Génération aléatoire d'automates et analyse des algorithmes de minimisation

Julien David
sous la direction de Frédérique Bassino et Cyril Nicaud

Laboratoire d'Informatique Gaspard Monge
Université Paris-Est Marne-la-Vallée
École Doctorale MSTIC

Mardi 28 septembre 2010

Analyse d'algorithmes

But

Estimer les quantités de ressources utilisées par un algorithme en fonction de la taille de l'instance et indépendamment de l'ordinateur utilisé.

On s'intéresse le plus souvent à la complexité en temps et à la complexité en espace.

Le pire des cas et le cas moyen

Pire cas

Souvent, on analyse la complexité en temps de l'algorithme dans le pire des cas.
$\longrightarrow$ peut n'être pas du tout représentatif du cas pratique.
Exemple : Quicksort

Cas moyen

Apporte une information supplémentaire. On fixe une loi de probabilité sur l'ensemble des instances possibles et on calcule le coût moyen pour cette distribution.

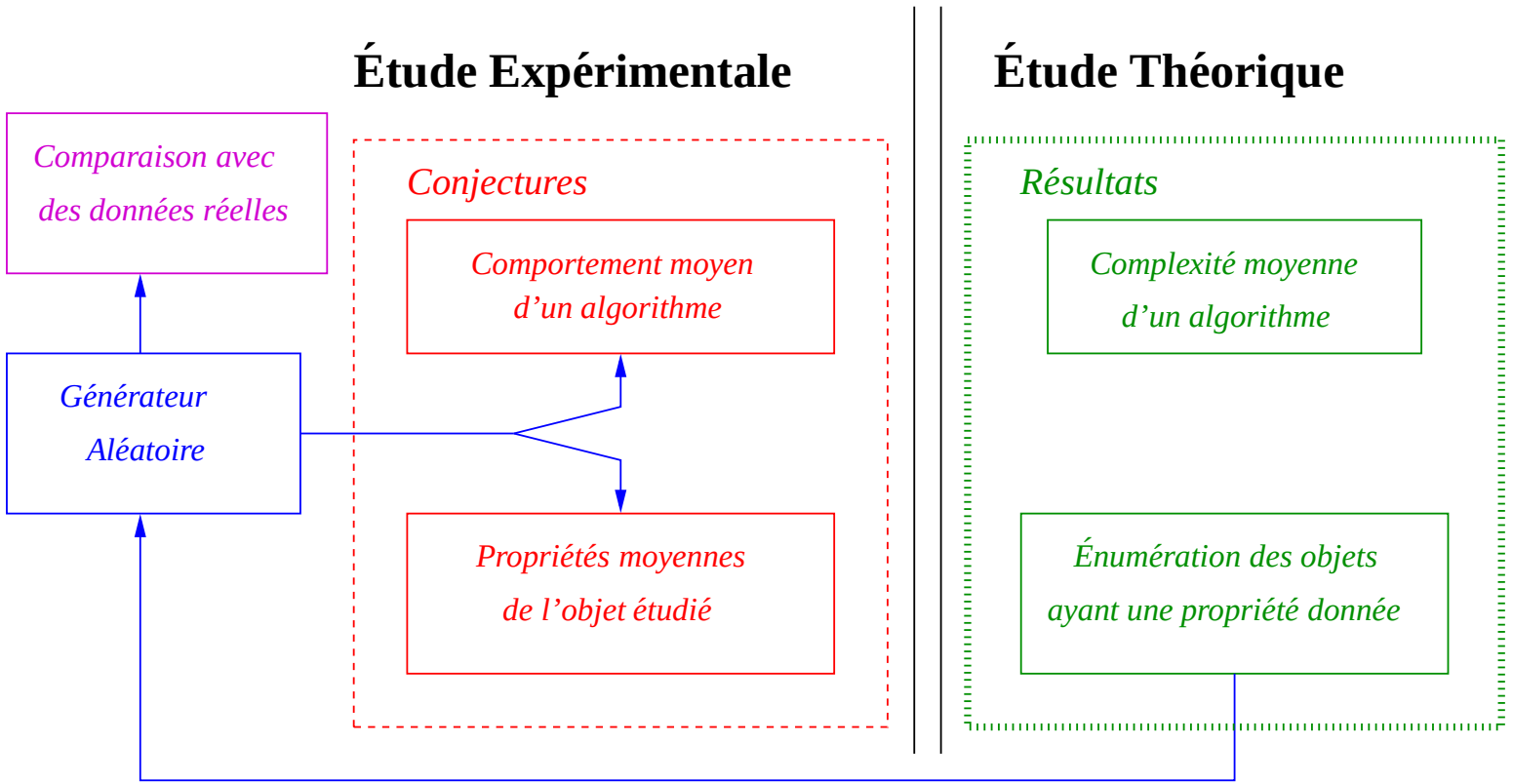
Un générateur aléatoire...

...est un algorithme...

- permettant d'engendrer des objets combinatoires d'une taille donnée, selon une loi de probabilité déterminée.
  On prendra souvent la distribution uniforme sur les objets d'une même taille.

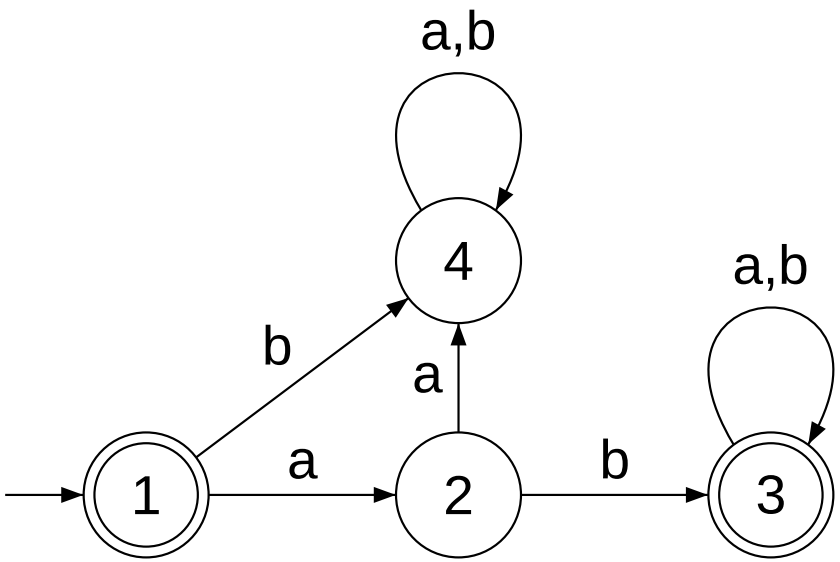
Familles de générateurs

- méthodes *ad hoc*,
- méthodes automatiques : méthode récursive, méthode de Boltzmann.


Étude Expérimentale
Étude Théorique
Comparaison avec des données réelles
Conjectures
Comportement moyen d'un algorithme
Générateur Aléatoire
Propriétés moyennes de l'objet étudié
Résultats
Complexité moyenne d'un algorithme
Énumération des objets ayant une propriété donnée

Automate

Un automate déterministe, accessible et complet à n états sur un alphabet à k lettres.

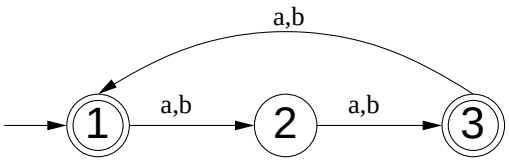


- Ensemble d'états : $Q = \{1, 2, 3, 4\}$
- Alphabet : $\{a, b\}$
- Ensemble d'états initiaux : $\{1\}$
- Ensemble d'états terminaux : $F = \{1, 3\}$
- Langage reconnu par l'automate : $L = \varepsilon + a.b.A^*$

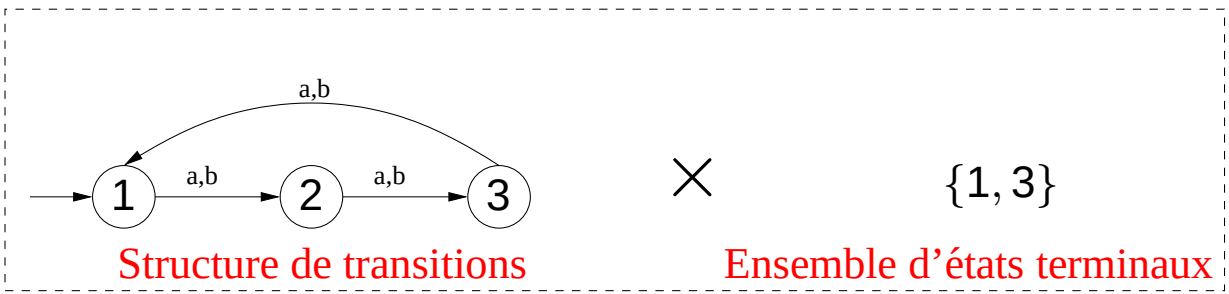
Automates et langages

- L'ensemble des langages reconnaissables par un automate fini est appelé ensemble des langages rationnels.
- Pour un langage rationnel donné, il existe une infinité d'automates finis reconnaissant ce langage.
- Pour tout langage rationnel, il existe un unique automate déterministe accessible complet qui reconnaît ce langage, tel que le nombre d'état est minimal.
  On parle de l'**automate minimal** du langage.

