Analyse de la dynamique des modèles biologiques par programmation logique

Léo-Paul Delsaux

Stage effectué au laboratoire CRIStAL de Villeneuve-d'Ascq

juin-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

ASP

ASP

parent(moi, papa).

ASP

parent(moi, papa).
parent(papa, papi).

ASP

parent(moi, papa).
parent(papa, papi).

$\Rightarrow$ grandparent(moi, papi).

ASP - Règles

parent(moi, papa).
parent(papa, papi).

ASP - Règles

parent(moi, papa).
parent(papa, papi).

grandparent(moi, papi) :- parent(moi, papa), parent(papa, papi).

ASP - Variables

```
parent(moi, papa).
parent(papa, papi).
```

ASP - Variables

parent(moi, papa).
parent(papa, papi).

grandparent(X, Z) :- parent(X, Y), parent(Y, Z).

ASP - Variables

parent(moi, papa).
parent(papa, papi).

grandparent(X, Z) :- parent(X, Y), parent(Y, Z).

$\Rightarrow$ grandparent(moi, moi) :- parent(moi, moi), parent(moi, moi).
grandparent(moi, moi) :- parent(moi, papa), parent(papa, moi).
[...] (24 lignes supplémentaires)
grandparent(papi, papi) :- parent(papi, papi), parent(papi, papi).

ASP - Agrégats et contraintes

parent(moi, papa).
parent(papa, papi).

ASP - Agrégats et contraintes

parent(moi, papa).
parent(papa, papi).

famille(moi). famille(papa). famille(papi).

ASP - Agrégats et contraintes

```
parent(moi, papa).
parent(papa, papi).

famille(moi). famille(papa). famille(papi).
1 { grandparent(X, Y) :- famille(X), famille(Y) } 1.
```

ASP - Agrégats et contraintes

```
parent(moi, papa).
parent(papa, papi).

famille(moi). famille(papa). famille(papi).
1 { grandparent(X, Y) :- famille(X), famille(Y) } 1.

 :- famille(X), famille(Y), famille(Z),
```

ASP - Agrégats et contraintes

```
parent(moi, papa).
parent(papa, papi).

famille(moi). famille(papa). famille(papi).
1 { grandparent(X, Y) :- famille(X), famille(Y) } 1.

 :- famille(X), famille(Y), famille(Z),
          not grandparent(X, Z), parent(X, Y), parent(Y, Z).
```

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 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)

Hitori - Règles

Hitori - Règles

```
size(5).

val(1..S) :- size(S).
column(1..S) :- size(S).
row(1..S) :- size(S).
```

Hitori - Règles

```
size(5).

val(1..S) :- size(S).
column(1..S) :- size(S).
row(1..S) :- size(S).

color(0;1).
```

Hitori - Règles

```
size(5).

val(1..S) :- size(S).
column(1..S) :- size(S).
row(1..S) :- size(S).

color(0;1).

1 { s(R, C, V) : color(V) } 1 :- column(C), row(R).
```

Hitori - Règles (suite)

```
 :- s(R, C, 1), s(R+1, C, 1).
:- s(R, C, 1), s(R-1, C, 1).
:- s(R, C, 1), s(R, C+1, 1).
:- s(R, C, 1), s(R, C-1, 1).
```

Hitori - Règles (suite)

```
 :- s(R, C, 1), s(R+1, C, 1).
:- s(R, C, 1), s(R-1, C, 1).
:- s(R, C, 1), s(R, C+1, 1).
:- s(R, C, 1), s(R, C-1, 1).

:- s(R, C, 0), s(R, C2, 0), C != C2, c(R, C, V), c(R, C2, V).
:- s(R, C, 0), s(R2, C, 0), R != R2, c(R, C, V), c(R2, C, V).
```

Hitori - Règles (suite)

```
 :- s(R, C, 1), s(R+1, C, 1).
:- s(R, C, 1), s(R-1, C, 1).
:- s(R, C, 1), s(R, C+1, 1).
:- s(R, C, 1), s(R, C-1, 1).

:- s(R, C, 0), s(R, C2, 0), C != C2, c(R, C, V), c(R, C2, V).
:- s(R, C, 0), s(R2, C, 0), R != R2, c(R, C, V), c(R2, C, V).

chemin((R1, C1), (R2, C2)) :- chemin((R1, C1), (X, Y)),
chemin((X, Y), (R2, C2)).
chemin((R, C),(R+1, C)) :- s(R, C, 0), s(R+1, C, 0).
chemin((R, C),(R-1, C)) :- s(R, C, 0), s(R-1, C, 0).
chemin((R, C),(R, C+1)) :- s(R, C, 0), s(R, C+1, 0).
chemin((R, C),(R, C-1)) :- s(R, C, 0), s(R, C-1, 0).
:- s(R, C, 0), s(R2, C2, 0), not chemin((R, C), (R2, C2)).
```

