Instance

# Hitori - Instance

c(1,1,2). c(2,1,2). c(3,1,1). c(4,1,5). c(5,1,3).
c(1,2,2). c(2,2,3). c(3,2,1). c(4,2,4). c(5,2,5).
c(1,3,1). c(2,3,1). c(3,3,1). c(4,3,3). c(5,3,5).
c(1,4,1). c(2,4,3). c(3,4,5). c(4,4,4). c(5,4,2).
c(1,5,5). c(2,5,4). c(3,5,3). c(4,5,2). c(5,5,1).

# Hitori - Règles

# Hitori - Règles

taille(5).

# Hitori - Règles

taille(5).

*Faits utiles*
colonne(1..S) :- taille(S).
ligne(1..S) :- taille(S).

# Hitori - Règles

taille(5).

*Faits utiles*
colonne(1..S) :- taille(S).
ligne(1..S) :- taille(S).

*Déclaration des colorations*
couleur(blanc). couleur(jaune).

# Hitori - Règles

taille(5).

*Faits utiles*

colonne(1..S) :- taille(S).
ligne(1..S) :- taille(S).

*Déclaration des colorations*

couleur(blanc). couleur(jaune).

*Génération des différents ensembles solutions*

1 { sol(L, C, V) : couleur(V) } 1 :- colonne(C), ligne(L).

# Hitori - Règles (suite)

# Hitori - Règles (suite)

*Il ne peut pas y avoir deux cases jaunes juxtaposées*

:- sol(L, C, jaune), sol(L+1, C, jaune).

# Hitori - Règles (suite)

*Il ne peut pas y avoir deux cases jaunes juxtaposées*

:- sol(L, C, jaune), sol(L+1, C, jaune).
:- sol(L, C, jaune), sol(L-1, C, jaune).
:- sol(L, C, jaune), sol(L, C+1, jaune).
:- sol(L, C, jaune), sol(L, C-1, jaune).

# Hitori - Règles (suite)

*Il ne peut pas y avoir deux cases jaunes juxtaposées*

:- sol(L, C, jaune), sol(L+1, C, jaune).
:- sol(L, C, jaune), sol(L-1, C, jaune).
:- sol(L, C, jaune), sol(L, C+1, jaune).
:- sol(L, C, jaune), sol(L, C-1, jaune).

*Les cases blanches sur une même ligne/colonne ont toutes une valeur différente*

:- sol(L, C, blanc), sol(L, C2, blanc), C != C2, c(L, C, V), c(L, C2, V).

# Hitori - Règles (suite)

*Il ne peut pas y avoir deux cases jaunes juxtaposées*

:- sol(L, C, jaune), sol(L+1, C, jaune).
:- sol(L, C, jaune), sol(L-1, C, jaune).
:- sol(L, C, jaune), sol(L, C+1, jaune).
:- sol(L, C, jaune), sol(L, C-1, jaune).

*Les cases blanches sur une même ligne/colonne ont toutes une valeur différente*

:- sol(L, C, blanc), sol(L, C2, blanc), C != C2, c(L, C, V), c(L, C2, V).
:- sol(L, C, blanc), sol(L2, C, blanc), L != L2, c(L, C, V), c(L2, C, V).

# Hitori - Règles (suite)

# Hitori - Règles (suite)

*L'ensemble des cases blanches est connexe (par déplacement haut/bas/gauche/droite)*

```
chemin((L, C),(L+1, C)) :- sol(L, C, blanc), sol(L+1, C, blanc).
```

# Hitori - Règles (suite)

*L'ensemble des cases blanches est connexe (par déplacement haut/bas/gauche/droite)*

```
chemin((L, C),(L+1, C)) :- sol(L, C, blanc), sol(L+1, C, blanc).
chemin((L, C),(L-1, C)) :- sol(L, C, blanc), sol(L-1, C, blanc).
chemin((L, C),(L, C+1)) :- sol(L, C, blanc), sol(L, C+1, blanc).
chemin((L, C),(L, C-1)) :- sol(L, C, blanc), sol(L, C-1, blanc).
```

# Hitori - Règles (suite)

*L'ensemble des cases blanches est connexe (par déplacement haut/bas/gauche/droite)*

```
chemin((L, C),(L+1, C)) :- sol(L, C, blanc), sol(L+1, C, blanc).
chemin((L, C),(L-1, C)) :- sol(L, C, blanc), sol(L-1, C, blanc).
chemin((L, C),(L, C+1)) :- sol(L, C, blanc), sol(L, C+1, blanc).
chemin((L, C),(L, C-1)) :- sol(L, C, blanc), sol(L, C-1, blanc).
chemin((L1, C1), (L2, C2)) :- chemin((L1, C1), (X, Y)),
chemin((X, Y), (L2, C2)).
```

# Hitori - Règles (suite)

*L'ensemble des cases blanches est connexe (par déplacement haut/bas/gauche/droite)*

```
chemin((L, C),(L+1, C)) :- sol(L, C, blanc), sol(L+1, C, blanc).
chemin((L, C),(L-1, C)) :- sol(L, C, blanc), sol(L-1, C, blanc).
chemin((L, C),(L, C+1)) :- sol(L, C, blanc), sol(L, C+1, blanc).
chemin((L, C),(L, C-1)) :- sol(L, C, blanc), sol(L, C-1, blanc).
chemin((L1, C1), (L2, C2)) :- chemin((L1, C1), (X, Y)),
chemin((X, Y), (L2, C2)).
:- sol(L, C, blanc), sol(L2, C2, blanc), not chemin((L, C), (L2,
C2)).
```

# Hitori résolu

| | | | | |
|---|---|---|---|---|
| 2 | 2 | 1 | 5 | 3 |
| 2 | 3 | 1 | 4 | 5 |
| 1 | 1 | 1 | 3 | 5 |
| 1 | 3 | 5 | 4 | 2 |
| 5 | 4 | 3 | 2 | 1 |

FIGURE – Grille de Hitori résolu (https://fr.wikipedia.org/wiki/Hitori)