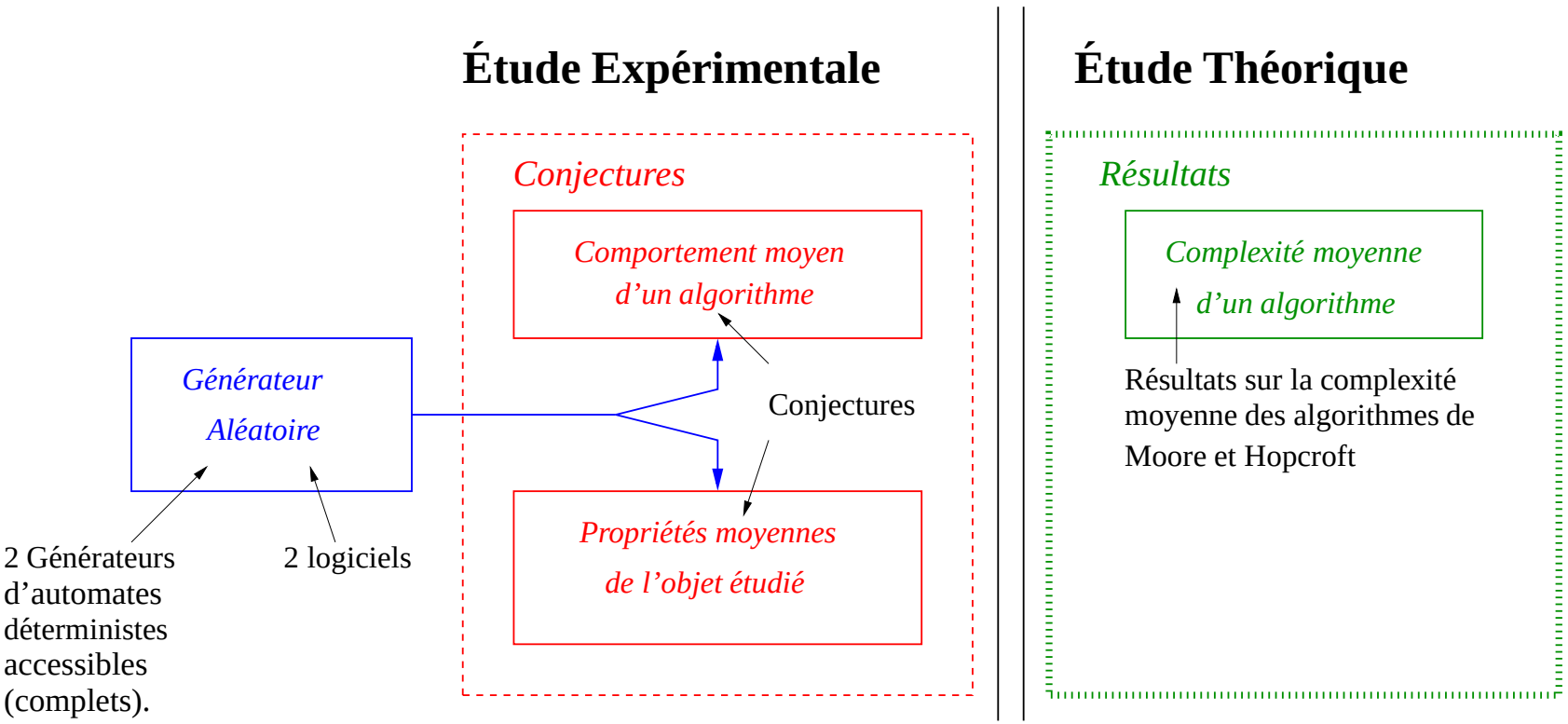
Structures de transitions



Automate



Contributions

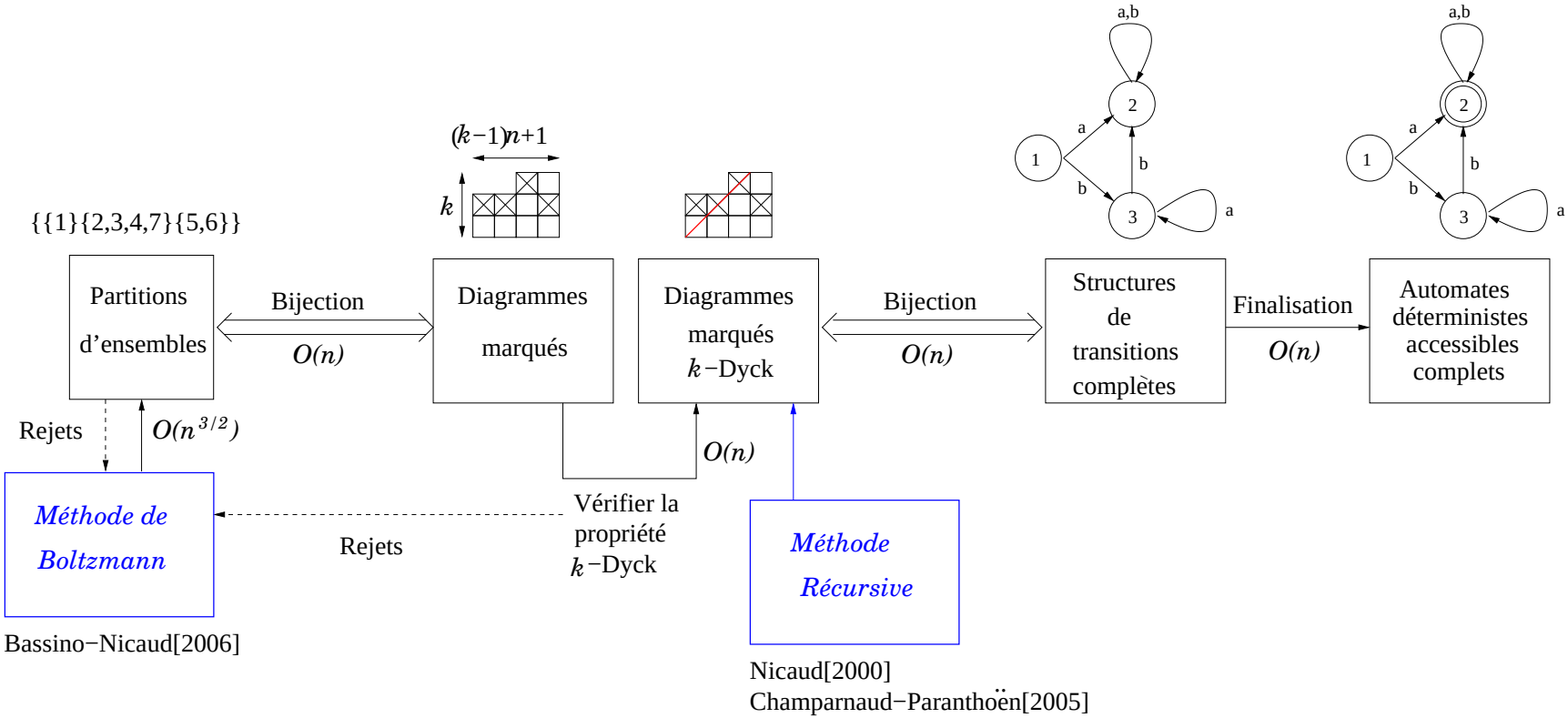


La méthode récursive (Nijenhuis, Wilf 1978) (Flajolet, Zimmermann, Van Cutsem 1994)

- Permet d'obtenir des générateurs aléatoires et exhaustifs
- **Inconvénient pour la génération aléatoire** :
  Nécessite de précalculer et de stocker le nombre exact d'objets de toutes les tailles allant de 1 à la taille souhaitée.

La méthode Boltzmann (Duchon, Flajolet, Louchard, Schaeffer 2004)

- La taille des objets engendrés n'est pas fixe.
- Deux objets de même taille ont la même probabilité d'être engendré.
- On peut paramétrer le générateur afin que la taille de l'objet engendré soit, avec une forte probabilité, proche de celle que l'on souhaite obtenir.
- Pour une génération de taille exacte, on utilise une méthode par rejet.


{{1}{2,3,4,7}{5,6}}
Partitions d'ensembles
Bijection
O(n)
Diagrammes marqués
(k−1)n+1
k
Diagrammes marqués k−Dyck
Bijection
O(n)
Structures de transitions complètes
Finalisation
O(n)
Automates déterministes accessibles complets
a,b
1
2
3
a
b
Rejets
O(n^{3/2})
Méthode de Boltzmann
Bassino−Nicaud[2006]
Rejets
Vérifier la propriété k−Dyck
O(n)
Méthode Récursive
Nicaud[2000]
Champarnaud−Paranthoën[2005]

Reformulation de l'algorithme de Bassino-Nicaud

Bijection [Bassino, David, Nicaud - PuMA'08]

Il existe une bijection entre

- **les structures de transitions déterministes accessibles complètes** avec n états définies sur un alphabet à k lettres.
- **des partitions d'ensembles spéciales** de taille $kn + 1$ en n sous-ensembles non vides.

L'idée

On associe un état à chaque sous-ensemble et une transition à chaque élément de la partition. On construit la structure de transitions en effectuant un parcours préfixe en suivant l'ordre lexicographique.

Reformulation de l'algorithme de Bassino-Nicaud

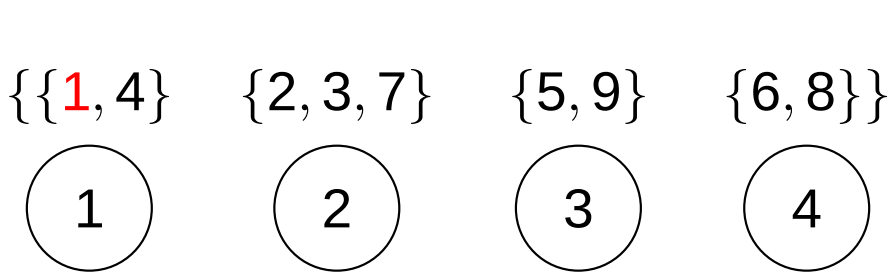
Les sous ensembles sont triés en fonction de leur plus petit élément.

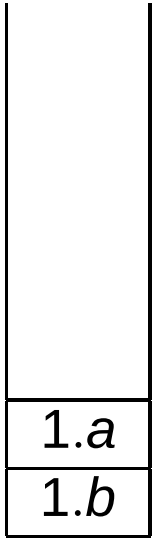
$\{\{1,4\}$ $\{2,3,7\}$ $\{5,9\}$ $\{6,8\}\}$

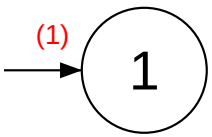
1 2 3 4

$$Alphabet = \{a, b\}$$

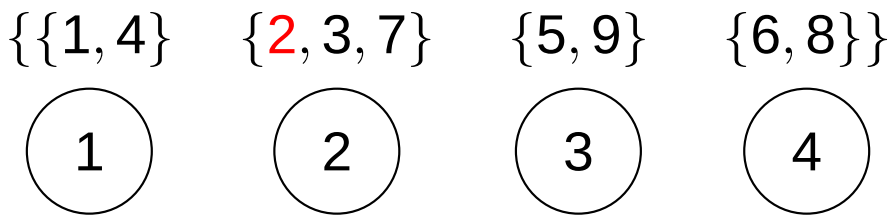
Reformulation de l'algorithme de Bassino-Nicaud

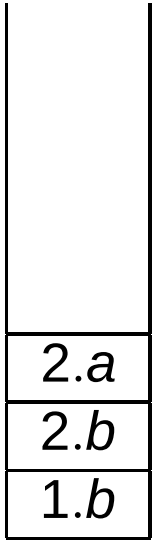


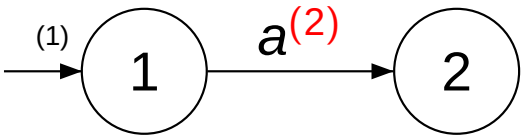




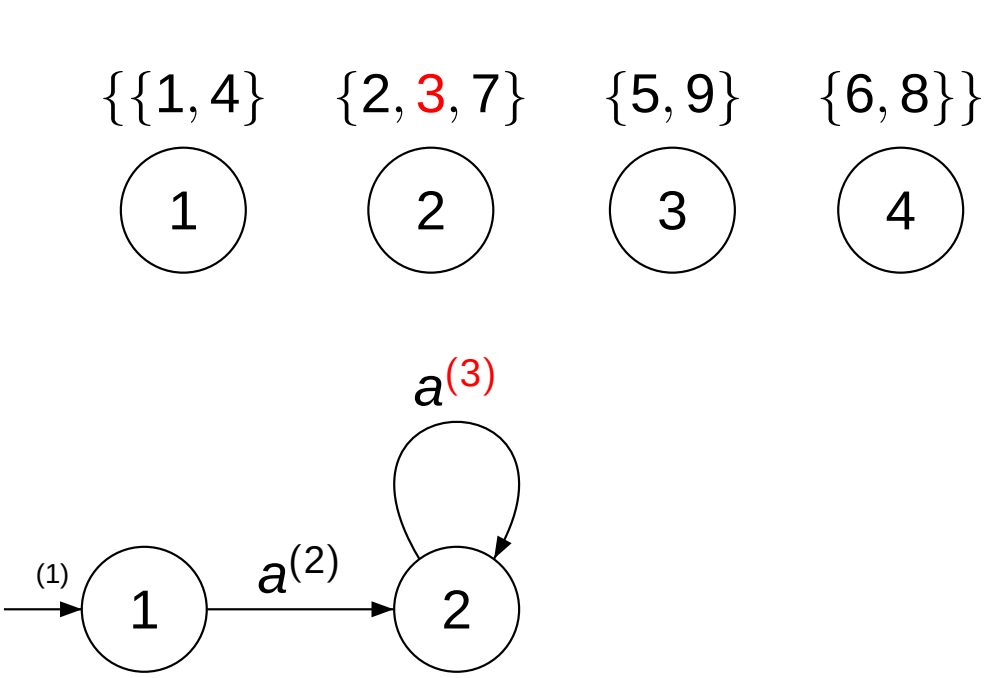
Reformulation de l'algorithme de Bassino-Nicaud

