

Inférence d'automates classifieurs

équipe : Aïda, IRISA

encadrant : Jacques NICOLAS

Bourse MESR

Plan de l'exposé

*De l'inférence d'accepteurs de séquence
à l'inférence de classifieurs de séquences*

- I. Inférence Régulière
- II. Inférence k -Régulière
 - A. Univoques
 - B. Univoques déterministes
 - C. Expérimentations

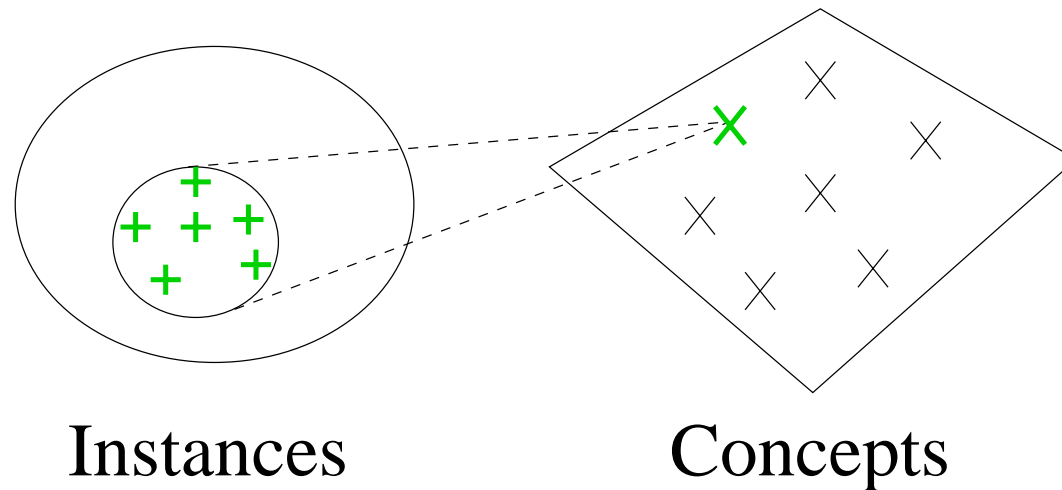
I. Inférence Régulière

[Fu, Booth, 75], [Angluin 81, 82],
[Dupont, Miclet, Vidal 94]

- ▷ Définition
- ▷ Espace de recherche
- ▷ Vers l'inférence d'automates classifieurs

Apprentissage inductif

Apprendre un concept à partir d'exemples



Choix importants :

- Espace de description des instances
- Espace des hypothèses (concepts)

Description des instances

- Approche classique (arbres de décisions, . . .) :

Description d'une instance par n attributs

$$D = \{A_1 = v_1, A_2 = v_2, \dots, A_n = v_n\}$$

- Intégrer la notion de séquentialité :

Description d'une instance par une séquence

de symboles $D = s_1 s_2 \dots$

Théorie des langages

- Séquence de symboles $s_1 s_2 \dots s_p$: **mot**
- Ensemble de mots $\{m_1, m_2, \dots\}$: **langage**
- Ensemble de règles de production des mots d'un langage : **grammaire**

*Apprentissage inductif d'un langage à partir
d'un échantillon de mots du langage :*

Inférence Grammaticale

Classification de Chomsky et apprenabilité

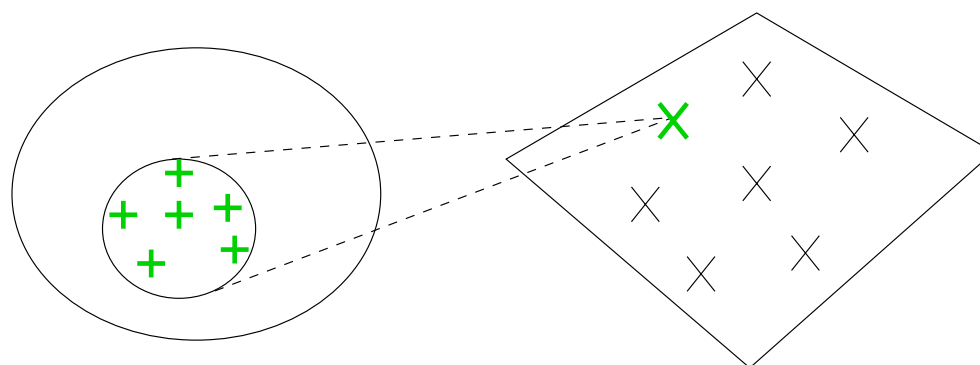
Grammaires :

- à structure de phrase (type 0)
- contextuelles (type 1)
- *algébriques* (type 2)
- **régulières** (type 3)

*Un langage régulier peut être représenté
par un automate*

Inférence régulière

Apprendre un langage régulier à partir d'un échantillon de mots du langage



Instances

Mots

Concepts

Automates



Hypothèse de complétude structurelle

I_+ est structurellement complet relativement à A
s'il existe une acceptation de I_+ par A telle que :

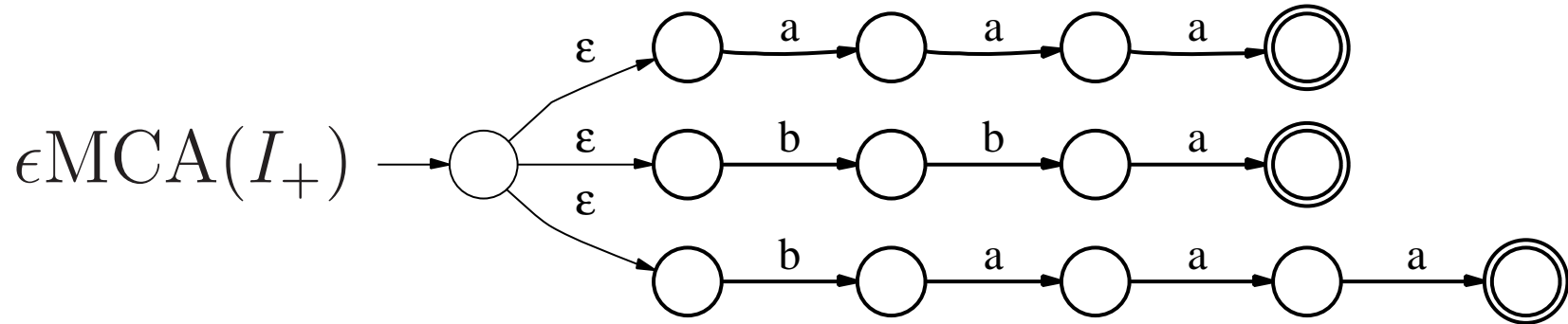
- Toute transition de A est exercée
- Tout état final de A est utilisé comme état d'acceptation

$$I_+ = \{aaa, bba, baaa\} \quad A = \begin{array}{c} \text{b} \\ \curvearrowright \\ \text{---} \bullet \xrightarrow{\text{a}} \bullet \text{---} \\ \text{a} \curvearrowleft \end{array}$$