# Sokoban

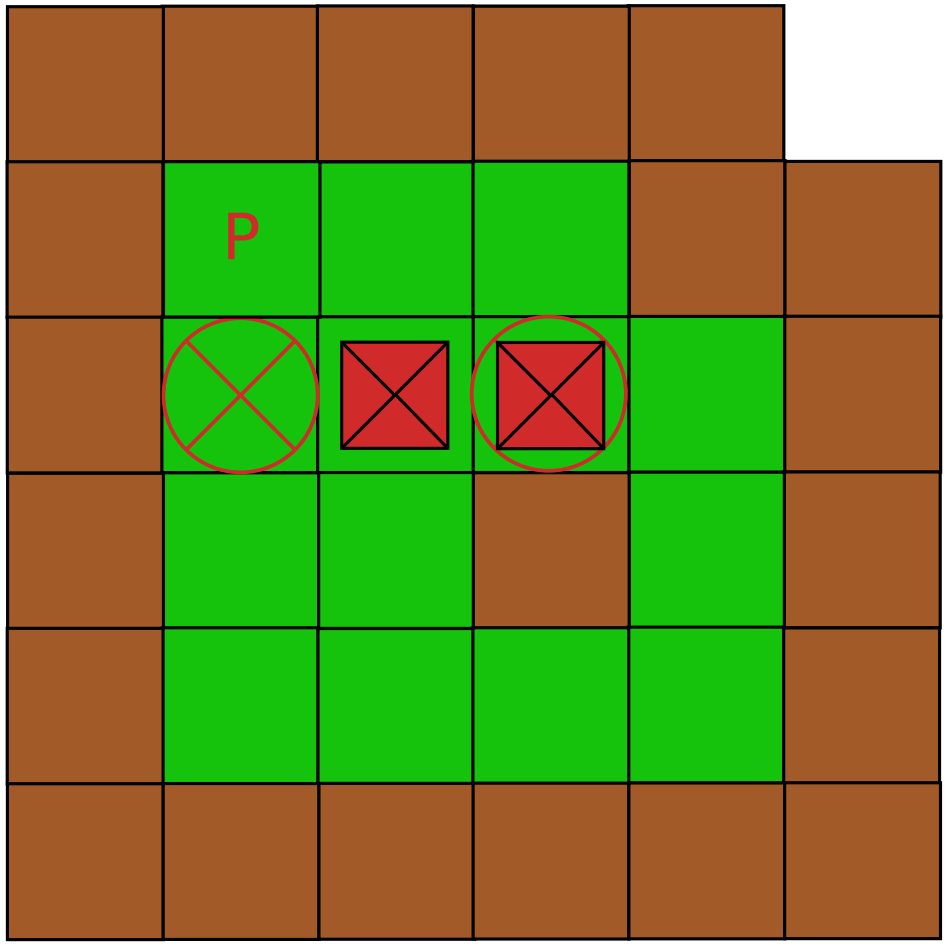



FIGURE – Grille de Sokoban. P symbolise le joueur, les ronds rouges sont les cases d'arrivée, et les carrés rouges représentent les caisses.