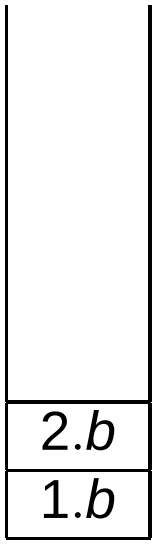




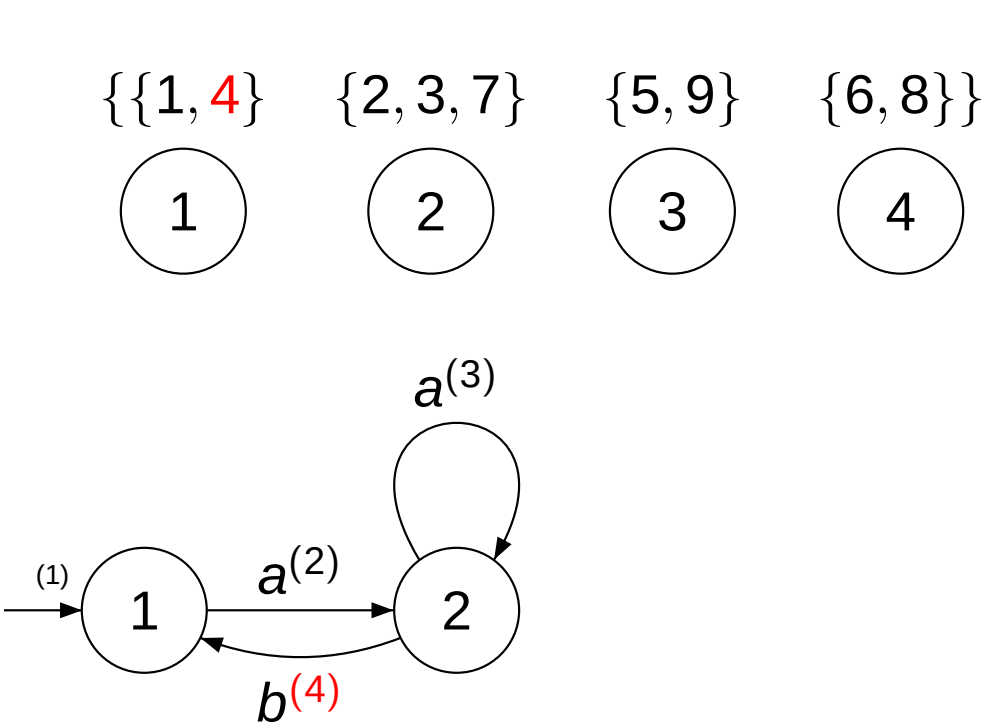


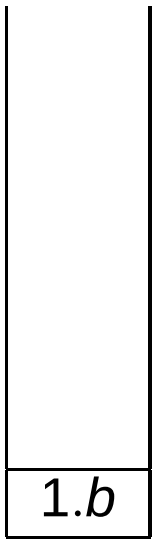
Reformulation de l'algorithme de Bassino-Nicaud





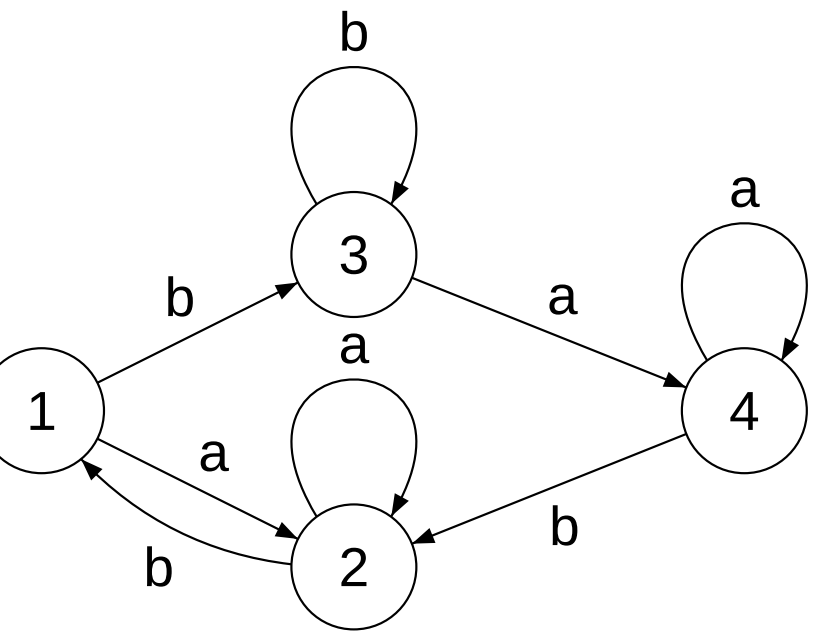
Reformulation de l'algorithme de Bassino-Nicaud





Reformulation de l'algorithme de Bassino-Nicaud

$\{\{1,4\}$ $\{2,3,7\}$ $\{5,9\}$ $\{6,8\}\}$

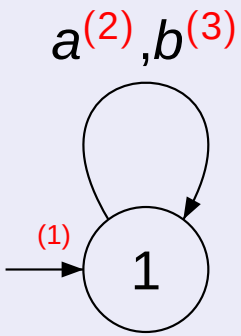


Reformulation de l'algorithme de Bassino-Nicaud

Contrainte sur les partitions d'ensembles

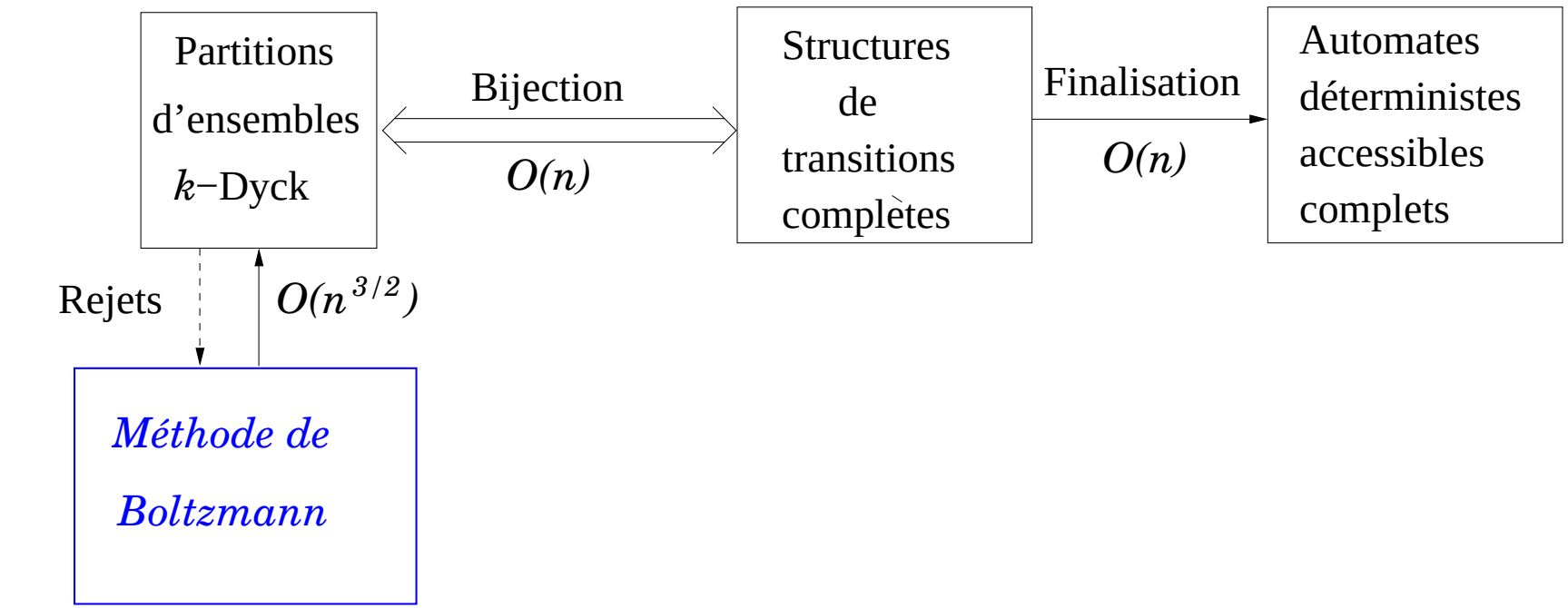
Exemple : $Alphabet = \{a, b\}$

$\{\{1, 2, 3\}...\}$



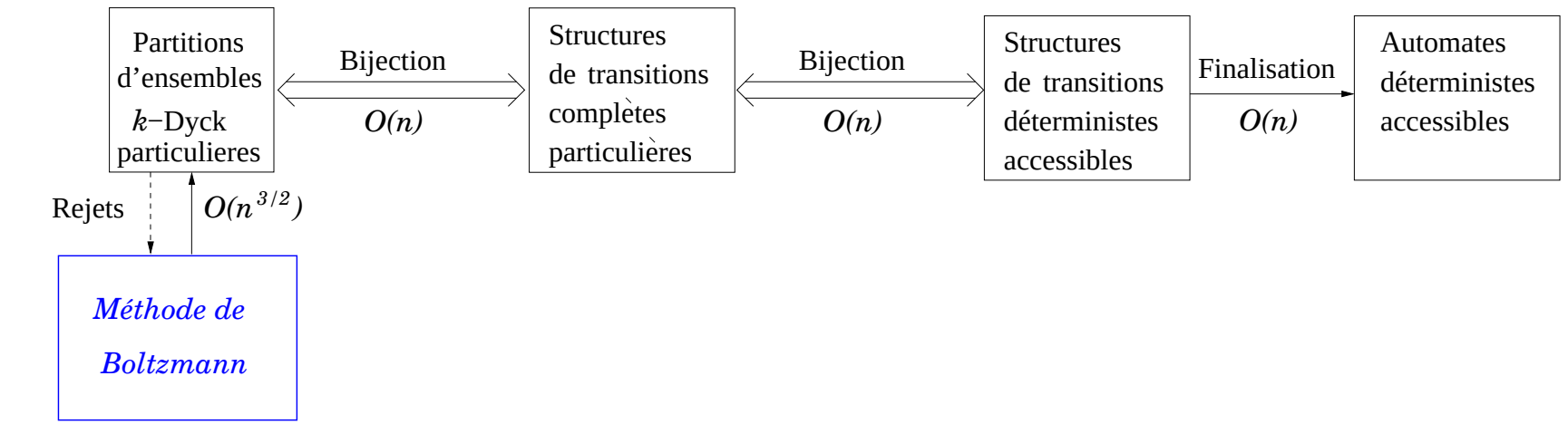
Il n'y a plus de transition disponible pour finir l'automate