Sokoban

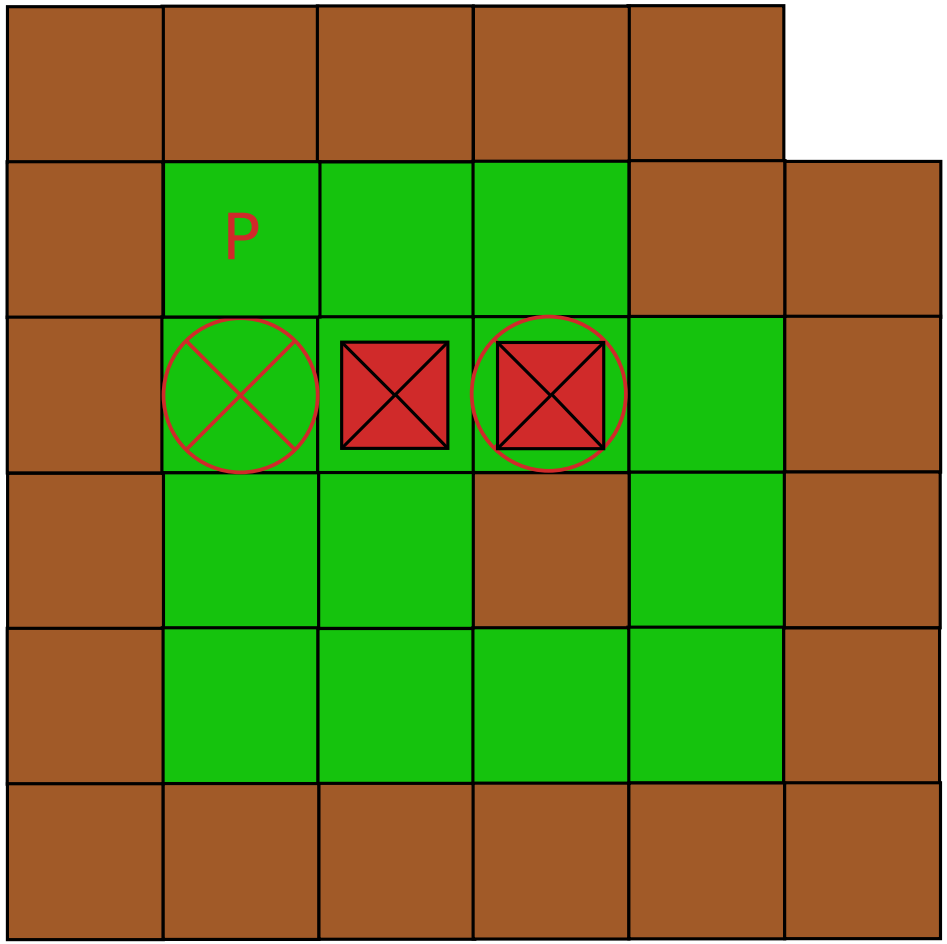

FIGURE – Grille de Sokoban. P symbolise le joueur, les ronds rouges sont les cases d'arrivée, et les carrés rouges représentent les caisses.

Sokoban - Stratégies de calcul

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

Sokoban - Stratégies de calcul

- Naïf : on considère un coup en tant que déplacement possible du personnage
- Plus rapide : on ne considère que les coups de déplacement de caisse. On considère alors un ensemble connexe de cases atteignables depuis celles du personnage
- Autres améliorations mineures :
  - Si une caisse est bloquée dans un coin sans arrivée, on arrête la recherche courante
  - Si deux caisses sont dans un couloir (cernées par des murs), on arrête la recherche courante
  - etc...

AAN - Formalismes

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

AAN - Formalismes

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

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

AAN - Formalismes

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

- $\Sigma = \{a, b, ...\}$ est un ensemble fini d'automates non vides.
- 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 (Σ, S, T), avec :

- $\Sigma = \{a, b, ...\}$ est un ensemble fini d'automates non vides.
- 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.
- Pour chaque $a \in \Sigma$, $T_a \subseteq \left\{ a_i \xrightarrow{I} a_j \in S_a \times \rho(LS/S_a) \times S_a | a_i \neq a_j \right\}$ est l'ensemble des **transitions locales** d'un automate a, avec ρ qui désigne la puissance ensembliste. $T = \bigcup_{a \in \Sigma} T_a$ est l'ensemble des transitions locales du modèle.