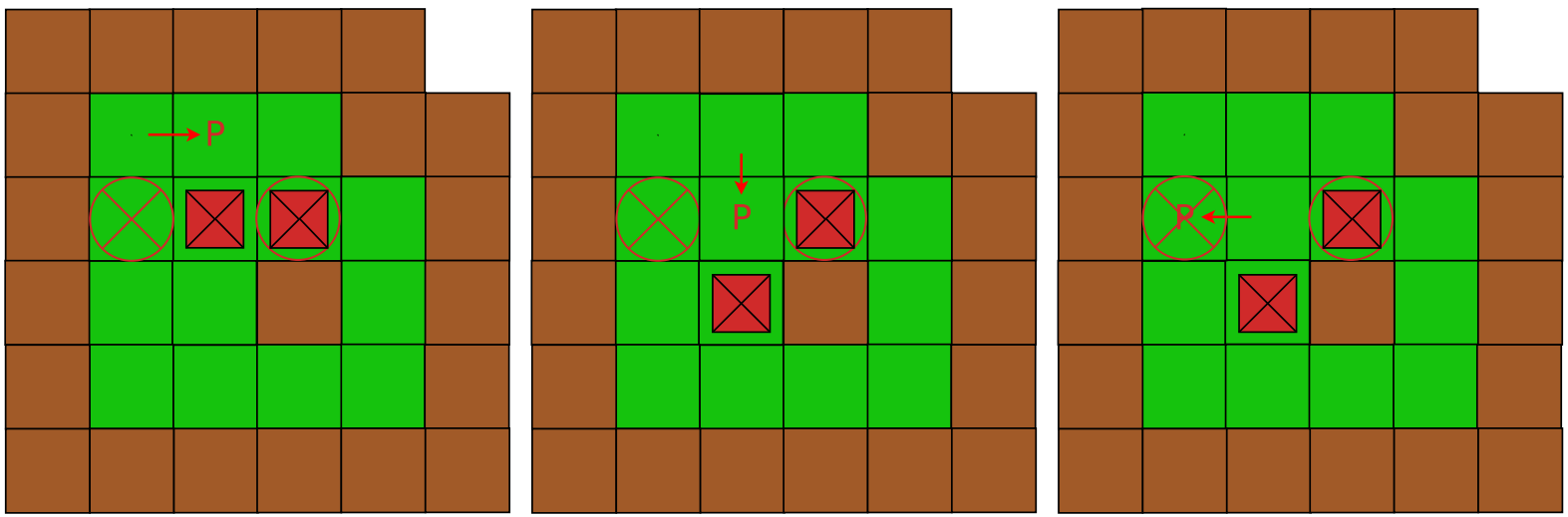
# Sokoban - Stratégies de calcul

# Sokoban - Stratégies de calcul

Naïf : on considère un coup en tant que déplacement possible du personnage

# Sokoban - Stratégies de calcul

Naïf : on considère un coup en tant que déplacement possible du personnage



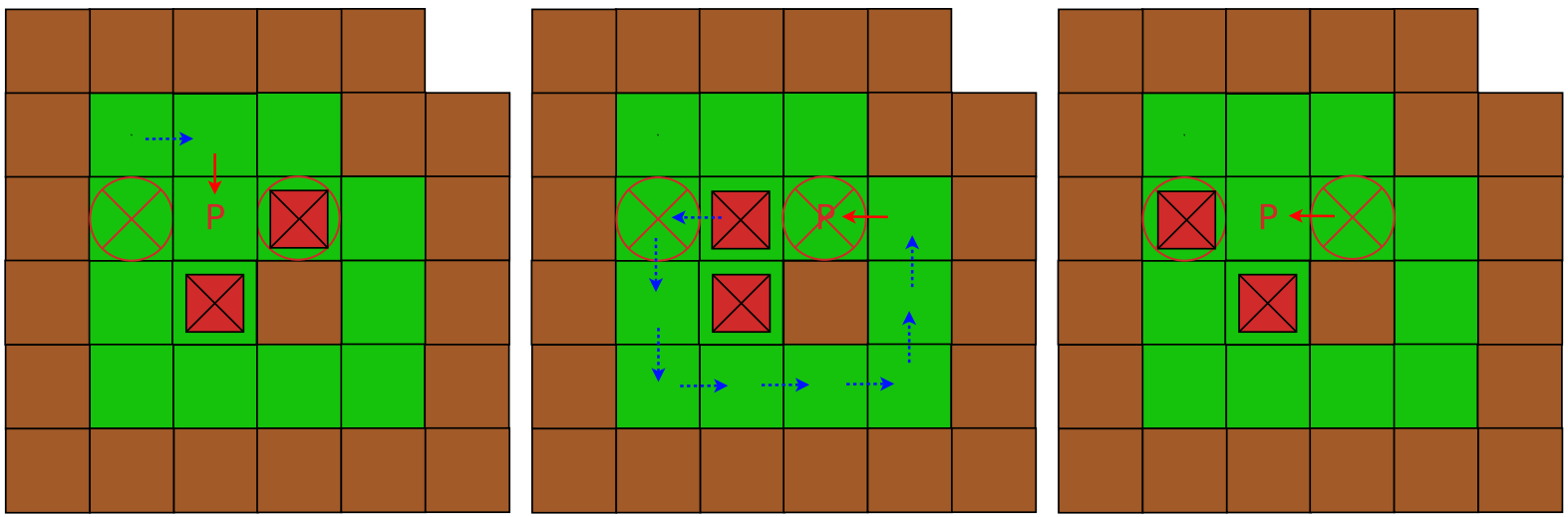
# Sokoban - Stratégies de calcul

# Sokoban - Stratégies de calcul

Plus rapide : on ne considère que les coups de déplacement de caisse. On considère alors l'ensemble connexe des cases atteignables depuis celles du personnage

# Sokoban - Stratégies de calcul

Plus rapide : on ne considère que les coups de déplacement de caisse. On considère alors l'ensemble connexe des cases atteignables depuis celles du personnage



# AAN - Schéma

# AAN - Schéma

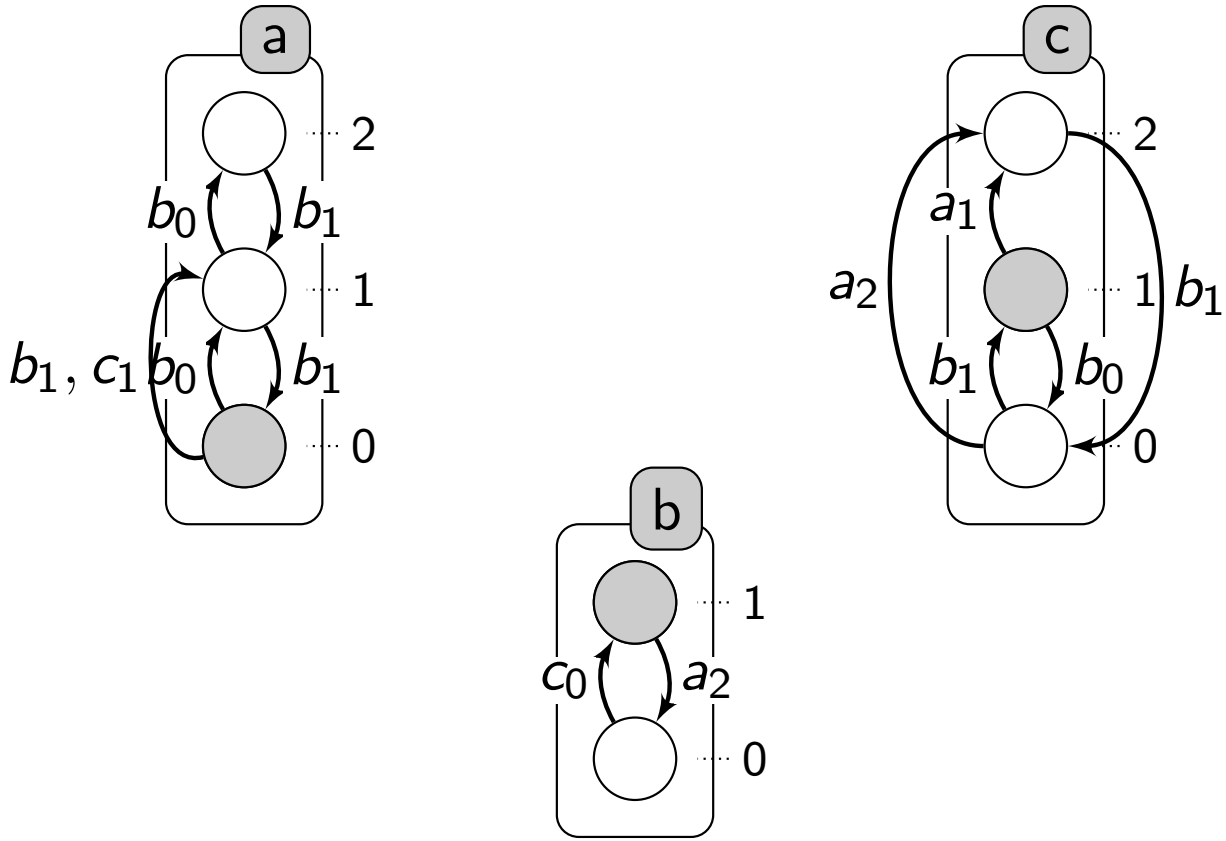



FIGURE – Schéma qui fera office d'exemple de référence

# AAN - Formalismes

Un réseau d'automates asynchrone est un triplet $(\Sigma, S, T)$, avec :

# AAN - Formalismes

Un réseau d'automates asynchrone est un triplet $(\Sigma, S, T)$, avec :

- $\Sigma = \{a, b, ...\}$ est un ensemble fini d'automates non vides.

