Model checking
et
encadrement stochastique

Sana YOUNES

Laboratoire PR*i*SM

Université de Versailles - Saint Quentin en Yvelines

Plan

- Introduction
- *Model checking* probabiliste
- Motivation
- Logique PRCTL(*Probabilistic Reward Computational Tree Logic*)
- Encadrement stochastique
- Méthode proposée de vérification des formules
- Exemple
- Conclusion et perspectives

Introduction

- Model checking : technique de vérification de la sureté de fonctionnement des systèmes
- Il existe des méthodes de vérification efficaces pour les systèmes déterministes
- Les systèmes probabilistes peuvent être modélisés par des chaînes de Markov : Comportement stochastique (Chaînes de Markov en temps discret ou continu)

Model checking probabiliste

- Modèle de vérification
  - entrée : Modèle de Markov M , une formule ϕ
  - sortie : $\mathrm{Sat}(\phi) = \{s \in S \mid M, s \models \phi\}$
  - vérification qualitative : réponse : oui ou non
  - vérification quantitative : nécessite un calcul de probabilité ou d'un indice de performance
- *Model checking* symbolique basé sur BDD(Binary Decision Diagram), MTBDD(Multi Terminal BDD)
- Outils : PRISM(*PRobabIlistic Symbolic Model checking*), ET-MCC(*Erlangen-Twente Markov Chain Checker*)

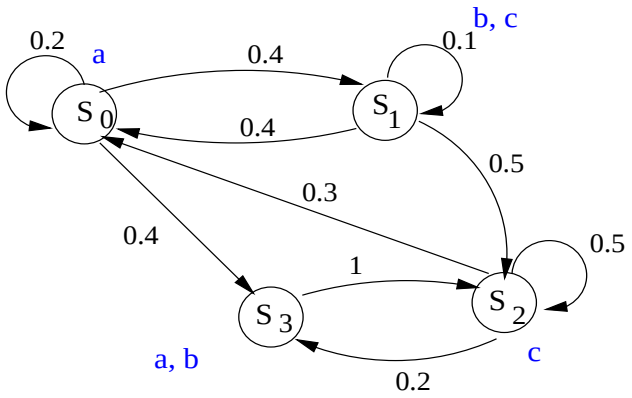
Motivation

- Problème :
  - Dans le cas déterministe, il existe des méthodes efficaces pour la vérification.(BDD(Binary Decision Diagram) , MTBDD(Multi Terminal BDD))
  - Dans le cas probabiliste :Explosion de l'espace d'états dans les modèles de vérification de performabilité
  - Le *model checking* probabiliste ne dispose pas d'algorithmes efficaces contre l'explosion des états
- Idée : Proposer des méthodes de vérification qui reposent sur l'encadrement stochastique permettant de donner des garanties plutôt que des valeurs exactes

Chaînes de Markov discrète avec récompense

DRMC(*Discret Reward Markov Chain*) définie par le tuple :
M=(S, P, L, ρ)

- S est un ensemble fini d'états
- $P : S \times S \longrightarrow [0, 1]$ la matrice de transition, $\sum_{s' \in S} P(s, s') = 1$
- $L : S \longrightarrow 2^{AP}$ fonction d'étiquetage, associe un ensemble de propositions atomiques à chaque état
- $\rho : S \longrightarrow \mathbb{R}_{\geq 0}$ fonction de récompense associe à chaque état une récompense ou coût.



Logique PRCTL(1)

Probabilistic Computational Reward Tree Logic

Basée sur PCTL. Soit a une proposition atomique. I est un intervalle de réel I=$[r_{min}, r_{max}]$

- Syntaxe

  $\phi ::= true \mid a \mid \phi \vee \phi \mid \neg\phi \mid \mathcal{E}_I^n(\phi) \mid \mathcal{E}_I(\phi) \mid \mathcal{I}_I^n(\phi) \mid \mathcal{C}_I^n(\phi)$

- Sémantique

  $s \models true$ pour tout $s \in S$

  $s \models a$ si $a \in L(s)$

  $s \models \phi \vee \psi$ si $s \models \phi$ ou $s \models \psi$

  $s \models \neg\phi$ si $s \not\models \phi$

Logique PRCTL(2)

Partant par l'état initial s :

- Mesure de la récompense stationnaire pour les états qui satisfont ϕ

  $s \models \mathcal{E}_I(\phi)$ si $\sum_{s' \models \phi} \pi(s')\rho(s') \in I$

- Mesure de la récompense transitoire à l'instant n pour les états qui satisfont ϕ.