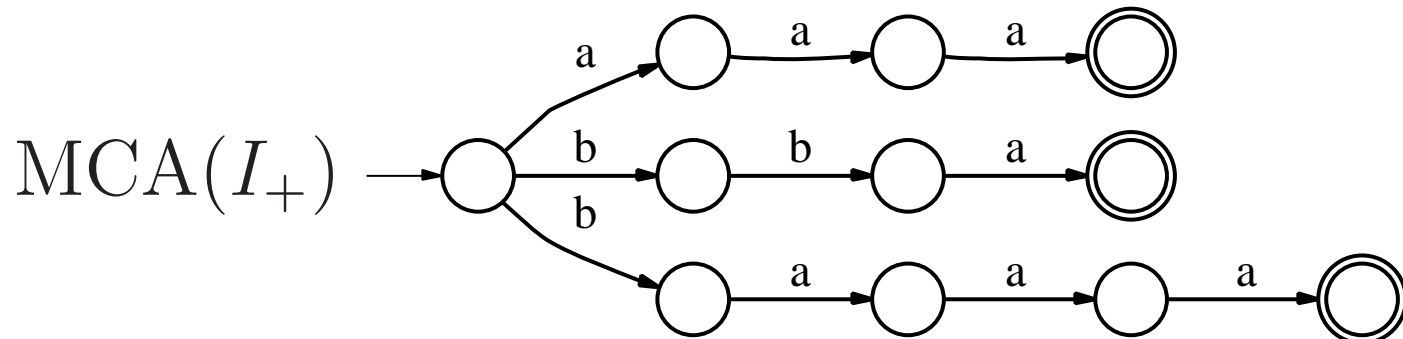
Maximal Canonical Automaton

Apprentissage par cœur de $I_+ = \{aaa, bba, baaa\}$

Union :

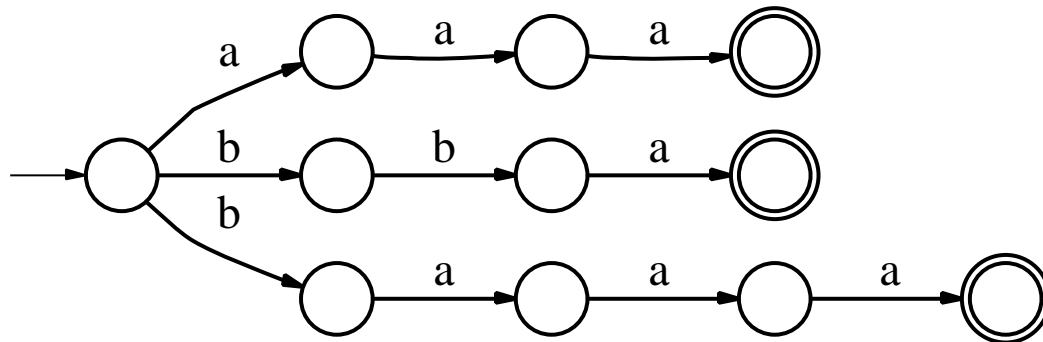


ϵ -élimination :