# AAN - Formalismes

Un réseau d'automates asynchrone est un triplet $(\Sigma, S, T)$, avec :

- $\Sigma = \{a, b, ...\}$ est un ensemble fini d'automates non vides.
  $\Sigma = \{a, b, c\}$

# AAN - Formalismes

Un réseau d'automates asynchrone est un triplet $(\Sigma, S, T)$, avec :

- $\Sigma = \{a, b, ...\}$ est un ensemble fini d'automates non vides.
  $\Sigma = \{a, b, c\}$

- Si $C_a$ est le nombre d'états d'un automate $a$, alors
  $S_a = \{a_0, a_1, ..., a_{C_a - 1}\}$ est l'ensemble des **états locaux** de $a$.
  $S = \prod_{a \in \Sigma} S_a$ est l'ensemble fini des **états globaux**, et
  $LS = \bigcup_{a \in \Sigma} S_a$ représente l'ensemble de tous les états locaux.

# AAN - Formalismes

Un réseau d'automates asynchrone est un triplet $(\Sigma, S, T)$, avec :

- $\Sigma = \{a, b, ...\}$ est un ensemble fini d'automates non vides.
  $\Sigma = \{a, b, c\}$

- Si $C_a$ est le nombre d'états d'un automate $a$, alors
  $S_a = \{a_0, a_1, ..., a_{C_a - 1}\}$ est l'ensemble des **états locaux** de $a$.
  $S = \prod_{a \in \Sigma} S_a$ est l'ensemble fini des **états globaux**, et
  $LS = \bigcup_{a \in \Sigma} S_a$ représente l'ensemble de tous les états locaux.
  $S_a = \{a_0, a_1, a_2\}$, $S_b = \{b_0, b_1\}$ et $S_c = \{c_0, c_1, c_2\}$

# AAN - Formalismes

# AAN - Formalismes

- Pour chaque $a \in \Sigma$,
  $T_a \subseteq \left\{ a_i \xrightarrow{I} a_j \in S_a \times \mathbb{P}(LS/S_a) \times S_a | a_i \neq a_j \right\}$ est l'ensemble des **transitions locales** d'un automate $a$. $T = \bigcup_{a \in \Sigma} T_a$ est l'ensemble des transitions locales du modèle.

# AAN - Formalismes

- Pour chaque $a \in \Sigma$,
  $T_a \subseteq \left\{ a_i \xrightarrow{I} a_j \in S_a \times \mathbb{P}(LS/S_a) \times S_a | a_i \neq a_j \right\}$ est l'ensemble des **transitions locales** d'un automate $a$. $T = \bigcup_{a \in \Sigma} T_a$ est l'ensemble des transitions locales du modèle.

$$T_a = \left\{ a_0 \xrightarrow{b_0} a_1, a_0 \xrightarrow{b_1, c_1} a_1, a_1 \xrightarrow{b_1} a_0, a_1 \xrightarrow{b_0} a_2, a_2 \xrightarrow{b_1} a_1 \right\}$$

$$T_b = \left\{ b_0 \xrightarrow{c_0} b_1, b_1 \xrightarrow{a_2} b_0 \right\}$$

$$T_c = \left\{ c_0 \xrightarrow{b_1} c_1, c_0 \xrightarrow{a_2} c_2, c_1 \xrightarrow{b_0} c_0, c_1 \xrightarrow{a_1} c_2, c_2 \xrightarrow{b_1} c_0 \right\}$$

# AAN - Traduction de l'exemple en ASP

En ASP, on définit l'exemple de référence en deux temps.

# AAN - Traduction de l'exemple en ASP

En ASP, on définit l'exemple de référence en deux temps.

- On déclare les niveaux :
  automaton_level("a", 0..2).
  automaton_level("b", 0..1).
  automaton_level("c", 0..2).

# AAN - Traduction de l'exemple en ASP

En ASP, on définit l'exemple de référence en deux temps.

- On déclare les niveaux :
  automaton_level("a", 0..2).
  automaton_level("b", 0..1).
  automaton_level("c", 0..2).
- Et les transitions à l'aide de labels :
  condition(t1, "a", 0). target(t1, "a", 1). condition(t1, "b", 0).
  condition(t2, "a", 1). target(t2, "a", 2). condition(t2, "b", 0).
  [...](11 lignes supplémentaires)
  condition(t12, "a", 0). target(t12, "a", 1). condition(t12, "b", 1). condition(t12, "c", 1).

# Sémantiques

# Sémantiques

On s'intéressera à 3 sémantiques :

# Sémantiques

On s'intéressera à 3 sémantiques :