  $s \models \mathcal{I}^n_I(\phi)$ si $\sum_{s' \models \phi} \pi(s, s', n)\rho(s') \in I$

- Mesure de la récompense cumulée depuis l'instant 0 jusqu'à l'instant n pour les états qui satisfont ϕ

  $s \models \mathcal{C}^n_I(\phi)$ si $\sum_{i=0}^{n-1} \sum_{s' \models \phi} \pi(s, s', i)\rho(s') \in I$

- Mesure du taux de récompense depuis l'instant 0 jusqu'à l'instant n pour les états qui satisfont ϕ

  $s \models \mathcal{E}^n_I(\phi)$ si $\frac{1}{n} \sum_{i=0, s' \models \phi}^{n-1} \sum_{s' \models \phi} \pi(s, s', i)\rho(s') \in I$

Encadrement stochastique et lumpabilité

Definition : Soit X et Y deux variables aléatoires dans un espace totalement ordonné. On dit que X est plus petit que Y suivant l'ordre stochastique fort ie X $\prec_{st}$ Y si E[f(X)] $\leq$E[f(Y)] pour toute fonction f croissante.

Definition(lumpabilité) : Soit X une chaîne DTMC irreductible, P est sa matrice de transition, soit A_k une répartition des états. X est lumpable suivant A_k si pour tout état e et f qui appartiennent à la même macro-état A_i, on a :

$$\sum_{j \in A_k} p_{e,j} = \sum_{j \in A_k} p_{f,j} \qquad \forall A_k \qquad (1)$$

Exemple : lumpabilité

$$\mathbf{P} = \left(\begin{array}{cc|ccc} 0.5 & 0.1 & 0.2 & 0.1 & 0.1 \\ 0.2 & 0.4 & 0.1 & 0.3 & 0 \\ \hline 0.2 & 0.3 & 0.1 & 0.2 & 0.2 \\ 0.25 & 0.25 & 0.3 & 0.1 & 0.1 \\ 0.5 & 0 & 0.1 & 0.2 & 0.2 \end{array}\right)$$

Partitions $A_1 = \{1, 2\}$, $A_2 = \{3, 4, 5\}$

$$\mathbf{L} = \begin{pmatrix} 0.6 & 0.4 \\ 0.5 & 0.5 \end{pmatrix}$$

Algorithme de construction d'une borne st-monotone et lumpable :LIMSUB(1)

Paramètre d'entrée : P matrice stochastique de taille n x n

Paramètre de sortie : Q matrice stochastique de taille m x m, m « n.

Reduction de la taille de l'espace d'état : $S = \{s_0, \cdots, s_n\} \rightarrow$ $\mathcal{A} = \{A_1, ..., A_m\}$

- P $\preceq_{st}$ Q
- Q est $\preceq_{st}$- monotone.
- Q est lumpable selon la partition $\mathcal{A}$

LIMSUB(2)

Possibilité de génération d'une borne supérieure et d'une borne inférieure $Q_{inf} \preceq_{st} P \preceq_{st} Q_{sup}$

En conséquence : $\forall 1 \leq i \leq m$

- Stationnaire :
  $\sum_{j=i}^{m} \pi_{inf}(A_j) \leq \sum_{j=i}^{m} \sum_{s \in A_j} \pi(s) \leq \sum_{j=i}^{m} \pi_{sup}(A_j)$
- Transitoire :
  $\sum_{j=i}^{m} \pi_{inf}(A_j, n) \leq \sum_{j=i}^{m} \sum_{s \in A_j} \pi(s, n) \leq \sum_{j=i}^{m} \pi_{sup}(A_j, n)$