Reformulation de l'algorithme de Bassino-Nicaud



Bassino−David−Nicaud[2008]

Génération aléatoire d'automates incomplets

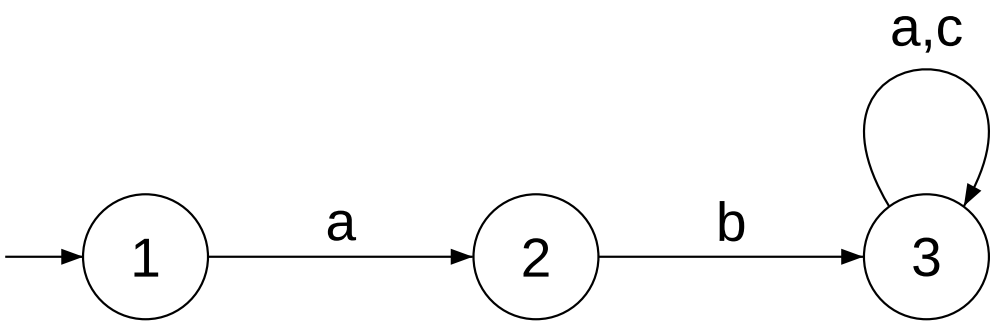


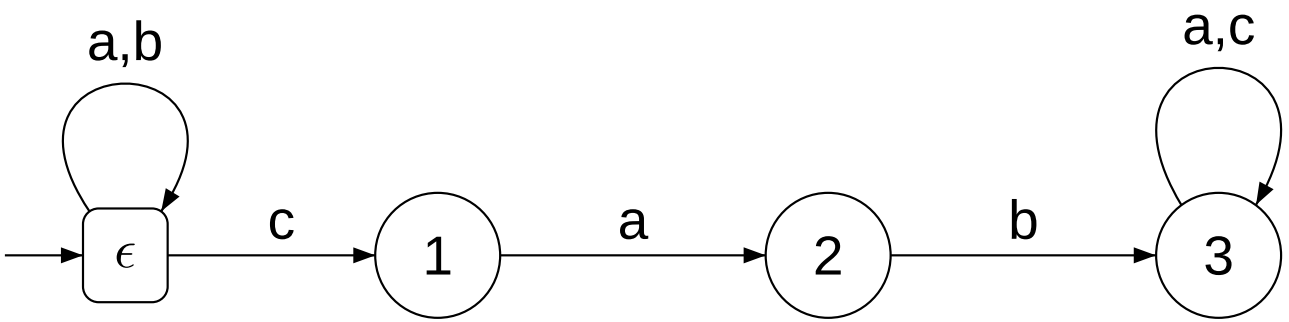
Bassino−David−Nicaud[2008]

Bijection [Bassino, David, Nicaud - PuMA'08]

Il existe une bijection entre

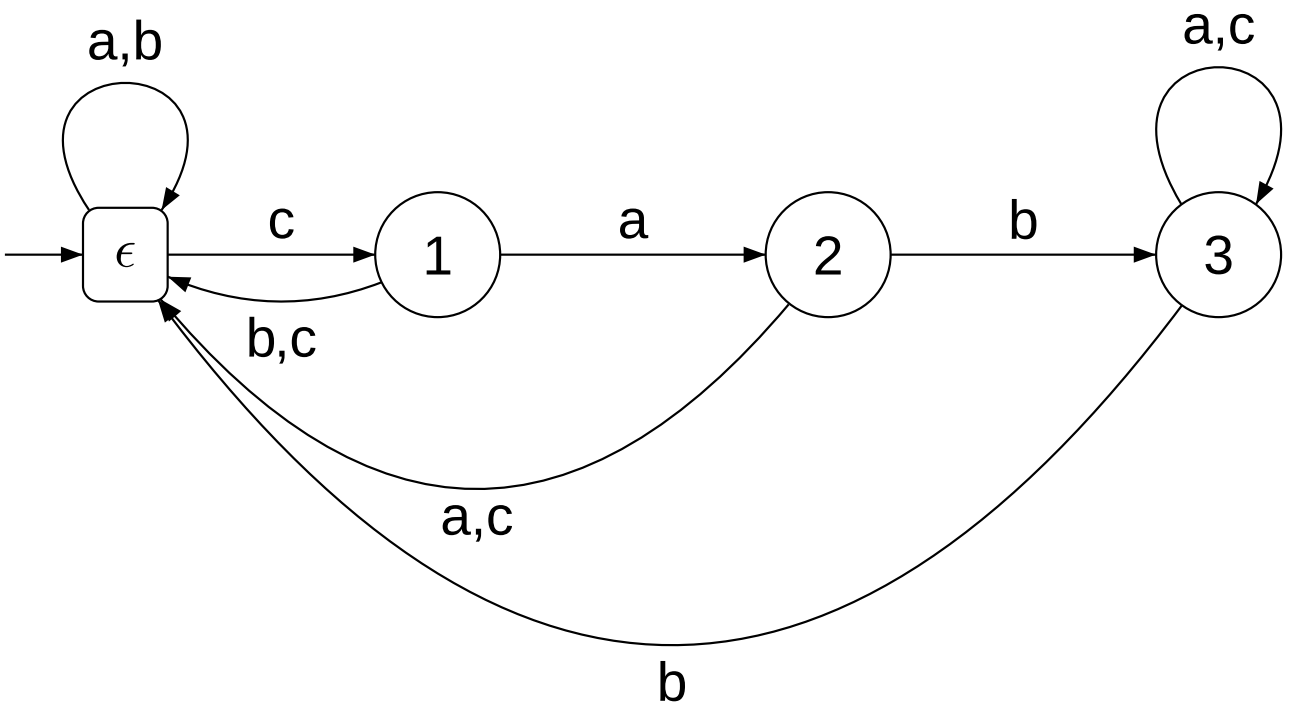
- L'ensemble des **automates déterministes accessibles** de taille n sur un alphabet à k lettres
- Un ensemble d'**automates déterministes accessibles complets particuliers** de taille $n + 1$ sur un alphabet à k lettres


a,c
1
a
2
b
3



$$\forall i \in \{1..k-1\} : \epsilon.a_i = \epsilon$$

$$\epsilon.a_k = 1$$



Le langage reconnu n'est pas préservé.

De l'automate complet aux automates possiblement incomplets

La transformation inverse se fait de la façon suivante :

- supprimer l'état ϵ
- Supprimer les transitions liées à l'état ϵ
- Rendre l'état 1 de nouveau initial.

Logiciels associés

REGAL [Bassino, David, Nicaud - CIAA'07]

Random and **E**xhaustive **G**enerators for **A**utomata - **L**ibrary

Bibliothèque en C++ contenant les générateurs uniformes suivants :

- Automates déterministes (accessibles) (complets),
- Automates fortement connexes,
- Automates minimaux,
- Automates émondés,
- Transducteurs (même propriétés),
- Tries.

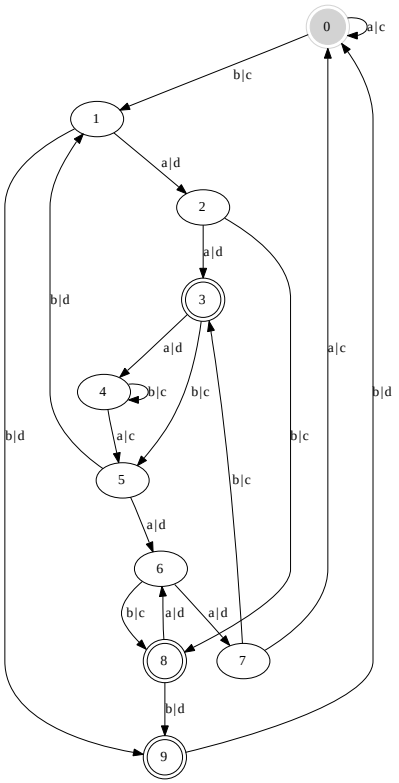
Téléchargeable à `http ://regal.univ-mlv.fr`

Logiciels associés

PREGA

Program to **R**andomly and **E**xhaustivly **G**enerate **A**utomata

- L'utilisateur décrit ce qu'il souhaite engendrer dans un fichier `XML`.
- Permet d'obtenir des automates sous différents formats : `XML`, `GasteX`, `DoT`...



REGAL

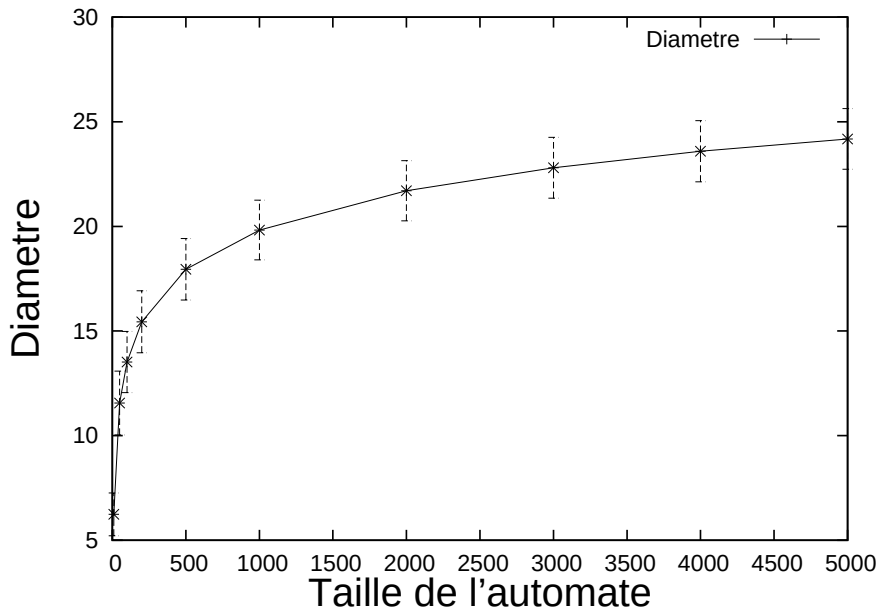
La complexité moyenne de la génération aléatoire est $\mathcal{O}(n\sqrt{n})$.