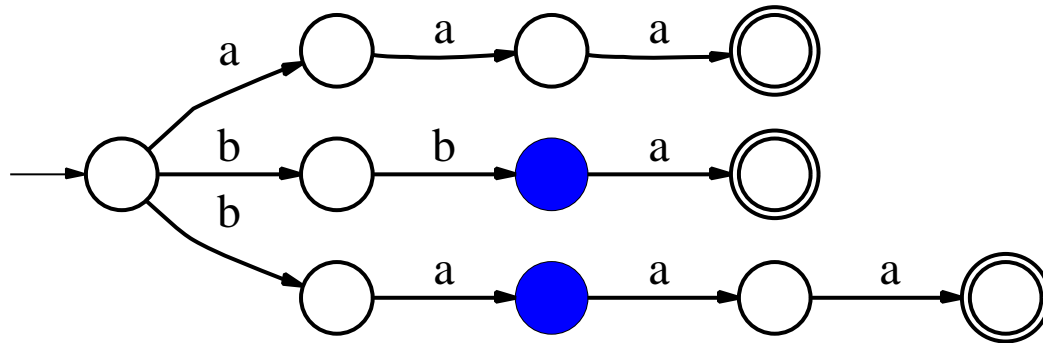
Fusion d'états

- Opération de généralisation du langage
- Conserve la complétude structurelle



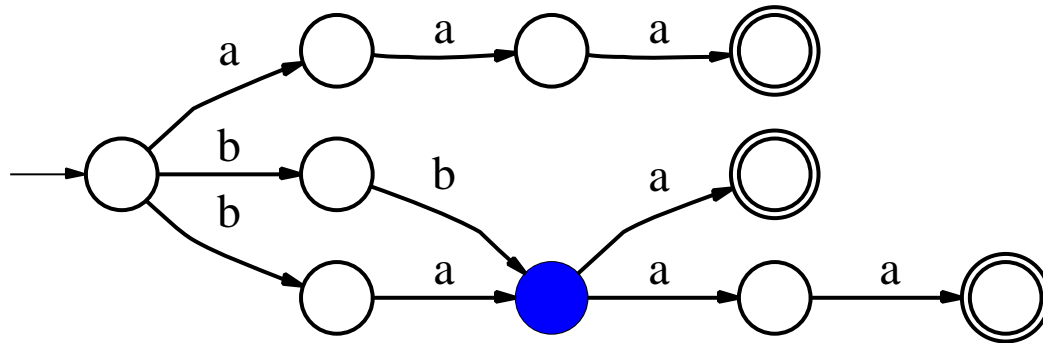
Fusion d'états

- Opération de généralisation du langage
- Conserve la complétude structurelle



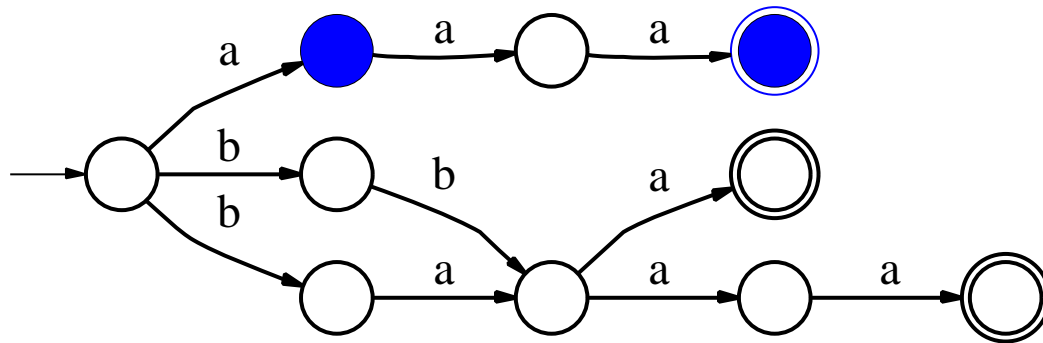
Fusion d'états

- Opération de généralisation du langage
- Conserve la complétude structurelle



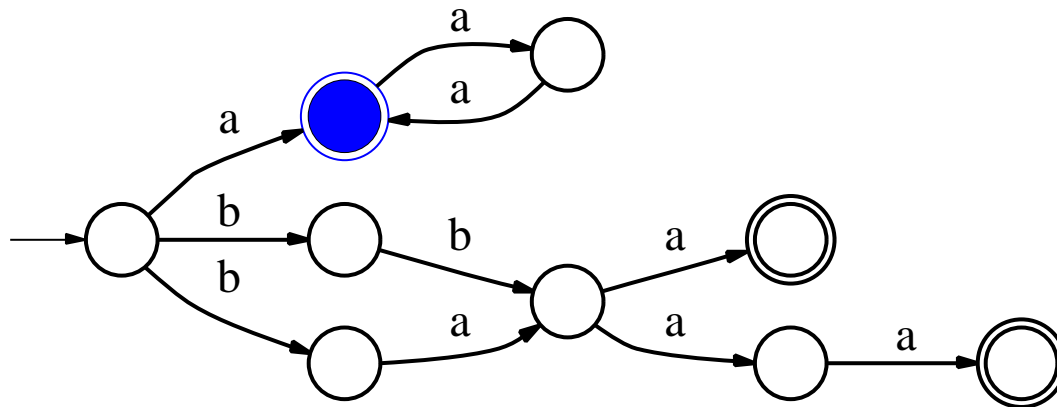
Fusion d'états

- Opération de généralisation du langage
- Conserve la complétude structurelle



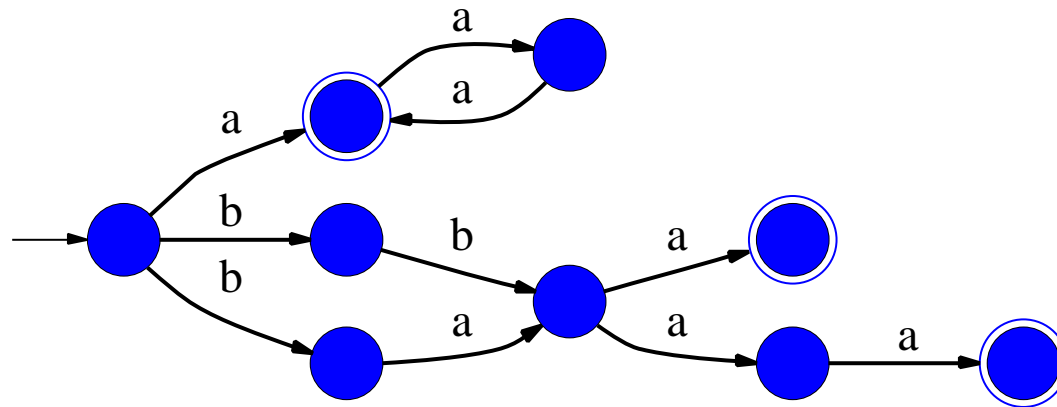
Fusion d'états

- Opération de généralisation du langage
- Conserve la complétude structurelle



Fusion d'états

- Opération de généralisation du langage
- Conserve la complétude structurelle



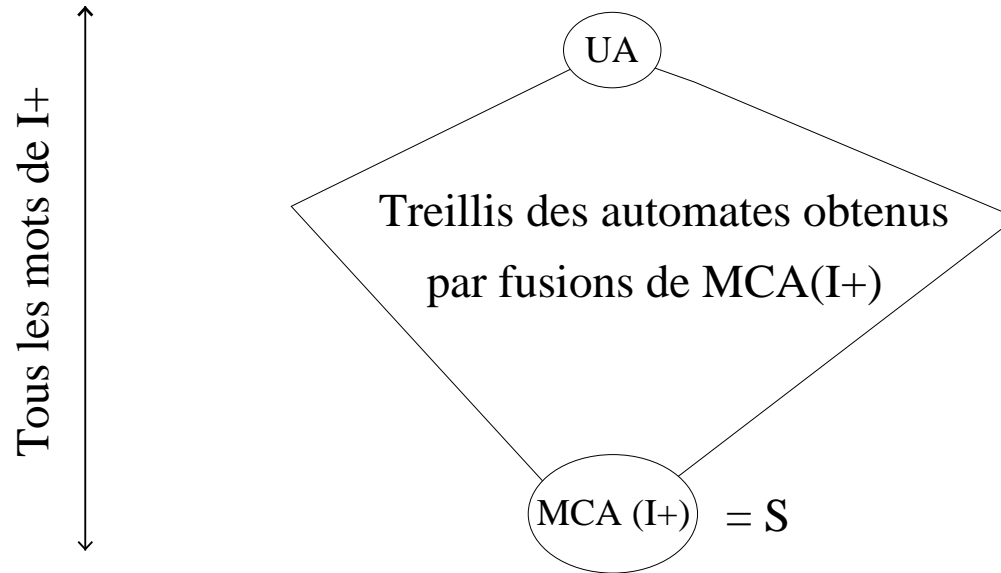
Fusion d'états

- Opération de généralisation du langage
- Conserve la complétude structurelle



Tout automate pour lequel I_+ est structurellement complet peut être obtenu par fusions d'états de $MCA(I_+)$

Espace de recherche



Identification à la limite à partir d'exemples positif seulement est impossible [Gold67]

Le problème : éviter la surgénéralisation

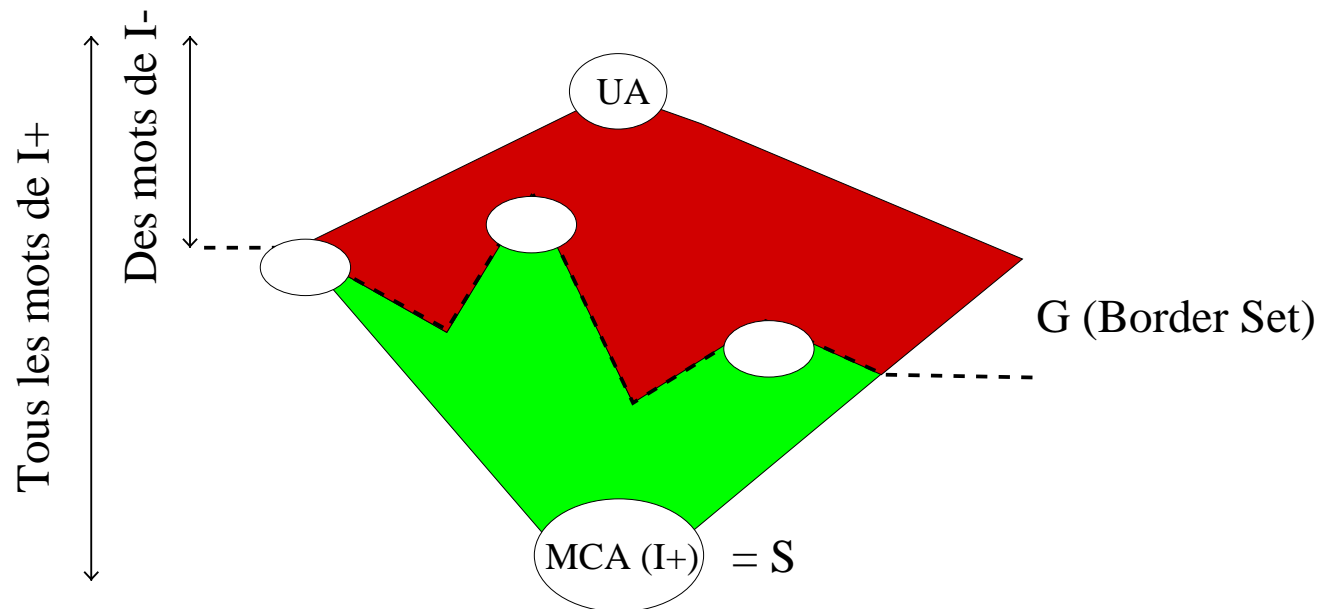
Limiter la généralisation

Ajout d'un critère supplémentaire

- Limitation à une sous classe d'automate
- Critère statistique d'arrêt des fusions
- Compatibilité avec des exemples négatifs