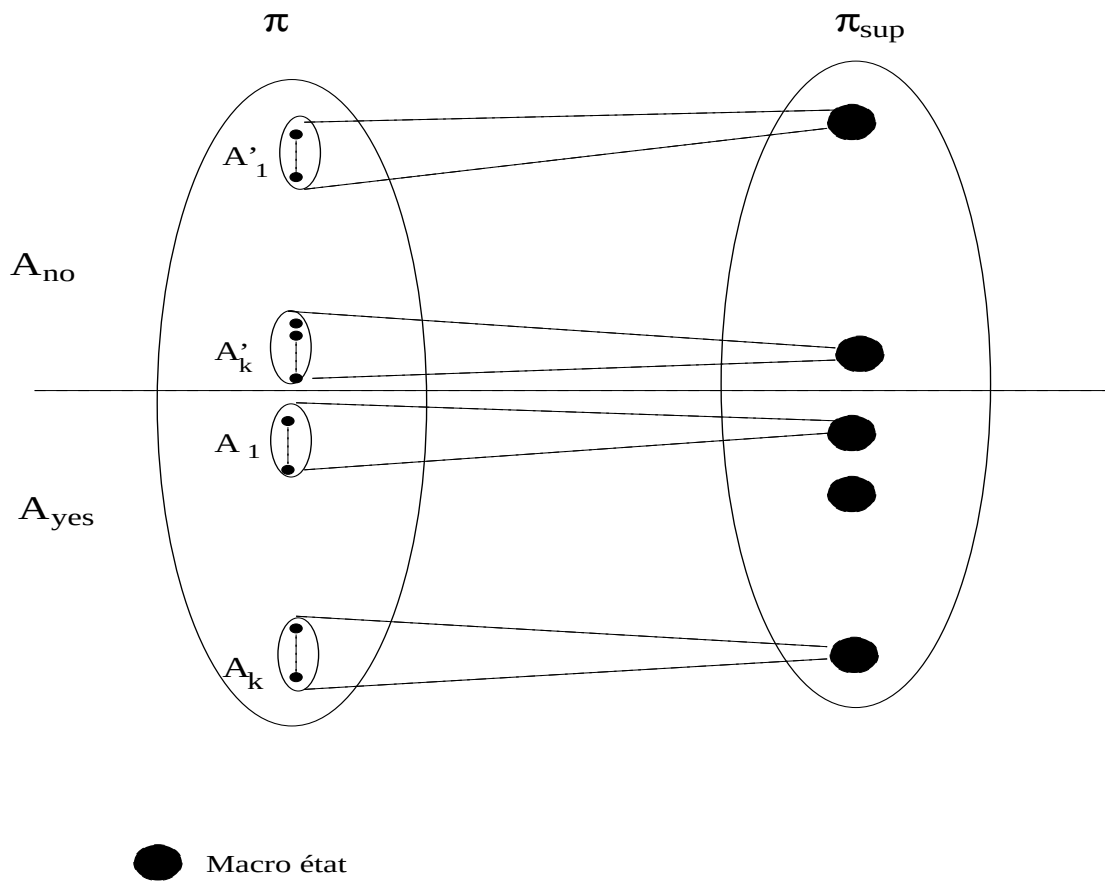
$\pi_{inf}(A_j, n)$ et $\pi_{inf}(A_j)$la probabilité transitoire et stationnaire de la borne inférieure d'être dans le macro-état A_j

Méthode proposée de $\mathcal{E}_I(\phi)$?

- $S_{no} = \{s \in S \text{ et } s \not\models \phi\}$, $S_{yes} = \{s \in S \text{ et } s \models \phi\}$

- Ordonner les états suivant $S_{no} = \{s_0, ..., s_n\}$ et $S_{yes} = \{s_{n+1}, ..., s_{N-1}\}$.
- $\mathcal{A} = \mathcal{A}_{no} \cup \mathcal{A}_{yes}$. $\mathcal{A}_{no}$ peut être partitionné ou non.
- Se focaliser sur $\mathcal{A}_{yes}$ et mettre les états avec les récompenses approchées dans la même partition.
- Ordonner suivant les récompenses croissantes, puisque $\preceq_{st}$ est associé aux fonctions croissantes.
- Donc S_{yes} est divisé en k macro-états $\{A_1, A_2, ...A_k\}$ avec $L(A_i) = \cap_{s \in A_i} L(s)$ et $\rho(A_i) = \max\{\rho(s), s \in A_i\}$

Partition et ordre des macro-états



Vérification de $\mathcal{E}_I(\phi)$?