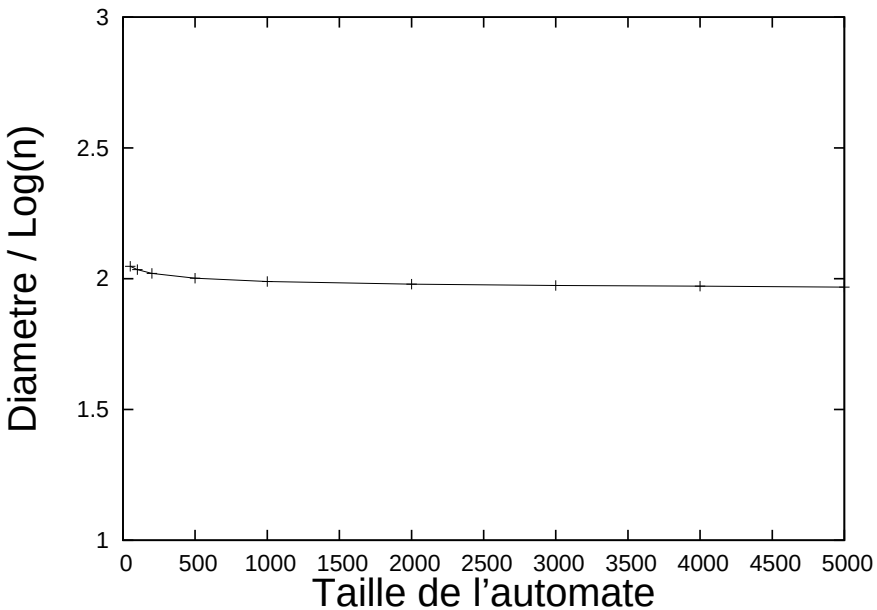
Performances

Taille de l'automate	10	100	1000	10000	100000
Temps moyen	0.1*ms*	2*ms*	50*ms*	1.7*s*	1*min* 30*s*

Les automates engendrés sont définis sur l'alphabet $\{a, b\}$. Tests réalisés sur un *Intel Dual Core 1.73Ghz, 2Go RAM*.

Diamètre moyen





Conjecture

Soit un entier $k \geq 2$. Pour la distribution uniforme sur l'ensemble des automates déterministes accessibles complets à n états sur un alphabet à k lettres, le diamètre moyen d'un automate est $\Theta(\log n)$.

Proportions d'automates non-minimaux

Taille	10	100	500	1 000	5 000	10 000
$k = 2$ (%)	18,67	14,94	14.68	14.91	14.68	14.76
$k = 3$ (%)	03.08	0.22	0.01	00.02	0.01	0.01
$k = 4$ (%)	0.47	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01

Conjecture

Soit un entier $k \geq 2$. Pour la distribution uniforme sur les automates déterministes accessibles complets à n états sur un alphabet à k lettres, il existe une proportion constante d'automates minimaux.

Algorithmes de minimisation

Données

La plupart des algorithmes de minimisation calculent l'automate minimal à partir d'un automate déterministe. Tout automate déterministe peut-être émondé et complété en temps linéaire. On s'intéresse donc principalement aux automates déterministes accessibles complets.

Complexité en temps dans le pire des cas

- Moore (1956) : $\mathcal{O}(n^2)$
- Hopcroft (1971) : $\mathcal{O}(n \log n)$

Partition de Myhill-Nerode

$p \sim q \iff$ les ensembles de mots reconnus en prenant p et q comme états initiaux sont égaux.

$p \sim_i q \iff$ les ensembles de mots de longueur $\leq i$ reconnus en prenant p et q comme états initiaux sont égaux.

On s'intéresse donc aux mots de longueurs croissantes.

$\sim_i$ induit une partition de l'ensemble des états.

Algorithme de Moore

- $\sim_0$: on partitionne Q en deux sous-ensembles (les états terminaux et les états non-terminaux).
- On calcule $\sim_i$ en raffinant $\sim_{i-1}$.
- On répète l'opération jusqu'à ce que $\sim_i = \sim_{i-1}$.

Il y a au plus $n - 1$ raffinements de la partition.

Algorithme de Moore

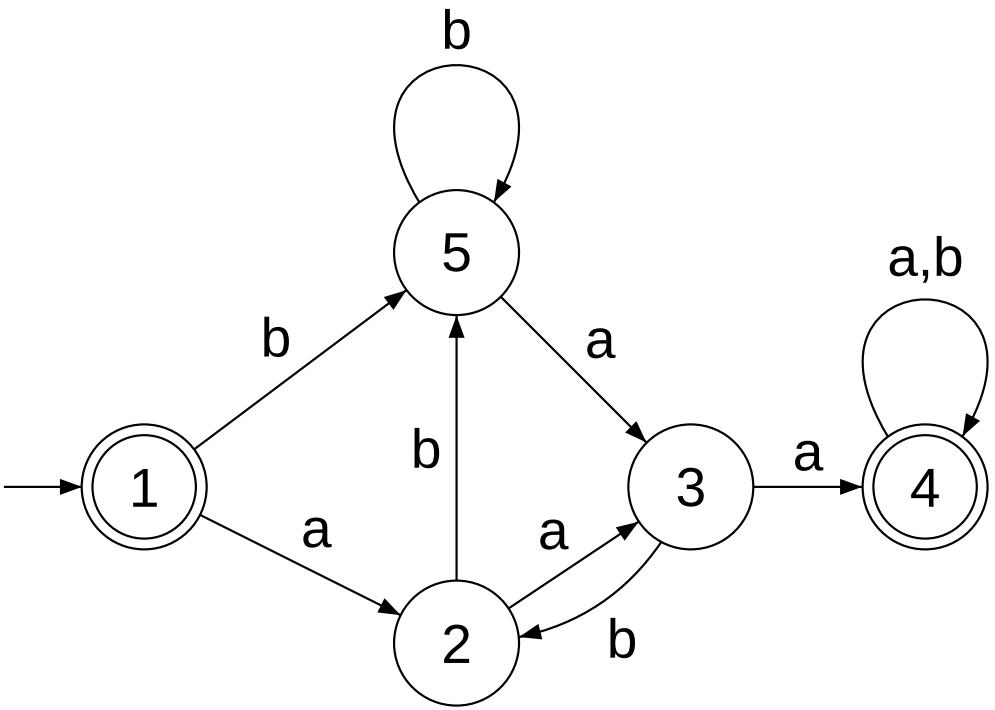


FIG.: Automate non-minimal

	$\sim_0$
1	1
2	0
3	0
4	1
5	0

$$\{2,3,5\}\{1,4\}$$

FIG.: Algorithme de Moore

Algorithme de Moore

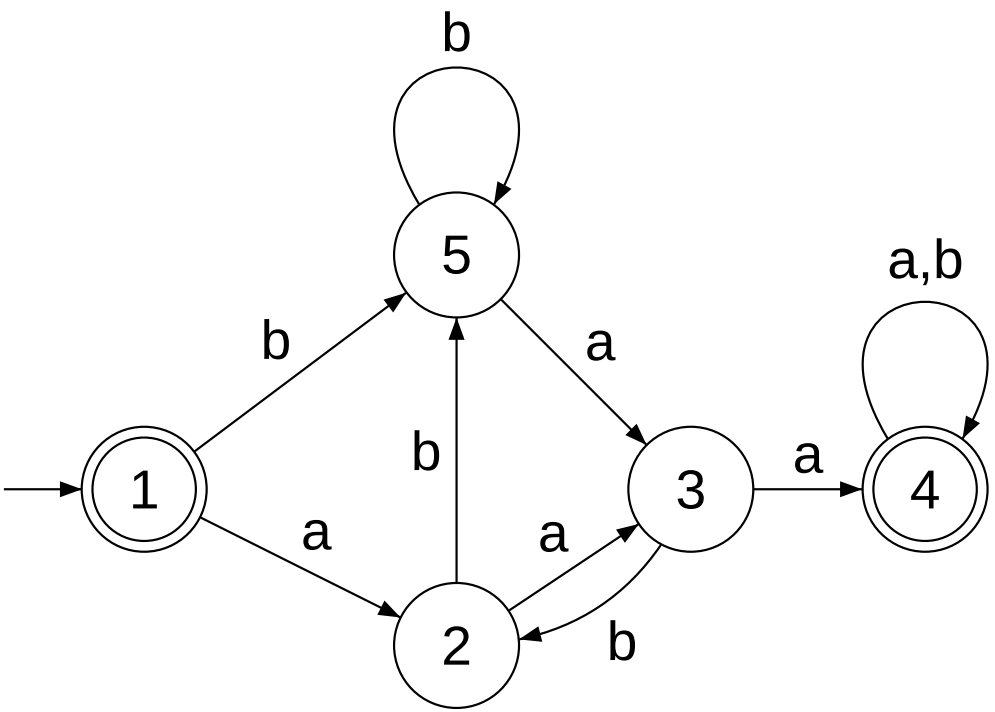


FIG.: Automate non-minimal

$$p \sim_1 q \iff \begin{cases} p \sim_0 q \\ \forall a \in A,\ p \cdot a \sim_0 q \cdot a \end{cases}$$

	$\sim_0$	a	b
1	1	0	0
2	0	0	0
3	0	1	0
4	1	1	1
5	0	0	0

$$\{2,3,5\}\{1,4\}$$

FIG.: Algorithme de Moore

Algorithme de Moore

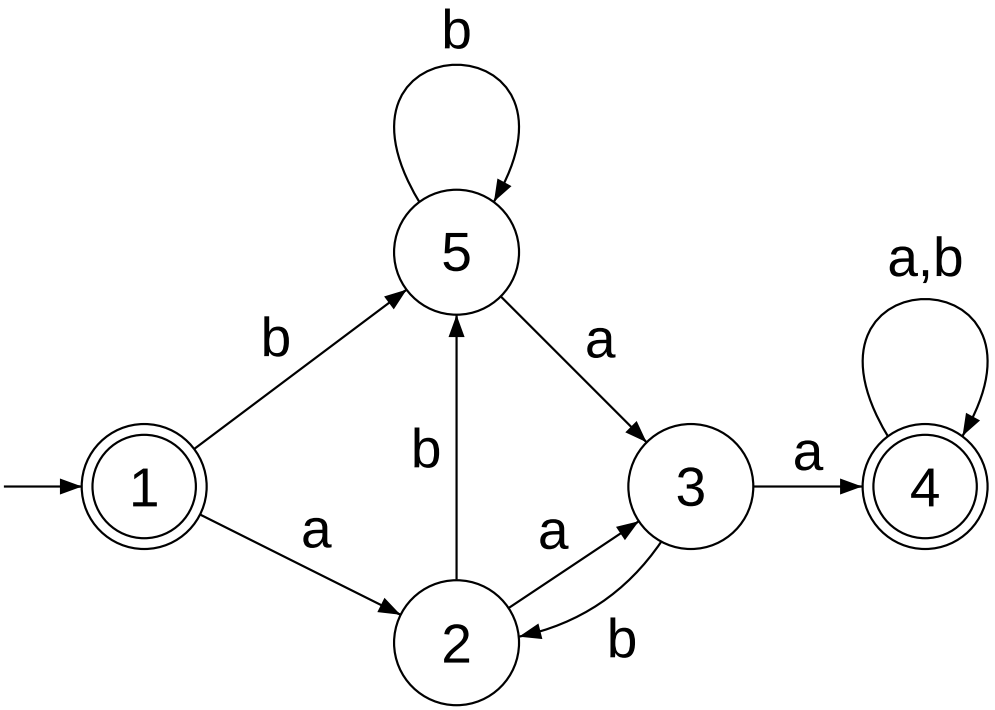


FIG.: Automate non-minimal

$$p \sim_1 q \iff \begin{cases} p \sim_0 q \\ \forall a \in A,\ p \cdot a \sim_0 q \cdot a \end{cases}$$