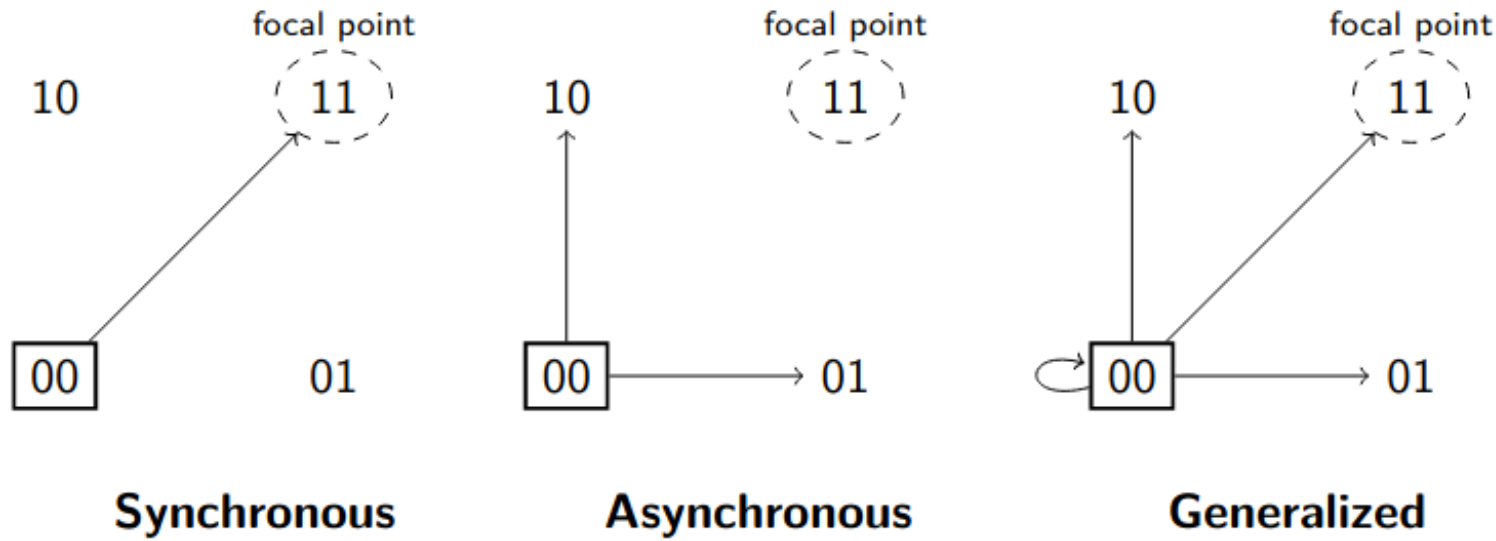



FIGURE – Schéma repris du pdf Folschette_Bioss18.pdf de Maxime Folschette

# Attracteurs

# Attracteurs

Un **domaine de piège** est un ensemble d'états globaux duquel toutes les transitions globales pour la sémantique choisie mènent à un élément de ce domaine.

# Attracteurs

Un **domaine de piège** est un ensemble d'états globaux duquel toutes les transitions globales pour la sémantique choisie mènent à un élément de ce domaine.

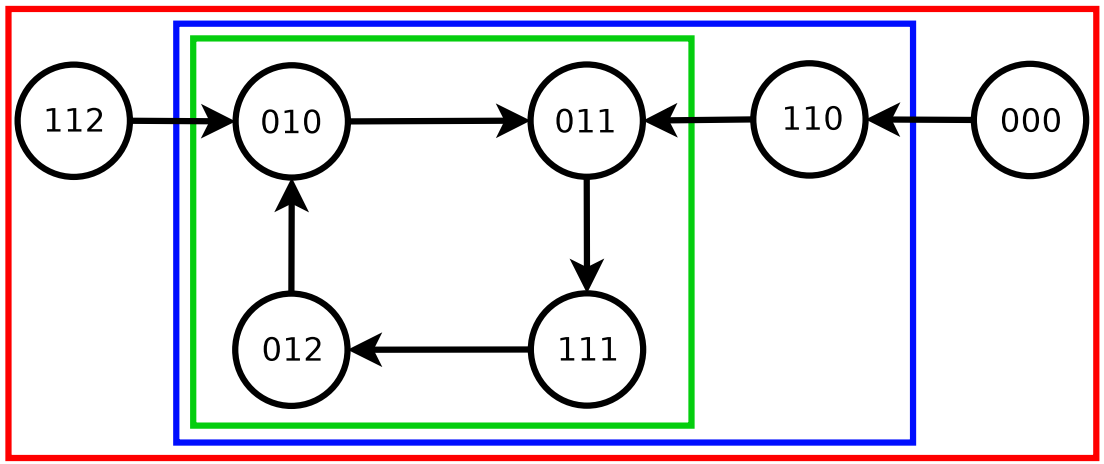



FIGURE – Sous-graphe du graphe produit de l'exemple de référence avec la sémantique synchrone. 3 des 6 domaines de piège y sont encadrés.

# Attracteurs (suite)

# Attracteurs (suite)

Un **attracteur** est un domaine de piège minimal en terme d'inclusion ensembliste.

# Attracteurs (suite)

Un **attracteur** est un domaine de piège minimal en terme d'inclusion ensembliste.

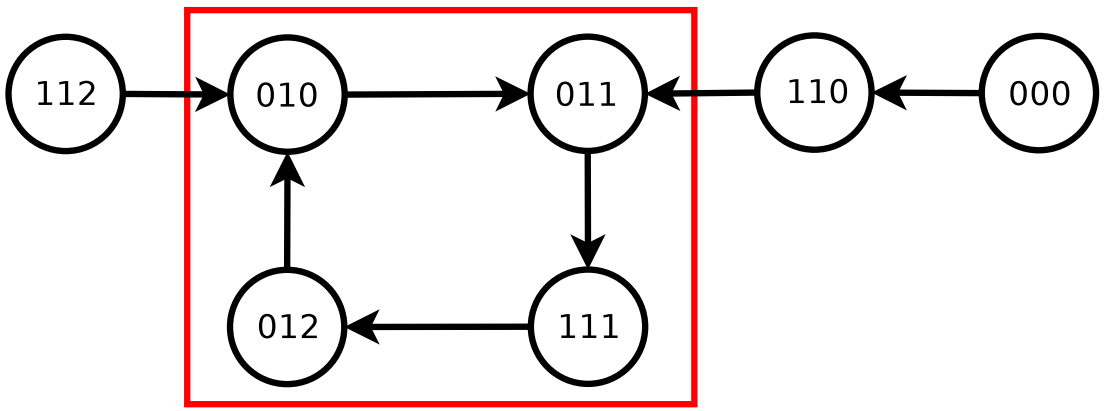



FIGURE – Sous-graphe du graphe produit de l'exemple de référence avec la sémantique synchrone. Le seul attracteur y est encadré.

# Attracteurs (suite)

Un **attracteur** est un domaine de piège minimal en terme d'inclusion ensembliste.

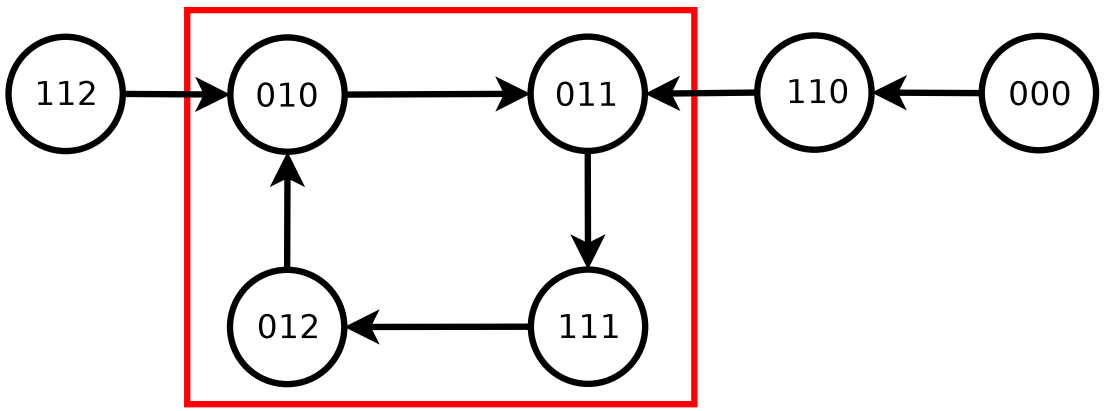



FIGURE – Sous-graphe du graphe produit de l'exemple de référence avec la sémantique synchrone. Le seul attracteur y est encadré.