L'automate doit être **compatible** avec I_+ et I_-
i.e. accepter tous les mots de I_+ et aucun de I_-

Limitation de la généralisation avec un échantillon négatif I_-



Plus petit automate déterministe compatible

Problème NP-complet [Gold78]

Apprentissage “symétrique”

- Définition d’un langage régulier
⇔ Définition du langage complémentaire
- [Alquezar, Sanfeliu 95] :

Traiter symétriquement I_+ et I_-

→ apprentissage de L_+ et L_-

Classification des mots en +, - ou ?

Apprentissage de “machines de Moore, de Mealy”

[Biermann, Feldmann 72], d’automates [Lang 92,

Oncina, Garcia 92]

Un nouveau paradigme : l'inférence d'automates “classifieurs”

- Acceptation d'un mot dans L_+
→ Classification d'un mot dans L_+ , L_- ou ?
- Généralisation de I_+ limitée par I_-
→ Généralisation simultanée de I_+ et I_- sous la
contrainte d'une intersection vide des langages

*Extension de l'inférence régulière
à l'inférence k -régulière*

II. Inférence k -régulière

*Inférence **simultanée** de k langages réguliers
à partir d'échantillons de ces langages*

$$\mathcal{I} = \{I_1, \dots, I_k\}$$

↓
Inférence

↓
$$\mathcal{L} = \{L_1, \dots, L_k\}$$

Applications

Apprentissage de classifieurs de séquences

- Inférence grammaticale à partir d'exemples positifs et négatifs
- Reconnaissance de formes
- Analyse de séquences génétiques
- Sécurité informatique
- Monitoring cardiaque
- ...

Automate classifieur (k-FSA)

Machine de Moore non déterministe : $(Q, q_s, \Sigma, \Gamma, \delta, \rho)$

Q ensemble fini d'états ;

$q_s \in Q$ état initial ;

Σ alphabet d'entrée ;

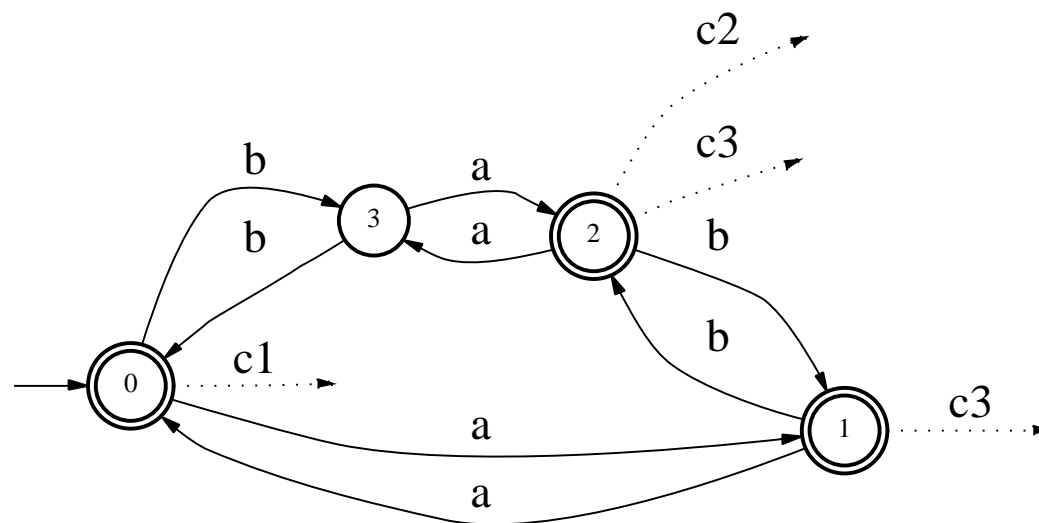
Γ alphabet de sortie ;

δ fonction de transition

$Q \times \Sigma \rightarrow \mathcal{P}(Q)$;

ρ fonction d'émission

$Q \times \Sigma \rightarrow \mathcal{P}(\Gamma)$.



Fonction de classification $\gamma : Q \times \Sigma^* \rightarrow \mathcal{P}(\Gamma)$

$$\gamma(q, m) = \bigcup_{q' \in \delta(q, m)} \rho(q')$$

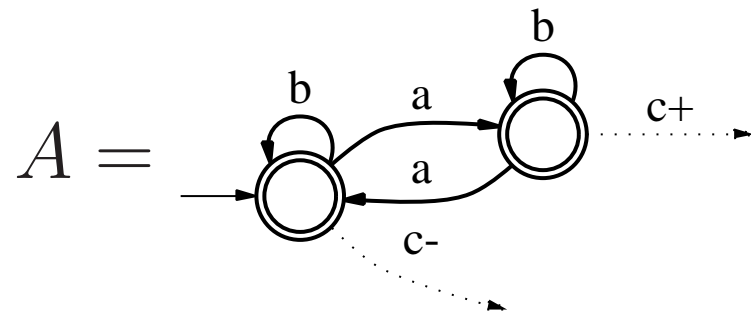
Complétude structurelle (k -FSA)

\mathcal{I} est structurellement complet relativement à A s'il existe une acceptation de \mathcal{I} par A telle que :

- Toute transition de A est exercée
- Toute émission finale de A est exercée

$$I_+ = \{aaa, bba, baaa\}$$

$$I_- = \{aaaa, baab, bbabab\}$$



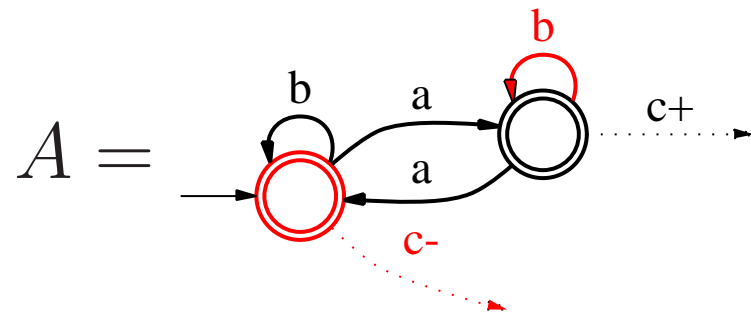
Complétude structurelle (k -FSA)

\mathcal{I} est structurellement complet relativement à A s'il existe une acceptation de \mathcal{I} par A telle que :