AAN - Exemple

On prend l'AAN suivant comme exemple de référence pour ce qui suit :

AAN - Exemple

On prend l'AAN suivant comme exemple de référence pour ce qui suit :

- $\Sigma = \{a, b, c\}$

AAN - Exemple

On prend l'AAN suivant comme exemple de référence pour ce qui suit :

- $\Sigma = \{a, b, c\}$
- $S_a = \{a_0, a_1, a_2\}$, $S_b = \{b_0, b_1\}$ et $S_c = \{c_0, c_1, c_2\}$

AAN - Exemple

On prend l'AAN suivant comme exemple de référence pour ce qui suit :

- $\Sigma = \{a, b, c\}$
- $S_a = \{a_0, a_1, a_2\}$, $S_b = \{b_0, b_1\}$ et $S_c = \{c_0, c_1, c_2\}$
- $T_a = \left\{ a_0 \xrightarrow{b_0} a_1, a_0 \xrightarrow{b_1, c_1} a_1, a_1 \xrightarrow{b_1} a_0, a_1 \xrightarrow{b_0} a_2, a_2 \xrightarrow{b_1} a_1 \right\}$
  $T_b = \left\{ b_0 \xrightarrow{c_0} b_1, b_1 \xrightarrow{a_2} b_0 \right\}$
  $T_c = \left\{ c_0 \xrightarrow{b_1} c_1, c_0 \xrightarrow{a_2} c_2, c_1 \xrightarrow{b_0} c_0, c_1 \xrightarrow{a_1} c_2, c_2 \xrightarrow{b_1} c_0 \right\}$

AAN - Schéma de l'exemple

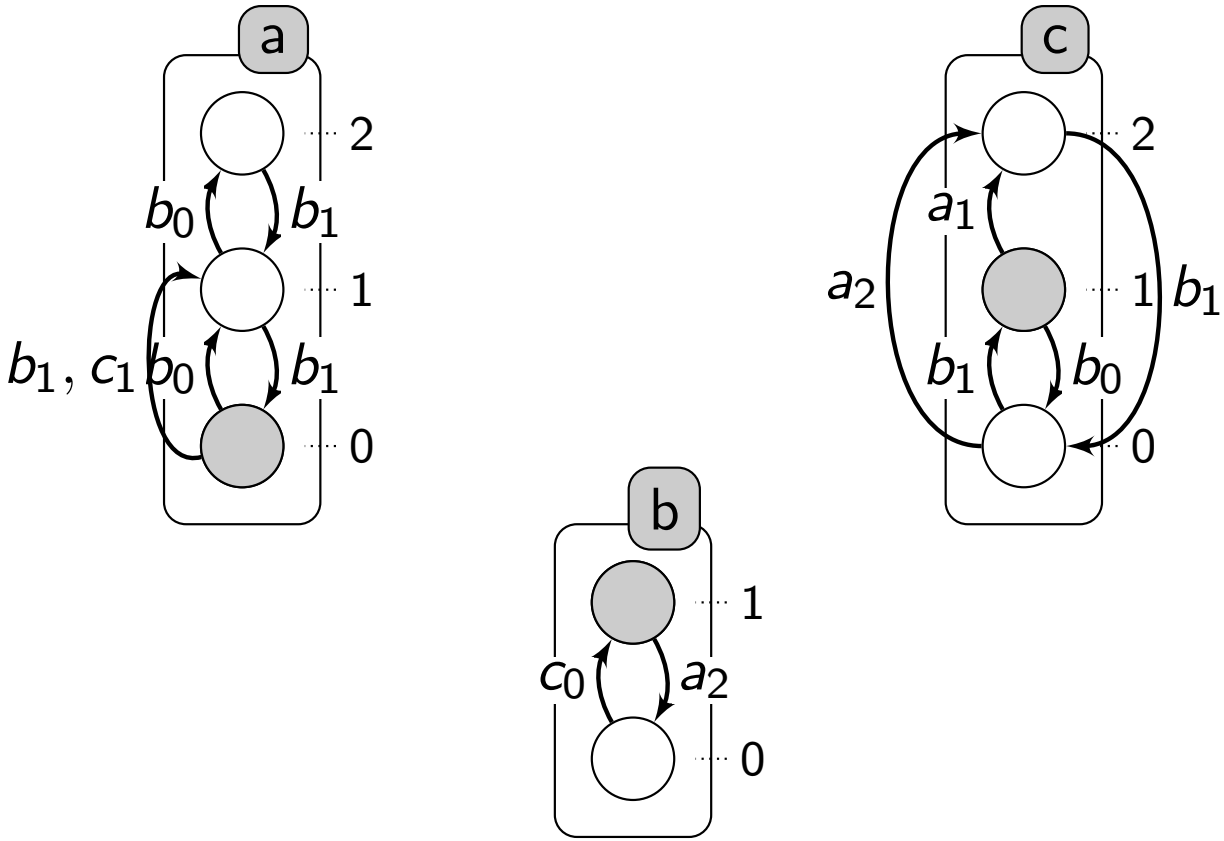

FIGURE – Schéma de notre exemple de référence

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 label :
  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