	$\sim_0$	a	b	$\sim_1$
1	1	0	0	0
2	0	0	0	1
3	0	1	0	2
4	1	1	1	3
5	0	0	0	1

$$\{1\}\{2,5\}\{3\}\{4\}$$

FIG.: Algorithme de Moore

Algorithme de Moore

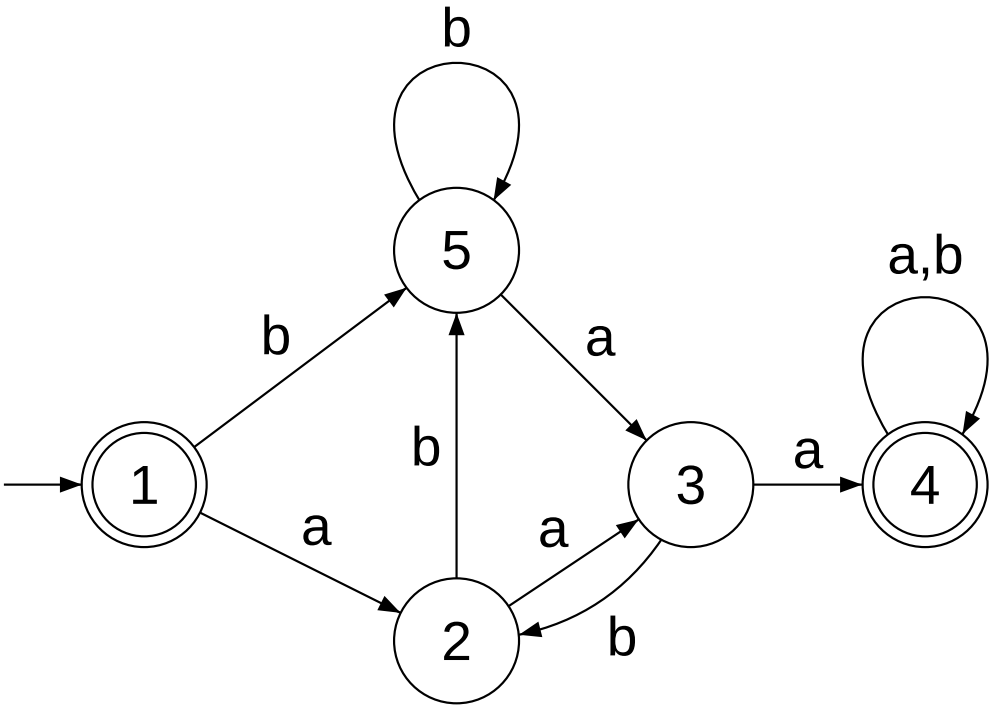


FIG.: Automate non-minimal

$$p \sim_2 q \iff \begin{cases} p \sim_1 q \\ \forall a \in A,\ p \cdot a \sim_1 q \cdot a \end{cases}$$

	$\sim_0$	a	b	$\sim_1$	a	b	$\sim_2$
1	1	0	0	0	1	1	0
2	0	0	0	1	2	1	1
3	0	1	0	2	3	1	2
4	1	1	1	3	3	3	3
5	0	0	0	1	2	1	1

$$\{1\}\{2,5\}\{3\}\{4\}$$

FIG.: Algorithme de Moore

Algorithme de Moore

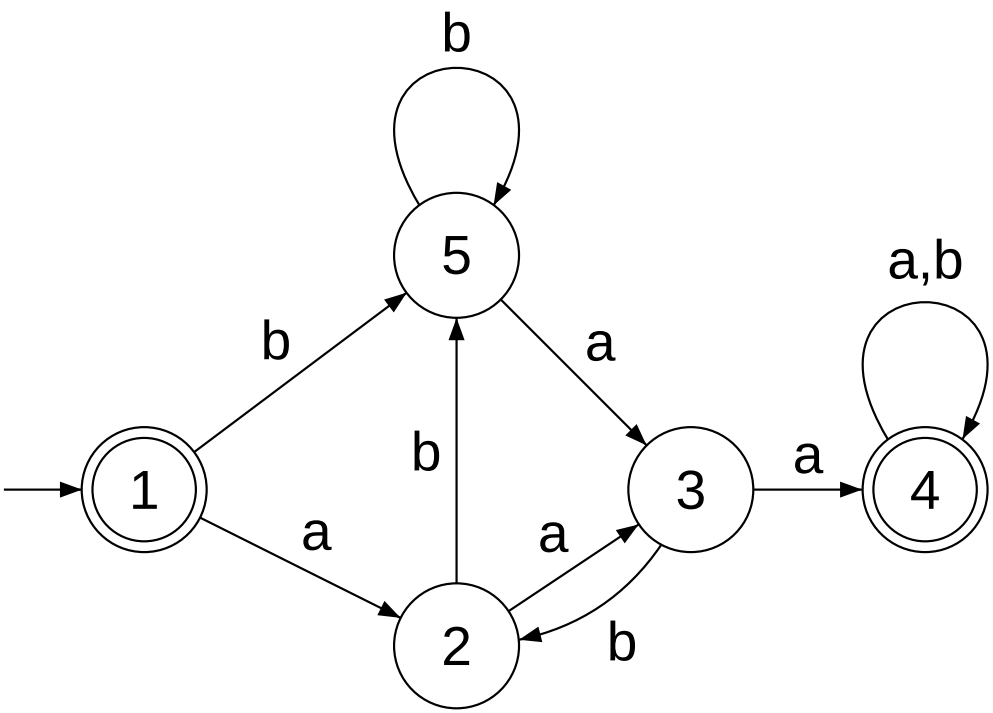


FIG.: Automate non-minimal

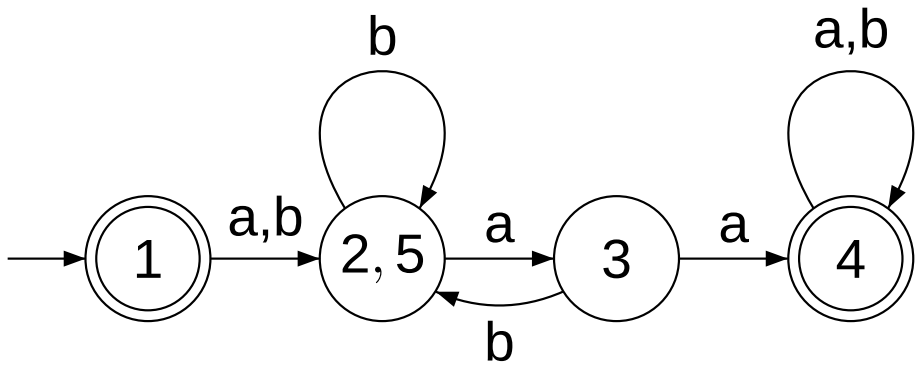


FIG.: Automate minimal

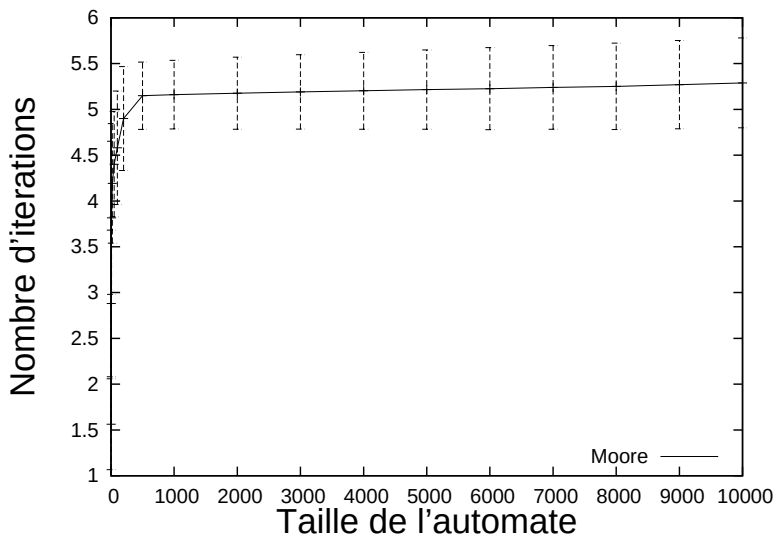
Calcul de la complexité

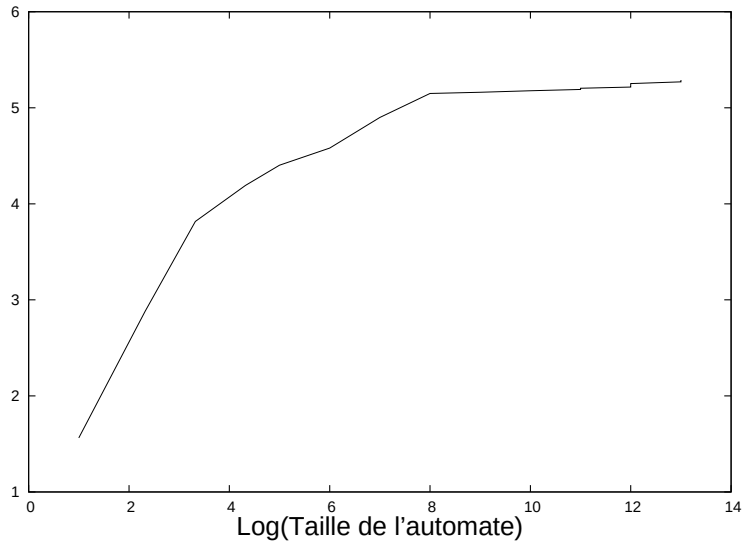
La complexité en temps dépend :

- du nombre de raffinements de partition fait par l'algorithme.
- Le coût d'un raffinement de partition.
  $\rightarrow \mathcal{O}(n)$ en utilisant un tri lexicographique.

Quel est le nombre moyen de raffinements de la partition ?

Résultat expérimental





Méthodologie

But

Montrer qu'il existe peu d'automates pour lequel l'algorithme s'exécute lentement.

Méthode

1. Caractériser l'ensemble des automates minimisés en plus de ℓ raffinements de partitions
2. Majorer le cardinal de cet ensemble.

Caractérisation des automates minimisés en au moins ℓ itérations

Il existe deux états p et q, séparés lors du ℓ-ème raffinement de la partition :

$$\iff \begin{cases} p \sim_{\ell-1} q : & \text{pour tout mot } w \text{ de longueur inférieure à } \ell, \\ & p \cdot w \in F \iff q \cdot w \in F \\ p \nsim_{\ell} q : & \text{il existe un mot } u \text{ de longueur } \ell \text{ tel que,} \\ & p \cdot u \in F \iff q \cdot u \notin F \end{cases}$$

$\rightarrow$ contraintes sur l'ensemble des états terminaux.

Premier résultat

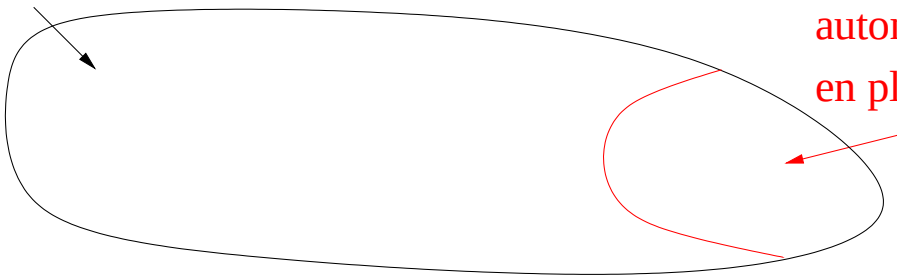
Théorème (Bassino, David, Nicaud - STACS'09)

Pour toute distribution sur l'ensemble des structures de transitions déterministes à n états sur un alphabet à k lettres, et la distribution de Bernoulli sur l'ensemble des états terminaux, la complexité moyenne de l'algorithme de Moore est $\mathcal{O}(n \log n)$.