Pour tout $1 \leq i \leq k$

- $r_{min} \leq^? \underbrace{\sum_{Ai \models \phi} \pi_{inf}(A_i)\rho_{inf}(A_i)}_{B_{inf}} \leq \sum_{s' \models \phi} \pi(s')\rho(s') \leq$

  $\underbrace{\sum_{Ai \models \phi} \pi_{sup}(A_i)\rho_{sup}(A_i)}_{B_{sup}} \leq^? r_{max}$

- Si $B_{inf} \geq r_{min}$ et $B_{sup} \leq r_{max}$ alors $s \models \mathcal{E}_I(\phi)$
- Si $B_{sup} \leq r_{min}$ ou $B_{inf} \geq r_{max}$ alors $s \not\models \mathcal{E}_I(\phi)$
- Sinon, on peut pas vérifier la formule en utilisant l'encadrement stochastique.

Vérification de $\mathcal{I}_I^n(\phi)$?

Pour tout $1 \leq i \leq k$

- $r_{min} \leq^? \underbrace{\sum_{A_i \models \phi} \pi_{inf}(A_i, n)\rho_{inf}(A_i)}_{B_{inf}} \leq \sum_{s' \models \phi} \pi(s', n)\rho(s') \leq$

  $\underbrace{\sum_{Ai \models \phi} \pi_{sup}(A_i, n)\rho_{sup}(A_i)}_{B_{sup}} \leq^? r_{max}$

- Si $B_{inf} \geq r_{min}$ et $B_{sup} \leq r_{max}$ alors $s \models \mathcal{I}_I^n(\phi)$
- Si $B_{sup} \leq r_{min}$ ou $B_{inf} \geq r_{max}$ alors $s \not\models \mathcal{I}_I^n(\phi)$

- Sinon, on peut pas vérifier la formule en utilisant l'encadrement stochastique.

Vérification de $\mathcal{C}_I^n(\phi)$?

Pour tout $1 \leq i \leq k$

- $r_{min} \leq^? \underbrace{\sum_{j=0}^{n-1} \sum_{Ai \models \phi} \pi_{inf}(A_i, j)\rho_{inf}(A_i)}_{B_{inf}} \leq$

  $\sum_{j=0}^{n-1} \sum_{s' \models \phi} \pi(s', j)\rho(s') \leq \underbrace{\sum_{j=0}^{n-1} \sum_{Ai \models \phi} \pi_{sup}(A_i, j)\rho_{sup}(A_i)}_{B_{sup}} \leq^? r_{max}$

- Si $B_{inf} \geq r_{min}$ et $B_{sup} \leq r_{max}$ alors $s \models \mathcal{C}_I^n(\phi)$
- Si $B_{sup} \leq r_{min}$ ou $B_{inf} \geq r_{max}$ alors $s \not\models \mathcal{C}_I^n(\phi)$
- Sinon, on peut pas vérifier la formule en utilisant l'encadrement stochastique.

Vérification de $\mathcal{E}_I^n(\phi)$?

Pour tout $1 \leq i \leq k$

- $r_{min} \leq^? \underbrace{\frac{1}{n} \sum_{j=0}^{n-1} \sum_{Ai \models \phi} \pi_{inf}(A_i, j) \rho_{inf}(A_i)}_{B_{inf}} \leq$

  $\frac{1}{n} \sum_{j=0}^{n-1} \sum_{s' \models \phi} \pi(s', j) \rho(s') \leq$

  $\underbrace{\frac{1}{n} \sum_{j=0}^{n-1} \sum_{Ai \models \phi} \pi_{sup}(A_i, j) \rho_{sup}(A_i)}_{B_{sup}} \leq^? r_{max}$

- Si $B_{inf} \geq r_{min}$ et $B_{sup} \leq r_{max}$ alors $s \models \mathcal{E}_I^n(\phi)$

- Si $B_{sup} \leq r_{min}$ ou $B_{inf} \geq r_{max}$ alors $s \not\models \mathcal{E}_I^n(\phi)$
- Sinon, on peut pas vérifier la formule en utilisant l'encadrement stochastique.

Application : file d'attente D/D/1

Un buffer de taille B.