On s'intéressera à 3 sémantiques principalement :

Sémantiques

On s'intéressera à 3 sémantiques principalement :

- asynchrone : les transitions globales sont exactement les transitions locales jouables

Sémantiques

On s'intéressera à 3 sémantiques principalement :

- asynchrone : les transitions globales sont exactement les transitions locales jouables
- synchrone : les transitions globales sont les ensembles de transitions locales jouables de cardinal maximal (on doit jouer une transition locale pour chaque automate quand c'est psosible)

Sémantiques

On s'intéressera à 3 sémantiques principalement :

- asynchrone : les transitions globales sont exactement les transitions locales jouables
- synchrone : les transitions globales sont les ensembles de transitions locales jouables de cardinal maximal (on doit jouer une transition locale pour chaque automate quand c'est psosible)
- générale : les transitions globales sont générés par les parties de l'ensemble des transitions globales de la sémantique synchrone (on peut jouer une transition locale pour un automate, mais il faut qu'on en joue au moins une)

Attracteurs

Attracteurs

Un **domaine de piège** est un ensemble d'états globaux duquel toutes les transitions globales pour la sémantique choisie mène 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 mène à un élément de ce domaine.
Un **attracteur** est un domaine de piège minimal en terme d'inclusion ensembliste.

Attracteurs

Un **domaine de piège** est un ensemble d'états globaux duquel toutes les transitions globales pour la sémantique choisie mène à un élément de ce domaine.
Un **attracteur** est un domaine de piège minimal en terme d'inclusion ensembliste.

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

Cœur de la contribution personnelle

Cœur de la contribution personnelle

Solutions étudiées :

Cœur de la contribution personnelle

Solutions étudiées :

- correction de la troisième contrainte en Python

Cœur de la contribution personnelle

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.
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)

Correction de la troisième contrainte en Python - Résultats

n	exam.	lamb.
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)

Correction de la troisième contrainte en Python - Résultats

n	exam.	lamb.	trp.
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)

Correction de la troisième contrainte en Python - Résultats

n	exam.	lamb.	trp.	fis.
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)

Correction de la troisième contrainte en Python - Résultats

n	exam.	lamb.	trp.	fis.	mamm.
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)

Correction de la troisième contrainte en Python - Résultats

n	exam.	lamb.	trp.	fis.	mamm.	tcr.
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)

Correction de la troisième contrainte en Python - Résultats

n	exam.	lamb.	trp.	fis.	mamm.	tcr.	t-helper
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)

Seconde solution : travail sur les états globaux - Résultats

Seconde solution : travail sur les états globaux - Résultats

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.
2	2
5	2
10	2

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

n	exam.
2	3.943
5	5.435
10	9.790

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec états globaux)

REFAIRE LES BONS TESTS ! ! !

Seconde solution : travail sur les états globaux - Résultats

n	exam.	lamb.
2	2	2
5	2	2
10	2	

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

n	exam.	lamb.
2	3.943	40.267
5	5.435	58.742
10	9.790	T.O

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec états globaux)

REFAIRE LES BONS TESTS ! ! !

Seconde solution : travail sur les états globaux - Résultats

n	exam.	lamb.	trp.
2	2	2	0
5	2	2	1
10	2		1

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

n	exam.	lamb.	trp.
2	3.943	40.267	4.004
5	5.435	58.742	5.461
10	9.790	T.O	9.965

FIGURE – Temps (en s) de résolution pour la sémantique synchrone (version avec états globaux)

REFAIRE LES BONS TESTS ! ! !

Conclusions et pistes

Conclusions et pistes

- 2 versions fonctionnelles dont une peu efficace

Conclusions et pistes

- 2 versions fonctionnelles dont une peu efficace
- La seconde version pourrait être améliorée avec de l'incrémental

Conclusions et pistes

- 2 versions fonctionnelles dont une peu efficace
- 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"

Conclusions et pistes

- 2 versions fonctionnelles dont une peu efficace
- 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"
- Combiner les deux idées précédentes (?)

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