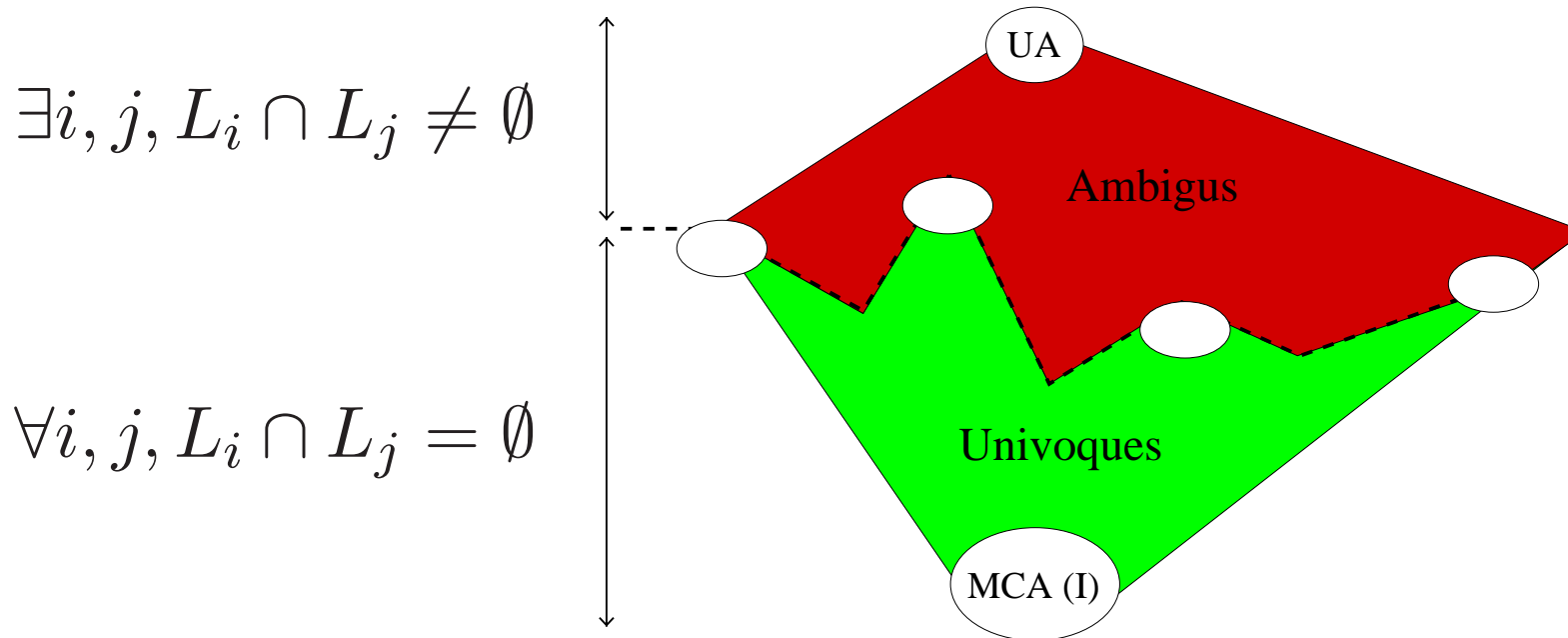
- Toute transition de A est exercée
- Toute émission finale de A est exercée

$$I_+ = \{aaa, bba, baaa\}$$

$$I_- = \{aaaa, baab, bbabab\}$$



Espace de recherche



Inférence d'automate compatible

→ inférence d'automate classifieur univoque :

classification unique pour chaque séquence

(intersection vide des langages)

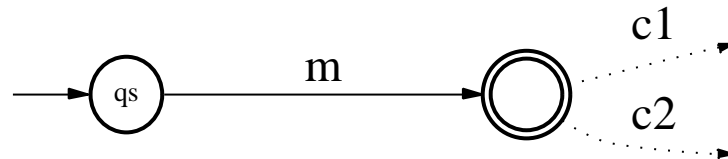
II.A. Inférence d'automates classifieurs univoques

- ▷ Ambiguïté
- ▷ Classifications incompatibles
- ▷ États incompatibles
- ▷ Fusions incompatibles

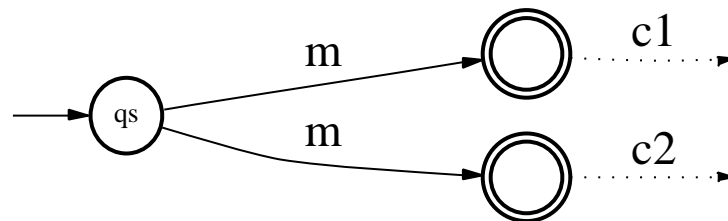
Ambiguïté

Un k -FSA est ambigu si :

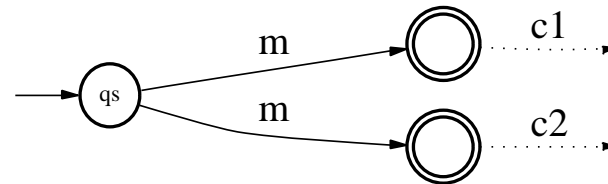
- La fonction d'émission d'un état retourne au moins deux valeurs



- Il existe deux chemins étiquetés par le même mot m permettant d'atteindre deux états d'émission définie et différente.



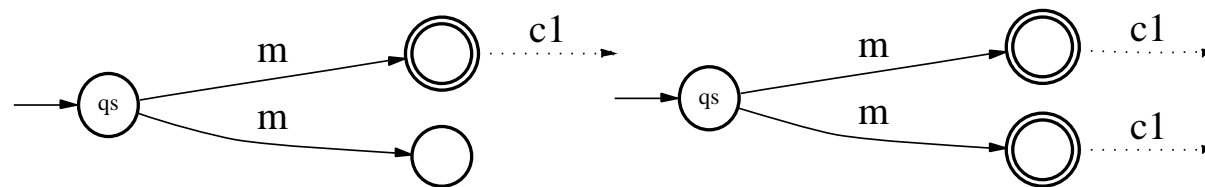
Incompatibilité



- Classifications γ_1 et γ_2 incompatibles

$$\gamma_1 \not\sim \gamma_2 \Leftrightarrow (\gamma_1 \neq \gamma_2) \wedge (\gamma_1 \neq \emptyset) \wedge (\gamma_2 \neq \emptyset)$$

- Sinon compatibles, noté \sim



- Etats q_1 et q_2 incompatibles

$$q_1 \not\sim q_2 \Leftrightarrow \exists m \in \Sigma^*, \gamma(q_1, m) \not\sim \gamma(q_2, m)$$

Compatibilité et équivalence

- Etats compatibles

$$\forall m \in \Sigma^*, \forall (s_1, s_2) \in \delta(q_1, m) \times \delta(q_2, m) : \rho(s_1) \sim \rho(s_2)$$

- Etats équivalents

(Minimisation d'automates [Aho, Ullman 72],
opposé à états distinguables)