**Lemme :** Les attracteurs d'un AAN sont exactement les domaines de piège cycliques.

# Problématique

Pour la version synchrone, le code préexistant ne fonctionnait que partiellement : seuls les attracteurs simples (dont les états globaux ont exactement une transition sortante) étaient trouvés.

# Problématique

Pour la version synchrone, le code préexistant ne fonctionnait que partiellement : seuls les attracteurs simples (dont les états globaux ont exactement une transition sortante) étaient trouvés.

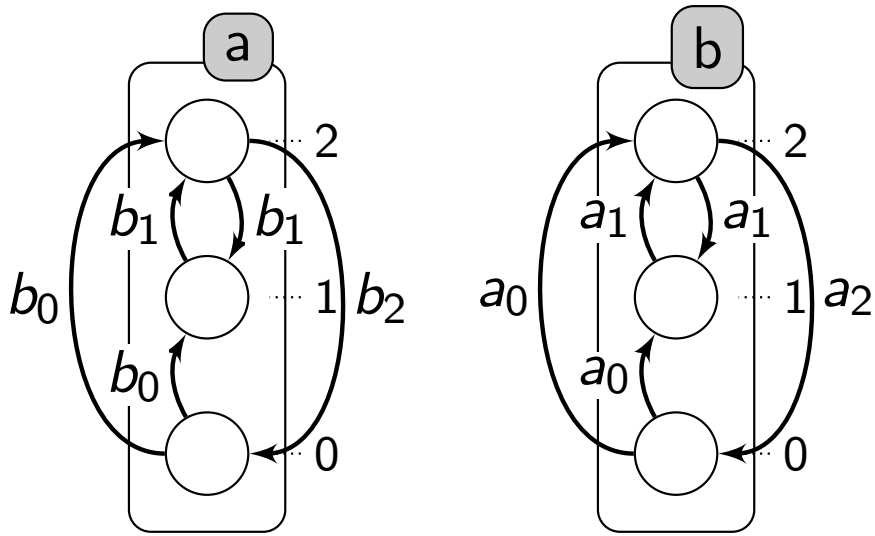



FIGURE – Exemple d'AAN sur lequel le code pré-existant ne trouvait pas l'attracteur

# Problématique

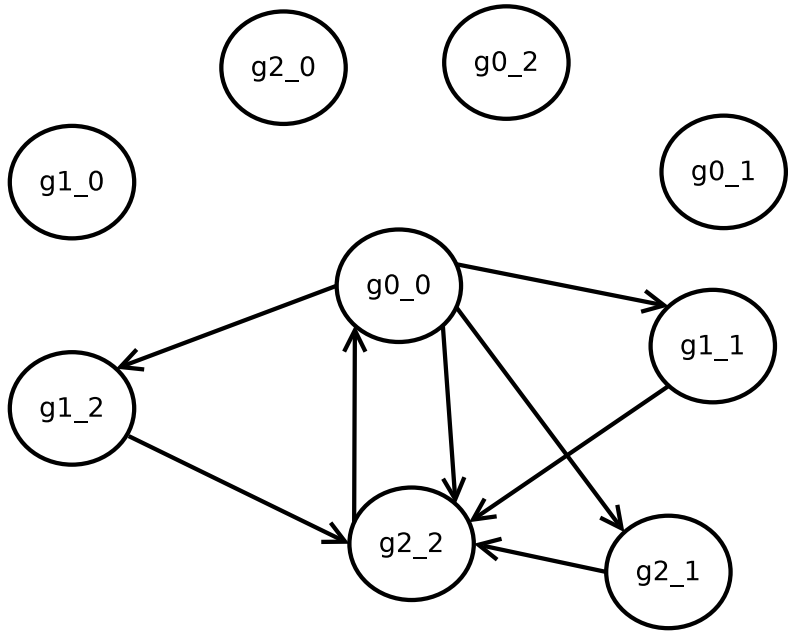



FIGURE – Graphe produit de l'AAN précédent pour la sémantique synchrone

# Pistes étudiées

# Pistes étudiées

Solutions étudiées :

# Pistes étudiées

Solutions étudiées :

- correction de la troisième contrainte en Python

# Pistes étudiées

Solutions étudiées :

- correction de la troisième contrainte en Python
- utilisation des états globaux en ASP

# Correction de la troisième contrainte en Python

# Correction de la troisième contrainte en Python

Une fois que l'on a généré tous les chemins possibles dans un AAN à l'aide d'agrégats, il nous faut filtrer les ensembles-solutions qui nous intéressent. On doit alors respecter 3 contraintes :

# Correction de la troisième contrainte en Python

Une fois que l'on a généré tous les chemins possibles dans un AAN à l'aide d'agrégats, il nous faut filtrer les ensembles-solutions qui nous intéressent. On doit alors respecter 3 contraintes :

- avoir un cycle

# Correction de la troisième contrainte en Python

Une fois que l'on a généré tous les chemins possibles dans un AAN à l'aide d'agrégats, il nous faut filtrer les ensembles-solutions qui nous intéressent. On doit alors respecter 3 contraintes :

- avoir un cycle
- tout les états globaux du chemin visités après l'étape de fin du visite du cycle doivent être des éléments de ce dernier

# Correction de la troisième contrainte en Python

Une fois que l'on a généré tous les chemins possibles dans un AAN à l'aide d'agrégats, il nous faut filtrer les ensembles-solutions qui nous intéressent. On doit alors respecter 3 contraintes :

- avoir un cycle
- tout les états globaux du chemin visités après l'étape de fin du visite du cycle doivent être des éléments de ce dernier
- toutes les transitions globales jouables depuis chacun des éléments du cycle doivent arriver dans un autre élément de ce cycle ( = domaine piège)