- Processus d'arrivée :

  Arrivée par groupe de taille maximale G. processus d'arrivée est représenté par $a(t_i)$, $[t_{i-1}, t_i[$, $0 \leq a(t_i) \leq G$

- Equation d'évolution :

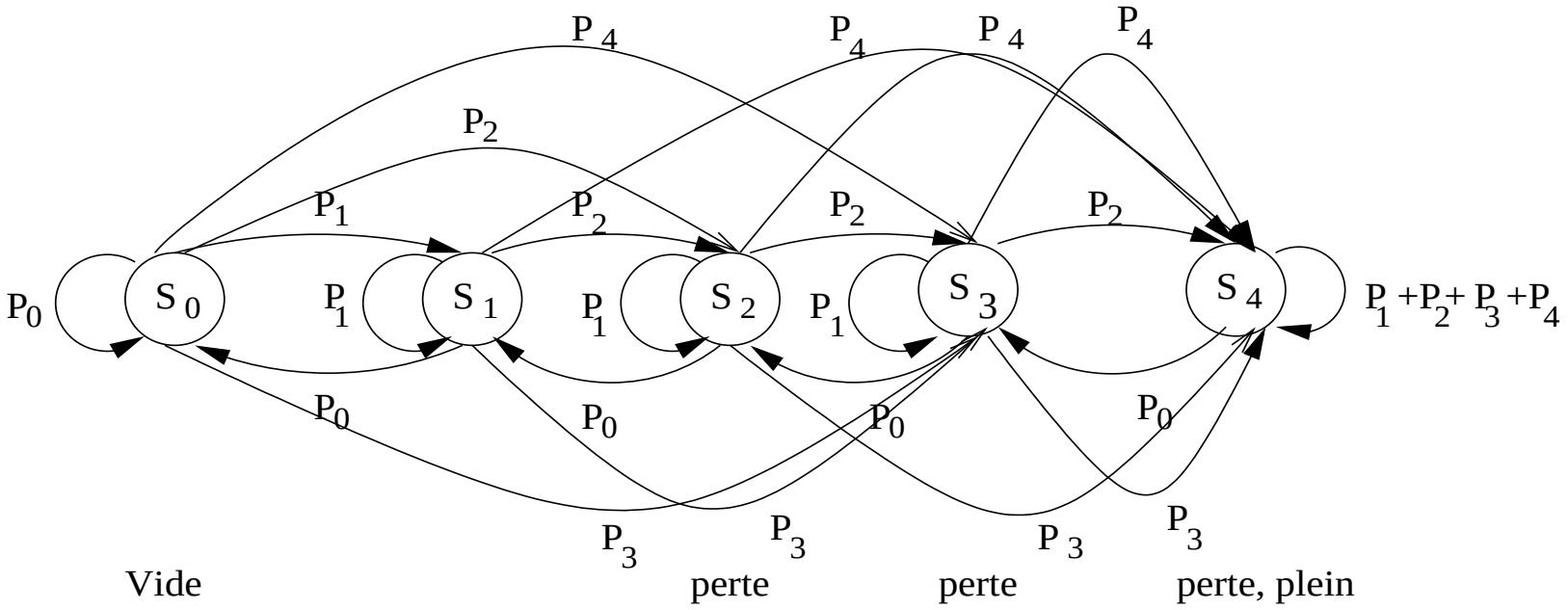
  La durée du service est constante. Soit $N(t_i)$ = nombre total de paquets dans la file durant le slot i.

  $N(t_{i+1}) = \min(B, N(t_i) - \mathbf{1}_{N(t_i)>0} + a(t_{i+1}))$

- La récompense : nombre de paquets perdus par à l'instant t_i :

  $N_{per}(t_i) = \sum_{i=0}^{G} P_i \cdot \max(0, (N(t_i) - \mathbf{1}_{N(t_i>0)} + i - B))$

Exemple : B=4, G=4



On pourra vérifier :

- $\mathcal{E}_{\leq r}(perte)$: Le nombre moyen de paquets perdus par slot à l'état stationnaire ne dépasse pas la valeur r
- $\mathcal{E}^n_{\leq r}(perte)$: Le taux de perte depuis l'instant 0 jusqu'à l'instant n ne dépasse pas la valeur r

Vérification $\mathcal{E}_{\leq r}(perte)$

- $\mathcal{C}^n_{\leq r}(perte)$: Le nombre de paquets perdus depuis l'instant 0 jusqu' à l'instant n ne dépasse pas une valeur r.
- $\mathcal{I}^n_{\leq r}(perte)$ le nombre moyen de paquets perdus à l'instant n est inférieur à la valeur r.