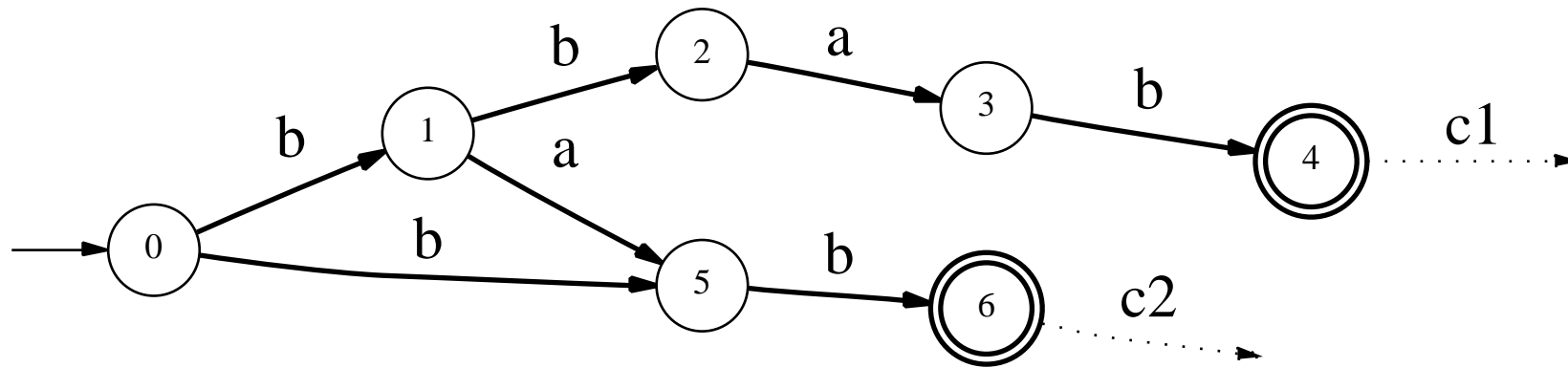
$$\forall m \in \Sigma^*, \forall (s_1, s_2) \in \delta(q_1, m) \times \delta(q_2, m) : \rho(s_1) = \rho(s_2)$$

= est transitive, mais pas \sim

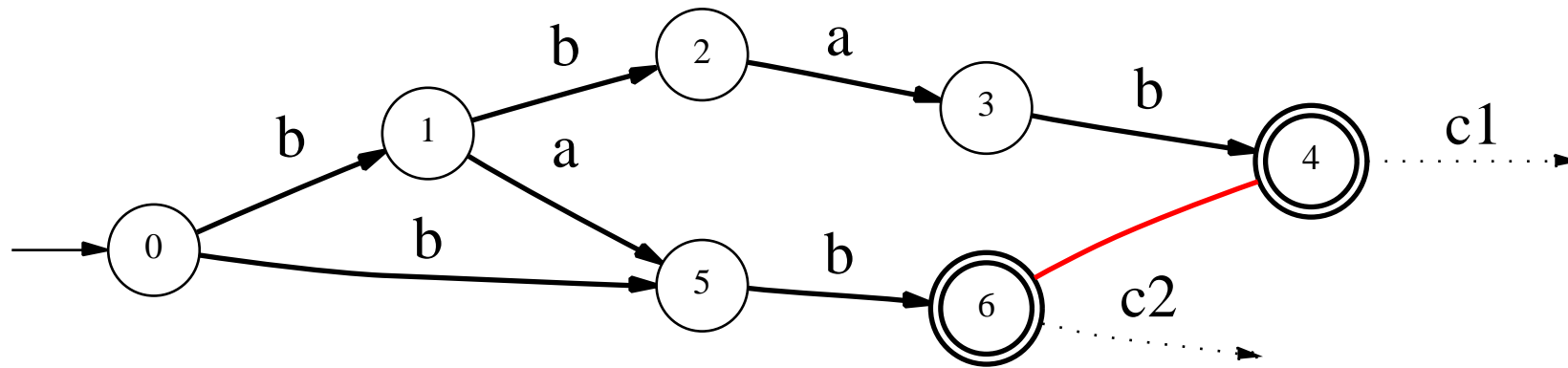
Algorithme de détection de l'ambiguïté d'un k -FSA

- Adaptation de l'algorithme de marquage des états distinguables pour la minimisation [Hopcroft, Ullman 80] par remplacement de \neq par $\not\sim$
- Nouvelle stratégie de propagation
 - Suppression des listes
 - $O(n^2)$ pour les automates à structure d'arbre
- k -FSA ambigu : $\exists q \in Q : q \not\sim q$