# Correction de la troisième contrainte en Python - Résultats

| n | exam. |
|---|---|
| $\|\Sigma\|$ | 4 |
| 2 | 2 |
| 5 | 2 |
| 10 | 2 |
| 15 | 2 |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec python)

| n | exam. |
|---|---|
| 2 | .051 |
| 5 | .052 |
| 10 | .054 |
| 15 | .093 |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec python) - timeout(T.O) = 100s

# Correction de la troisième contrainte en Python - Résultats

| n | exam. | lamb. |
|---|---|---|
| $\|\Sigma\|$ | 4 | 4 |
| 2 | 2 | 2 |
| 5 | 2 | 2 |
| 10 | 2 | 2 |
| 15 | 2 | 2 |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec python)

| n | exam. | lamb. |
|---|---|---|
| 2 | .051 | .053 |
| 5 | .052 | .060 |
| 10 | .054 | .076 |
| 15 | .093 | .096 |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec python) - timeout(T.O) = 100s

# Correction de la troisième contrainte en Python - Résultats

| n | exam. | lamb. | trp. |
|---|---|---|---|
| $\|\Sigma\|$ | 4 | 4 | 4 |
| 2 | 2 | 2 | 0 |
| 5 | 2 | 2 | 1 |
| 10 | 2 | 2 | 1 |
| 15 | 2 | 2 | 1 |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec python)

| n | exam. | lamb. | trp. |
|---|---|---|---|
| 2 | .051 | .053 | .044 |
| 5 | .052 | .060 | .039 |
| 10 | .054 | .076 | .050 |
| 15 | .093 | .096 | .051 |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec python) - timeout(T.O) = 100s

# Correction de la troisième contrainte en Python - Résultats

| n | exam. | lamb. | trp. | fis. |
|---|---|---|---|---|
| $\|\Sigma\|$ | 4 | 4 | 4 | 9 |
| 2 | 2 | 2 | 0 | 1 |
| 5 | 2 | 2 | 1 | 1 |
| 10 | 2 | 2 | 1 | 1 |
| 15 | 2 | 2 | 1 | 1 |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec python)

| n | exam. | lamb. | trp. | fis. |
|---|---|---|---|---|
| 2 | .051 | .053 | .044 | .047 |
| 5 | .052 | .060 | .039 | .057 |
| 10 | .054 | .076 | .050 | .084 |
| 15 | .093 | .096 | .051 | .108 |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec python) - timeout(T.O) = 100s

# Correction de la troisième contrainte en Python - Résultats

| n | exam. | lamb. | trp. | fis. | mamm. |
|---|---|---|---|---|---|
| $\|\Sigma\|$ | 4 | 4 | 4 | 9 | 10 |
| 2 | 2 | 2 | 0 | 1 | 0 |
| 5 | 2 | 2 | 1 | 1 | 0 |
| 10 | 2 | 2 | 1 | 1 | 1 |
| 15 | 2 | 2 | 1 | 1 | 1 |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec python)

| n | exam. | lamb. | trp. | fis. | mamm. |
|---|---|---|---|---|---|
| 2 | .051 | .053 | .044 | .047 | .047 |
| 5 | .052 | .060 | .039 | .057 | .043 |
| 10 | .054 | .076 | .050 | .084 | .082 |
| 15 | .093 | .096 | .051 | .108 | .123 |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec python) - timeout(T.O) = 100s

# Correction de la troisième contrainte en Python - Résultats

| n | exam. | lamb. | trp. | fis. | mamm. | tcr. |
|---|---|---|---|---|---|---|
| $\|\Sigma\|$ | 4 | 4 | 4 | 9 | 10 | 40 |
| 2 | 2 | 2 | 0 | 1 | 0 | 0 |
| 5 | 2 | 2 | 1 | 1 | 0 | 0 |
| 10 | 2 | 2 | 1 | 1 | 1 | 1 |
| 15 | 2 | 2 | 1 | 1 | 1 | 1 |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec python)

| n | exam. | lamb. | trp. | fis. | mamm. | tcr. |
|---|---|---|---|---|---|---|
| 2 | .051 | .053 | .044 | .047 | .047 | .049 |
| 5 | .052 | .060 | .039 | .057 | .043 | .079 |
| 10 | .054 | .076 | .050 | .084 | .082 | .201 |
| 15 | .093 | .096 | .051 | .108 | .123 | .362 |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec python) - timeout(T.O) = 100s

# Correction de la troisième contrainte en Python - Résultats

| n | exam. | lamb. | trp. | fis. | mamm. | tcr. | t-helper |
|---|---|---|---|---|---|---|---|
| $\|\Sigma\|$ | 4 | 4 | 4 | 9 | 10 | 40 | 101 |
| 2 | 2 | 2 | 0 | 1 | 0 | 0 | 8878+ |
| 5 | 2 | 2 | 1 | 1 | 0 | 0 | 5477+ |
| 10 | 2 | 2 | 1 | 1 | 1 | 1 | 4072+ |
| 15 | 2 | 2 | 1 | 1 | 1 | 1 | 2850+ |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec python)

| n | exam. | lamb. | trp. | fis. | mamm. | tcr. | t-helper |
|---|---|---|---|---|---|---|---|
| 2 | .051 | .053 | .044 | .047 | .047 | .049 | T.O |
| 5 | .052 | .060 | .039 | .057 | .043 | .079 | T.O |
| 10 | .054 | .076 | .050 | .084 | .082 | .201 | T.O |
| 15 | .093 | .096 | .051 | .108 | .123 | .362 | T.O |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec python) - timeout(T.O) = 100s

# Utilisation des états globaux en ASP

# Utilisation des états globaux en ASP

Une autre manière de gérer la troisième contrainte consiste à créer des prédicats pour les états globaux, et de mémoriser dans la sémantique quels sont les coups jouables depuis un état global, et non une étape temporelle donnée.