Supposons qu'on veut vérifier $\phi = \mathcal{E}_{\leq r}(perte)$

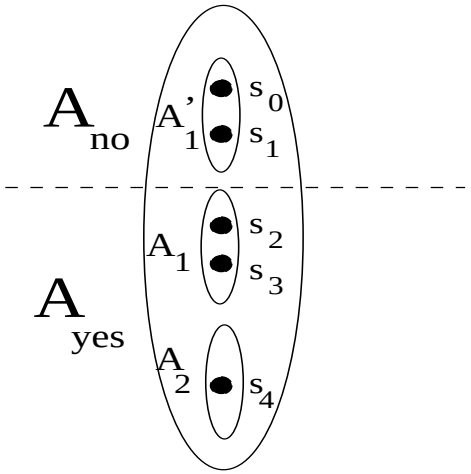
- $S_{yes} = \{s_2, s_3, s_4\}$, $S_{no} = \{s_0, s_1\}$
- $\rho(s_2) = 1 \cdot P_4 \qquad \rho(s_3) = 1 \cdot P_3 + 2 \cdot P_4$
  $\rho(s_4) = 1 \cdot P_2 + 2 \cdot P_3 + 3 \cdot P_4$
- ϕ est satisfaite ssi $\rho(s_2)\pi(s_2) + \rho(s_3)\pi(s_3) + \rho(s_4)\pi(s_4) \leq r$

- On partitionne S_{yes} à deux macro-états A_1 et A_2 : s_2 et s_3 ont des valeurs de récompense les plus rapprochés.

Vérification $\mathcal{E}_{\leq r}(perte)$

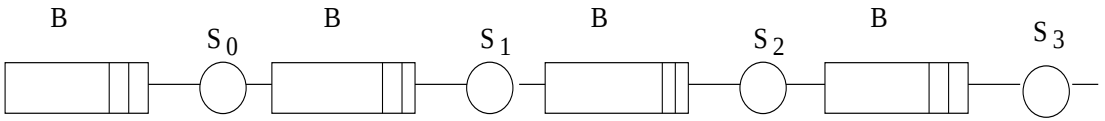
- Donc A_{yes}=$\{A_1, A_2\}$, avec $\rho_{sup}(A_1) = \max(\rho(s_2), \rho(s_3)) = \rho(s_3)$ et $\rho_{sup}(A_2) = \rho(s_4)$

- On ordonne les macro-états suivant la récompense croissante



- Si $\rho_{sup}(A_1)\pi_{sup}(A_1) + \rho_{sup}(A_2)\pi_{sup}(A_2) \leq r$ alors la formule $\phi = \mathcal{E}_{\leq r}(perte)$ est satisfaite.

File en Tandem(1)



- (N_1, N_2, N_3, N_4) est une chaîne de Markov de taille$(B+1)^4$avec :
  taille du Batch de service= G.
  p_{3j}= probabilité que le buffer 3 sert j paquets dans un slot.
  p_{4k}= probabilité que le buffer 4 sert k paquets dans un slot.
  $N_4(t_i)$ Nombre de paquets dans le buffer 4 à l'instant t_i
- Récompense : nombre de paquet perdus par slot dans le buffer 4
  $R(t_i) = \sum_{j=0}^{G} \sum_{k=0}^{G} p_{3j} \cdot p_{4k} \cdot (\max(0, N_4(t_i) + j - k - B))$

File en Tandem(2)

- Propositions atomiques *empty*, *arr-blocked*, *fst-blocked*, *snd-blocked* et *thd-blocked* correspond aux états où le buffer 4 est plein.
- Un macro-état est défini par (N_4, N_3) stockés suivant l'ordre lexicographique afin d'avoir une récompense croissante
- Par suite, la taille de la matrice bornante est $(B+1)^2$
- Pour B=100, l' espace d'état dépassera 10^8

Reslutats numériques

En exécutant sur un
PC de 1.3 GHz. Une méthode de résolution *Gauss seidel* on obtient :

B	20
Nombre d'état	194481
Temps de génération(mn)	10
Taille de la matrice(Mo)	507
Temps de rénumérotation(mn)	29
Taille de la matrice borne(Mo)	5
Temps de résolution(mn)	10

Conclusion et perspective

- Appliquer les méthodes de comparaison stochastique à la vérification de formules de *model checking*
- Fournir des algorithmes de bornes numériques associés à des structures employées en *model checking*(BDD, MTBDD)
- Intégrer ces algorithmes sur des outils de *model checking*(PRISM, ET-MCC)