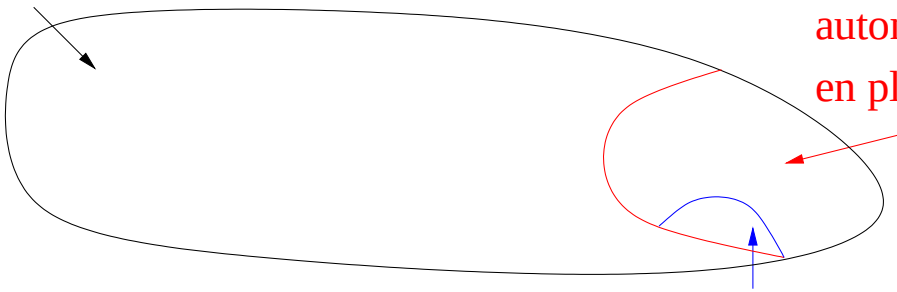
En quelques mots...

Le nombre moyen de raffinements de partitions est $\mathcal{O}(\log n)$.

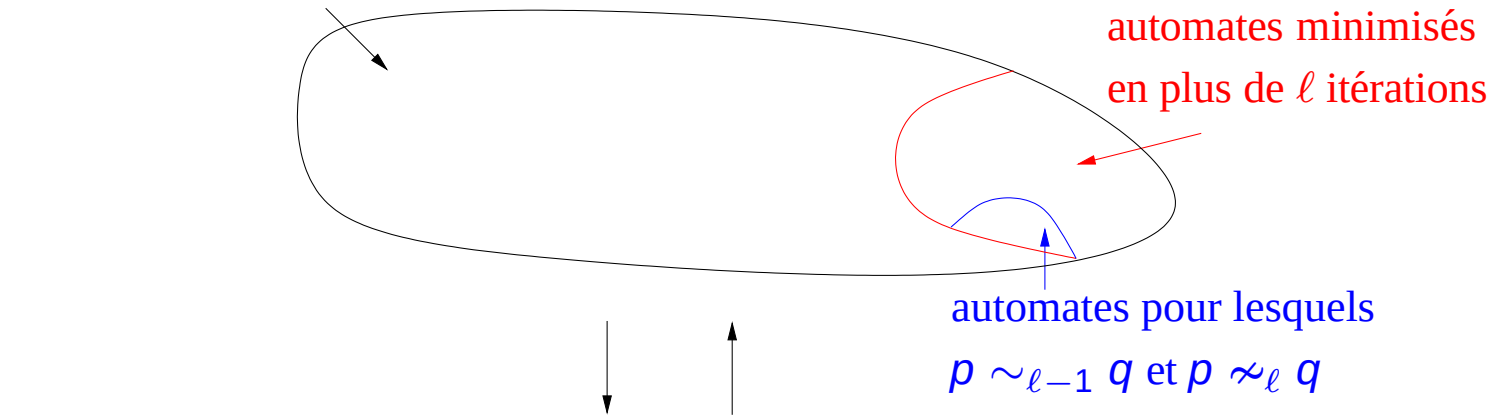
Ensemble d'automates déterministes de taille n

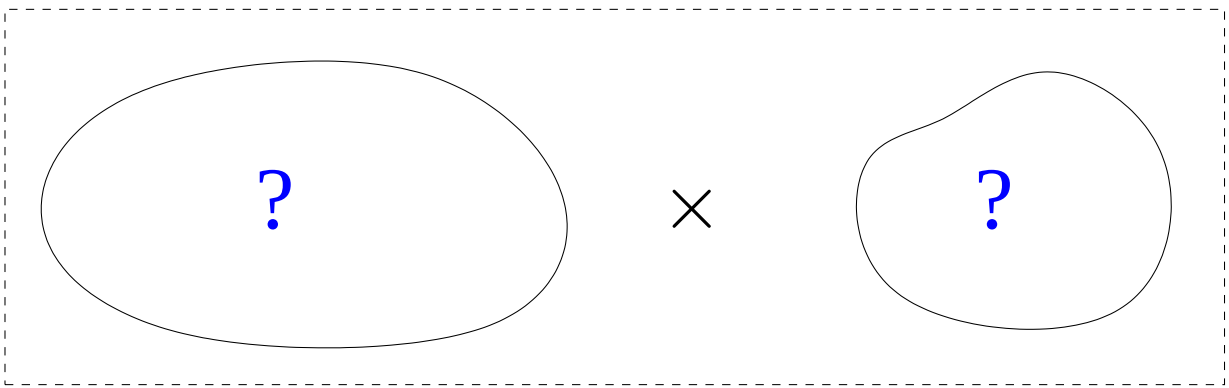


Ensemble d'automates déterministes de taille n



Ensemble d'automates déterministes de taille n



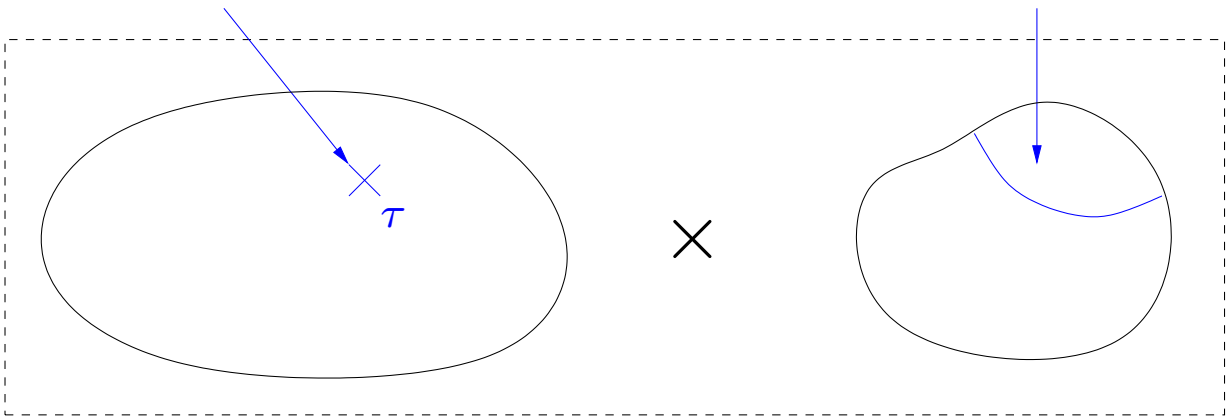


Ensemble de structures de transitions déterministes

$\mathcal{P}(Q)$

Il existe au plus $2^{n-\ell}$ ensembles F tels que pour l'automate (τ, F), $p \sim_{\ell-1} q$ et $p \not\sim_{\ell} q$

Pour toute structure de transitions τ fixée



Ensemble de structures de transitions déterministes

$\mathcal{P}(Q)$

Idée de la preuve

Pour les paramètres τ, p, q , ℓ fixés, on veut majorer l'ensemble des ensembles d'états terminaux F pour lesquels $p \sim_{\ell-1} q$ et $p \nsim_{\ell} q$.

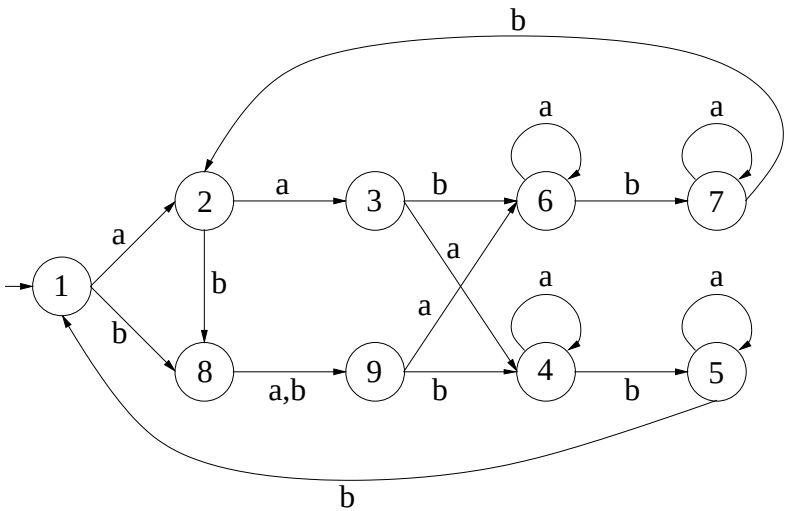
On modélise les contraintes de cet ensemble à l'aide d'un graphe de dépendance.

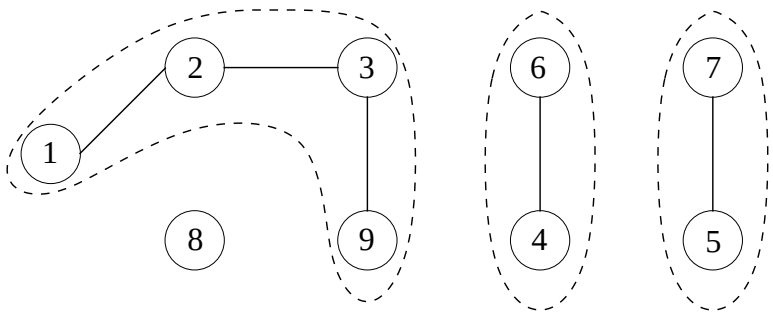
- ses sommets sont les états de la structure de transitions,
- il existe une arête $(p \cdot w, q \cdot w)$ ssi w est un mot de longueur inférieure à ℓ et $p \cdot w \in F \iff q \cdot w \in F$.

On regarde un sous-graphe associé à un mot u de longueur ℓ.

Graphe de dépendance

$p = 3$	$q = 9$	$u = abbaa$



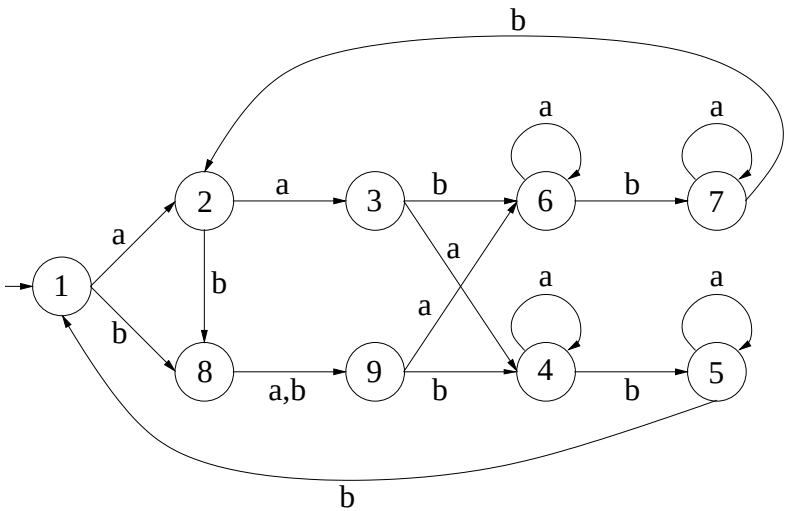


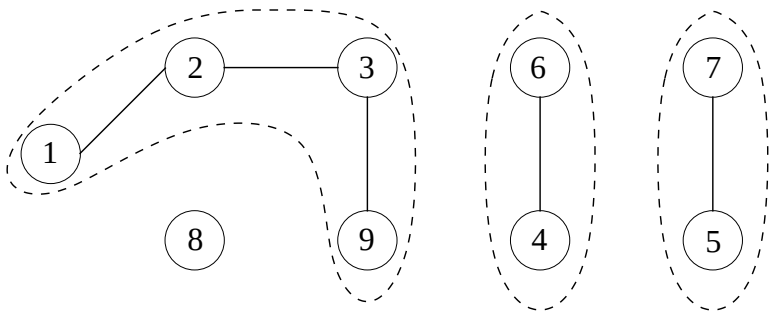
Lemme

Pour tout paramètre τ, p, q , ℓ fixés, le graphe de dépendance contient au plus $n - \ell$ composantes connexes.