# Seconde solution : travail sur les états globaux - Résultats

| n | exam. |
|---|---|
| $\|\Sigma\|$ | 4 |
| 2 | 2 |
| 5 | 2 |
| 10 | 2 |
| 15 | 2 |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec états globaux)

| n | exam. |
|---|---|
| 2 | 3.724 |
| 5 | 6.457 |
| 10 | 11.349 |
| 15 | 18.767 |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec états globaux) - timeout(T.O) = 100s

# Seconde solution : travail sur les états globaux - Résultats

| n | exam. | lamb. |
|---|---|---|
| $\|\Sigma\|$ | 4 | 4 |
| 2 | 2 | 2 |
| 5 | 2 | 2 |
| 10 | 2 | |
| 15 | 2 | |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec états globaux)

| n | exam. | lamb. |
|---|---|---|
| 2 | 3.724 | 43.623 |
| 5 | 6.457 | 71.786 |
| 10 | 11.349 | T.O |
| 15 | 18.767 | T.O |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec états globaux) - timeout(T.O) = 100s

# Seconde solution : travail sur les états globaux - Résultats

| n | exam. | lamb. | trp. |
|---|---|---|---|
| $\|\Sigma\|$ | 4 | 4 | 4 |
| 2 | 2 | 2 | 0 |
| 5 | 2 | 2 | 0 |
| 10 | 2 | | 0 |
| 15 | 2 | | 1 |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec états globaux)

| n | exam. | lamb. | trp. |
|---|---|---|---|
| 2 | 3.724 | 43.623 | 4.155 |
| 5 | 6.457 | 71.786 | 6.288 |
| 10 | 11.349 | T.O | 11.561 |
| 15 | 18.767 | T.O | 19.636 |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec états globaux) - timeout(T.O) = 100s

# Seconde solution : travail sur les états globaux - Résultats

| n | exam. | lamb. | trp. | fis. |
|---|---|---|---|---|
| $\|\Sigma\|$ | 4 | 4 | 4 | 9 |
| 2 | 2 | 2 | 0 | |
| 5 | 2 | 2 | 0 | |
| 10 | 2 | | 0 | |
| 15 | 2 | | 1 | |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec états globaux)

| n | exam. | lamb. | trp. | fis. |
|---|---|---|---|---|
| 2 | 3.724 | 43.623 | 4.155 | T.O |
| 5 | 6.457 | 71.786 | 6.288 | T.O |
| 10 | 11.349 | T.O | 11.561 | T.O |
| 15 | 18.767 | T.O | 19.636 | T.O |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec états globaux) - timeout(T.O) = 100s

# Seconde solution : travail sur les états globaux - Résultats

| n | exam. | lamb. | trp. | fis. | mamm. |
|---|---|---|---|---|---|
| $\|\Sigma\|$ | 4 | 4 | 4 | 9 | 10 |
| 2 | 2 | 2 | 0 | | |
| 5 | 2 | 2 | 0 | | |
| 10 | 2 | | 0 | | |
| 15 | 2 | | 1 | | |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec états globaux)

| n | exam. | lamb. | trp. | fis. | mamm. |
|---|---|---|---|---|---|
| 2 | 3.724 | 43.623 | 4.155 | T.O | T.O |
| 5 | 6.457 | 71.786 | 6.288 | T.O | T.O |
| 10 | 11.349 | T.O | 11.561 | T.O | T.O |
| 15 | 18.767 | T.O | 19.636 | T.O | T.O |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec états globaux) - timeout(T.O) = 100s

# Seconde solution : travail sur les états globaux - Résultats

| n | exam. | lamb. | trp. | fis. | mamm. | tcr. |
|---|---|---|---|---|---|---|
| $\|\Sigma\|$ | 4 | 4 | 4 | 9 | 10 | 40 |
| 2 | 2 | 2 | 0 | | | |
| 5 | 2 | 2 | 0 | | | |
| 10 | 2 | | 0 | | | |
| 15 | 2 | | 1 | | | |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec états globaux)

| n | exam. | lamb. | trp. | fis. | mamm. | tcr. |
|---|---|---|---|---|---|---|
| 2 | 3.724 | 43.623 | 4.155 | T.O | T.O | T.O |
| 5 | 6.457 | 71.786 | 6.288 | T.O | T.O | T.O |
| 10 | 11.349 | T.O | 11.561 | T.O | T.O | T.O |
| 15 | 18.767 | T.O | 19.636 | T.O | T.O | T.O |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec états globaux) - timeout(T.O) = 100s

# Seconde solution : travail sur les états globaux - Résultats

| n | exam. | lamb. | trp. | fis. | mamm. | tcr. | t-helper |
|---|---|---|---|---|---|---|---|
| $\|\Sigma\|$ | 4 | 4 | 4 | 9 | 10 | 40 | 101 |
| 2 | 2 | 2 | 0 | | | | |
| 5 | 2 | 2 | 0 | | | | |
| 10 | 2 | | 0 | | | | |
| 15 | 2 | | 1 | | | | |

FIGURE – Nombre d'attracteurs trouvés pour la sémantique synchrone (version avec états globaux)

| n | exam. | lamb. | trp. | fis. | mamm. | tcr. | t-helper |
|---|---|---|---|---|---|---|---|
| 2 | 3.724 | 43.623 | 4.155 | T.O | T.O | T.O | T.O |
| 5 | 6.457 | 71.786 | 6.288 | T.O | T.O | T.O | T.O |
| 10 | 11.349 | T.O | 11.561 | T.O | T.O | T.O | T.O |
| 15 | 18.767 | T.O | 19.636 | T.O | T.O | T.O | T.O |

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec états globaux) - timeout(T.O) = 100s

# Conclusions (et pistes)

# Conclusions (et pistes)

- 2 versions fonctionnelles :
  - une efficace avec du filtrage sous Python
  - l'autre moins efficace avec utilisation d'états globaux (avec quelques fonctions de calcul en Python)

# Conclusions (et pistes)

- 2 versions fonctionnelles :
  - une efficace avec du filtrage sous Python
  - l'autre moins efficace avec utilisation d'états globaux (avec quelques fonctions de calcul en Python)
- Pistes : la seconde version pourrait être améliorée avec de l'incrémental ; considérer des classes d'équivalence des attracteurs, et manipuler des sortes de "bassins d'attraction"

# Remerciements

Merci à :

- l'ENS de Lyon qui m'a proposé ce stage
- Maxime Folschette pour son encadrement
- les personnes au sein de l'équipe BioComputing
- mes quelques collègues stagiaires de bureau
- les auditeurs présents dans cette salle pour leur écoute