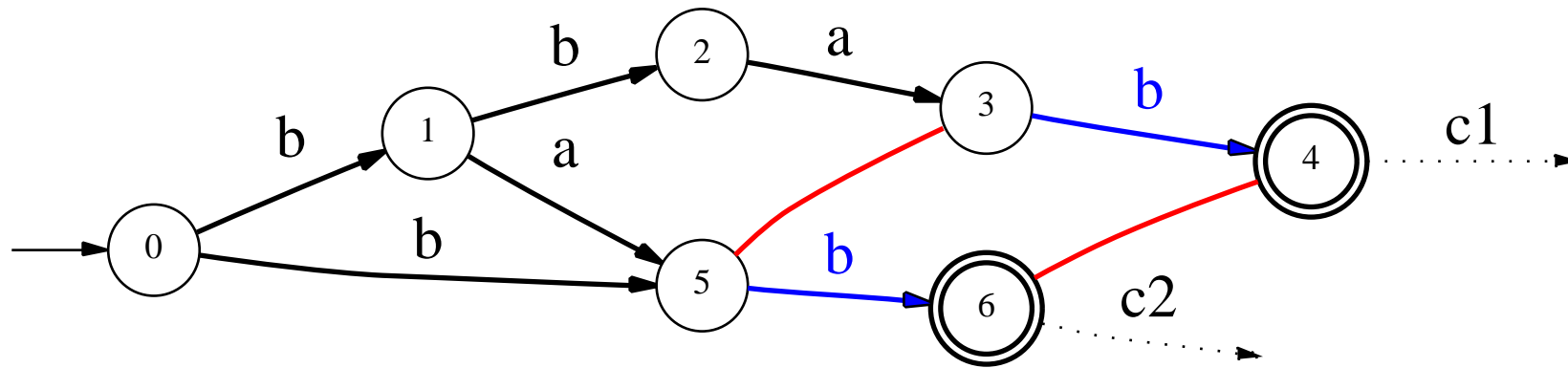
Exemple 1



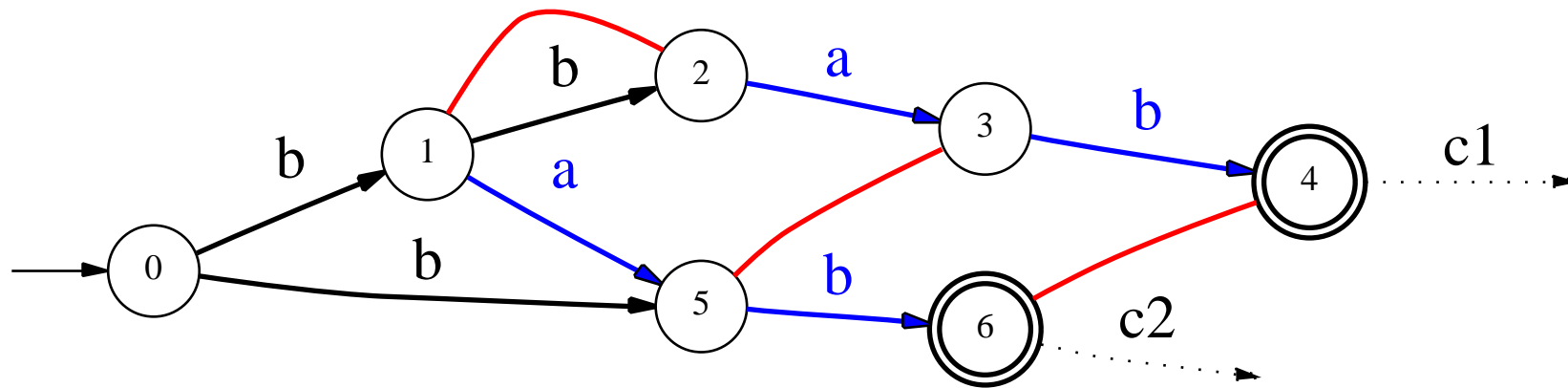
Exemple 1



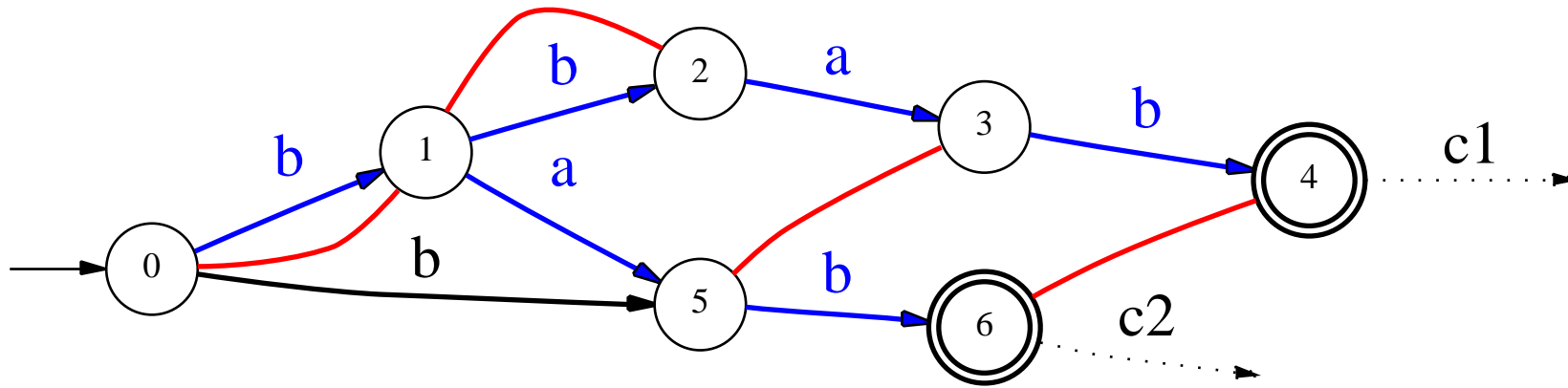
Exemple 1



Exemple 1

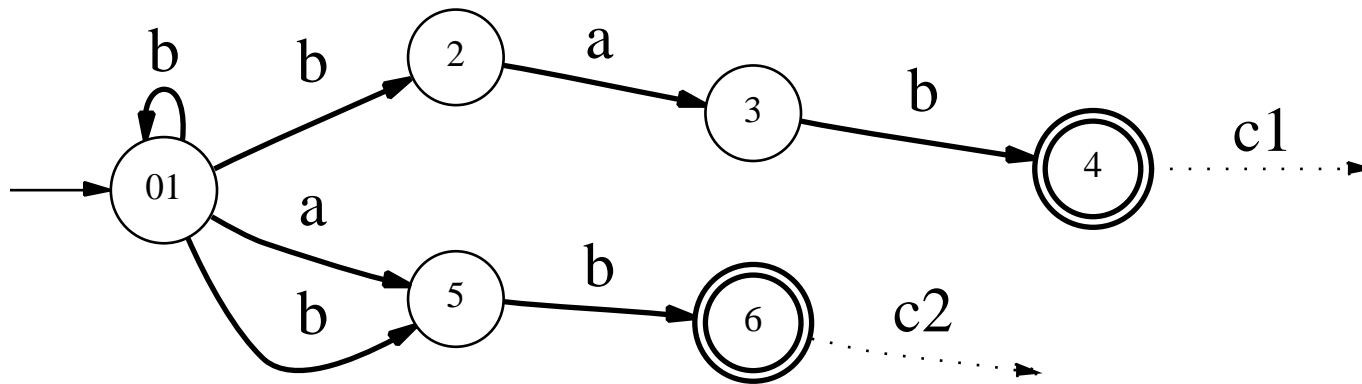


Exemple 1

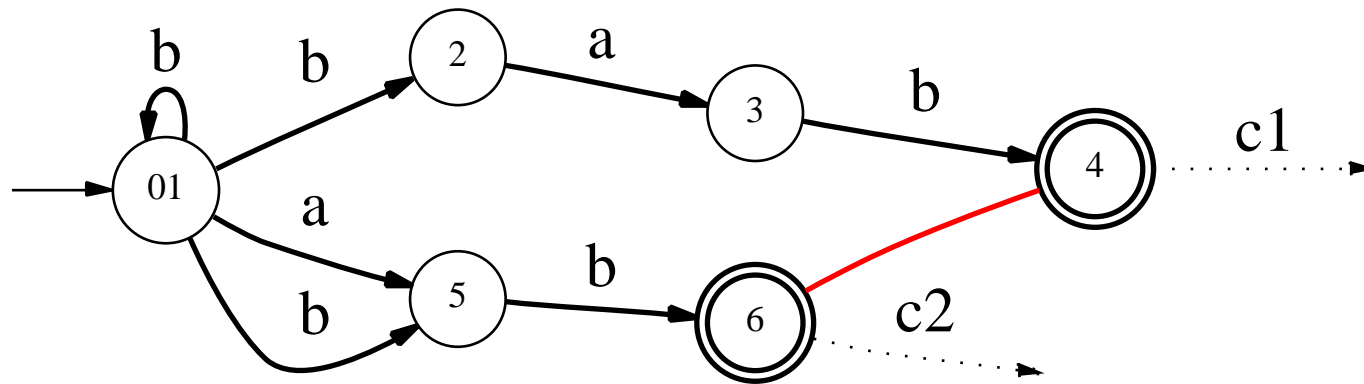


Univoque

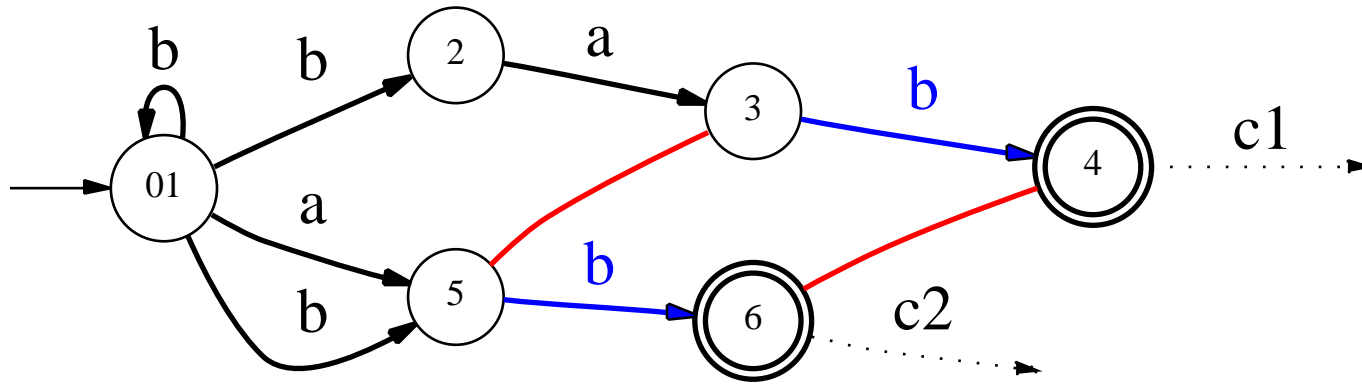
Exemple 2



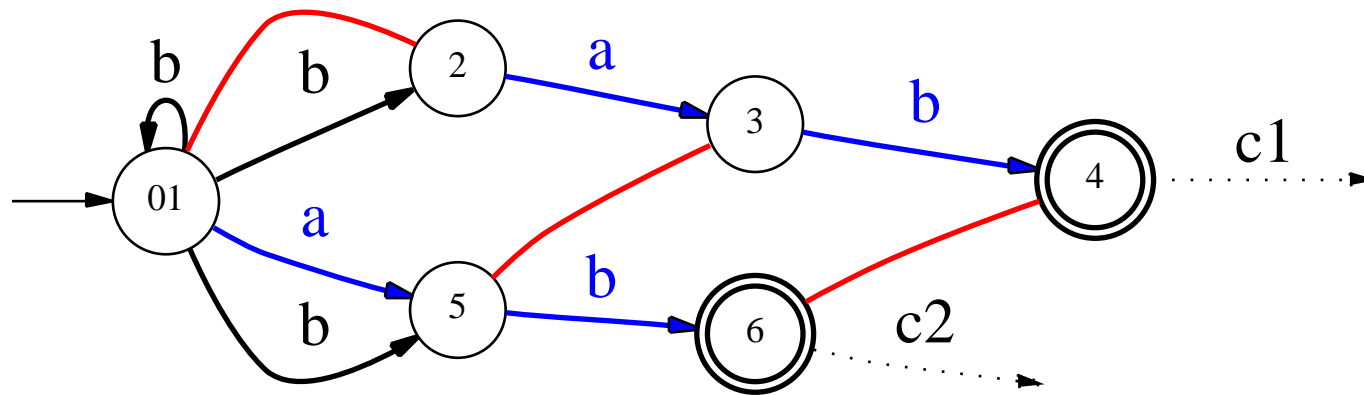
Exemple 2



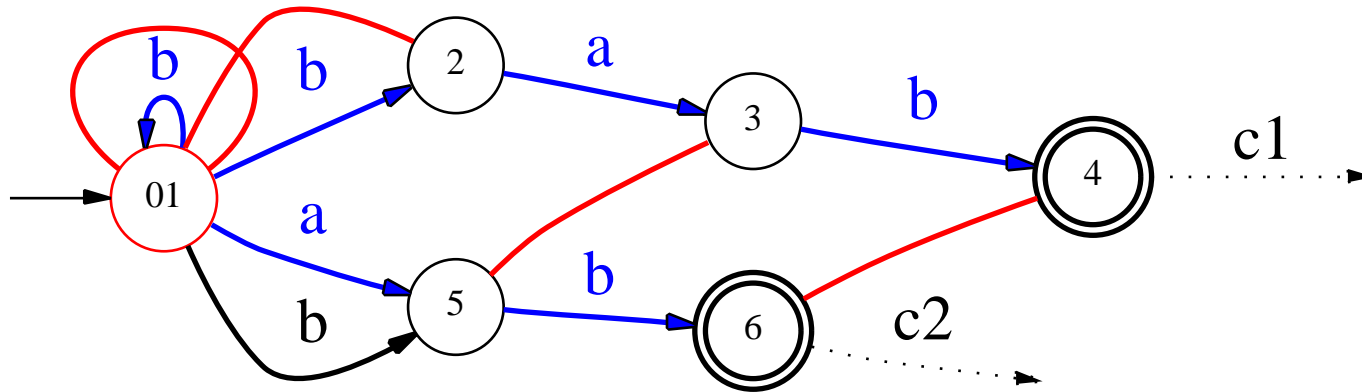
Exemple 2



Exemple 2



Exemple 2

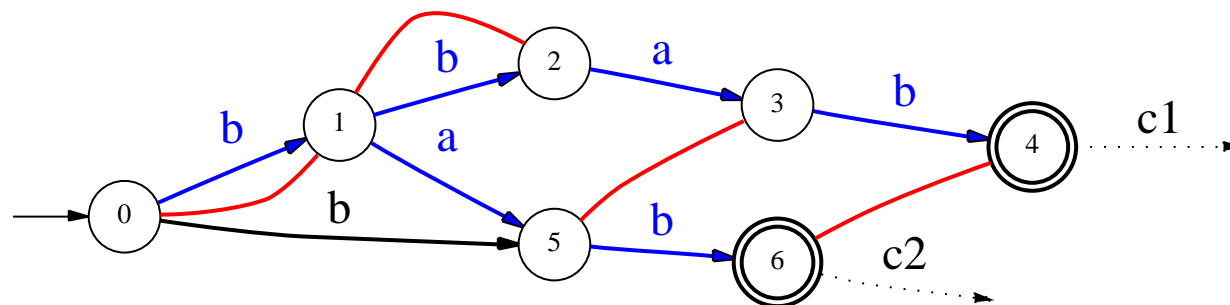