Graphe de dépendance

$p = 3$	$q = 9$	$u = abbaa$





Lemme

Pour tout paramètre τ, p, q , ℓ fixés, le graphe de dépendance contient au plus $n - \ell$ composantes connexes.

Premier résultat

Fonctionne pour la distribution uniforme sur les automates déterministes (accessibles) complets.

Ne fonctionne pas pour une distribution où le nombre d'états terminaux est fixé.

Optimal pour la distribution uniforme sur les automates unaires.

Premier résultat

Fonctionne pour la distribution uniforme sur les automates déterministes (accessibles) complets.

Ne fonctionne pas pour une distribution où le nombre d'états terminaux est fixé.

Optimal pour la distribution uniforme sur les automates unaires.

Premier résultat

Fonctionne pour la distribution uniforme sur les automates déterministes (accessibles) complets.

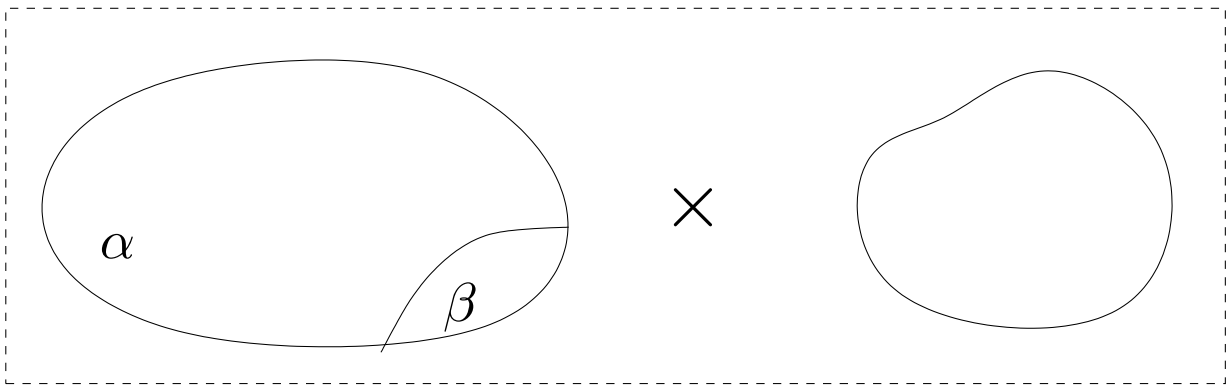
Ne fonctionne pas pour une distribution où le nombre d'états terminaux est fixé.

Optimal pour la distribution uniforme sur les automates unaires.

Second résultat

Théorème (David - MFCS'10)

Pour la distribution uniforme sur l'ensemble des automates déterministes complets à n états sur un alphabet à k lettres, avec $k \geq 2$, la complexité moyenne de l'algorithme de Moore est $\mathcal{O}(n \log \log n)$.

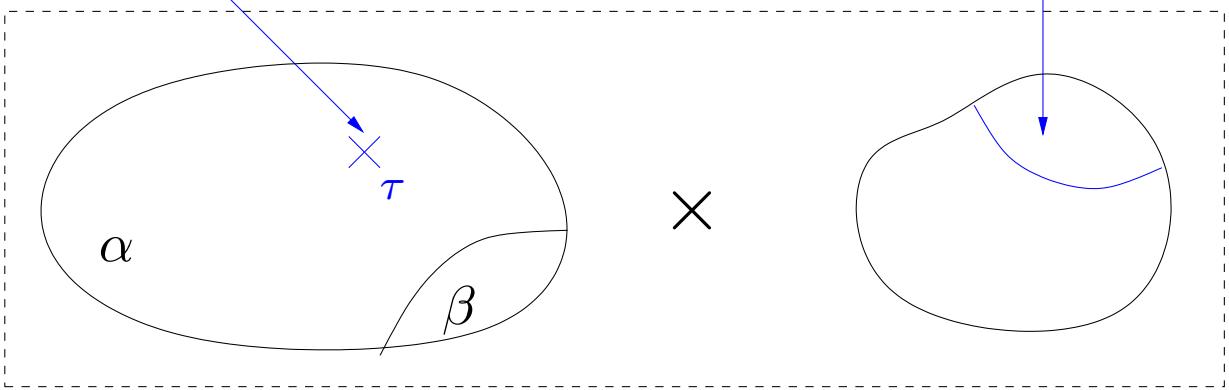


Ensemble des structures de transitions déterministes complètes

$\mathcal{P}(Q)$

Pour toute structure de transitions τ fixée telle que $\tau \in \alpha$

Il existe au plus 2^{n-k^ℓ} d'ensembles F tels que pour tout automate (τ, F) $p \sim_{\ell-1} q$ et $p \not\sim_{\ell-1} q$



Ensemble des structures de transitions déterministes complètes

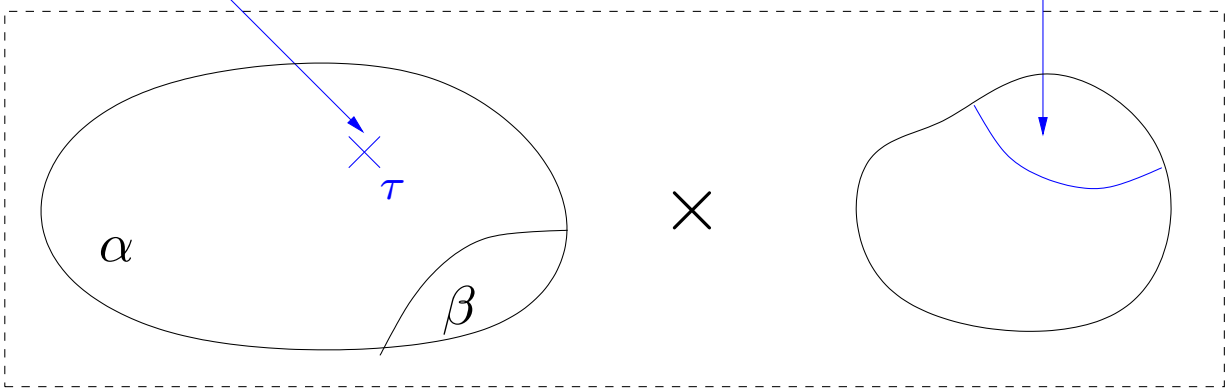
$\mathcal{P}(Q)$

L'idée

Le graphe de dépendance contient au plus $n - k^\ell$ composantes connexes.

Pour toute structure de transitions τ fixée telle que $\tau \in \alpha$

Il existe au plus 2^{n-k^ℓ} d'ensembles F tels que pour tout automate (τ, F) $p \sim_{\ell-1} q$ et $p \not\sim_{\ell-1} q$



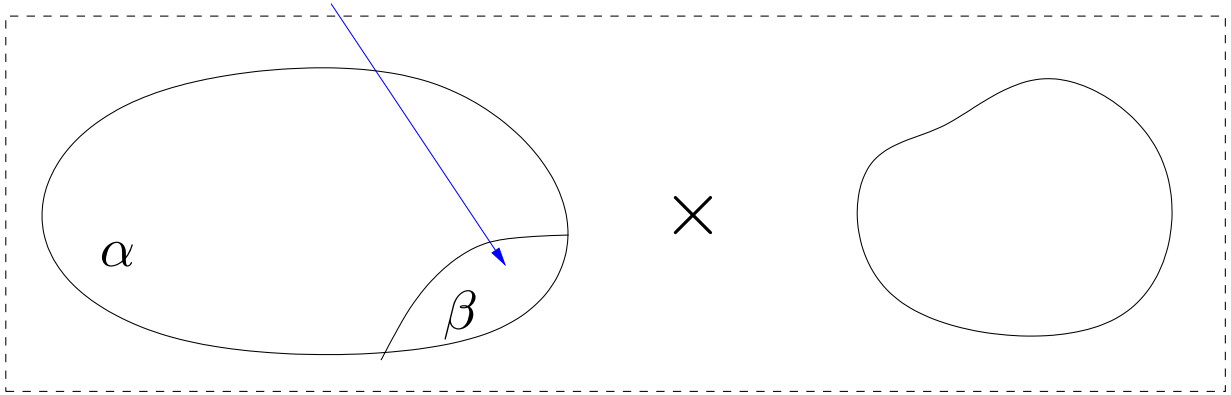
Ensemble des structures de transitions déterministes complètes

$\mathcal{P}(Q)$

L'idée

Le graphe de dépendance contient au plus $n - k^\ell$ composantes connexes.

Il existe très peu de
structures de transitions dans β



Ensemble des structures de
transitions déterministes complètes

$\mathcal{P}(Q)$

Algorithme de Hopcroft

Théorème (David 2010)

Il existe une famille d'implantations de l'algorithme d'Hopcroft dont la complexité moyenne est $\mathcal{O}(n \log \log n)$, pour la distribution uniforme sur l'ensemble des automates déterministes complets à n états sur un alphabet à k lettres, avec $k \geq 2$.

En quelques mots...

Cette famille d'implantations est toujours aussi rapide (et parfois plus rapide) que l'algorithme de Moore.

Problèmes ouverts

Contributions

- Générateurs aléatoires d'automates
- Logiciels de générations aléatoire d'automates.
- Complexité moyenne des algorithmes de Moore et Hopcroft.

Perspectives

- Le cas où il existe un nombre fixé d'états terminaux.
- Le cas des automates accessibles.
- Énumération des automates minimaux.

Perspectives dans un futur plus proche

- Étude d'algorithmes d'énumération, réductions. . .

Publications

Génération aléatoire

Algorithmes

- F. Bassino, J. David, C. Nicaud. *Enumeration and random generation of possibly incomplete deterministic automata*. Dans *Pure Mathematics and Applications*, volume 19 (2-3), pages 1–16, 2008.
- F. Bassino, J. David, C. Nicaud. *Random generation of possibly incomplete deterministic automata*. Dans *Génération Aléatoire de Structures COMbinatoires (Gascom'08)*, pages 31–40, 2008.

Logiciels

- F. Bassino, J. David, C. Nicaud. *REGAL : a library to randomly and exhaustively generate automata*. Présenté à CIAA'07. Prague, Republique Tchèque. Juillet 2007.

Analyse d'algorithmes de minimisation

- J. David. *The Average Complexity of Moore's State Minimization Algorithm is $\mathcal{O}(n \log\log n)$* In *35th International Symposium on Mathematical Foundations of Computer Science (MFCS'10), Brno, Czech Republic,* 12 pages, 2010.
- F. Bassino, J. David, C. Nicaud. *On the average complexity of Moore's state minimization algorithm*. Dans *26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009), Freiburg, Germany,* volume 3, pages 123–134, 2009.