Ambigu

Utilisation pour l'inférence

*L'ensemble des états incompatibles
est croissant par fusion*

- Détection des états incompatibles de $\text{MCA}(\mathcal{I})$
- Version incrémentale de l'algorithme de détection de l'ambiguïté
- États incompatibles \Rightarrow fusions incompatibles

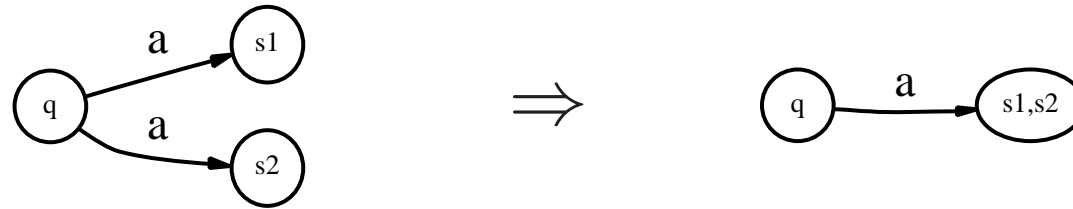


II.B. Inférence de k -DFA univoques

- ▷ Déterminisme : Fusion pour déterminisation
- ▷ Propagation en sens opposé à incompatibilité
 - Expression des solutions sous la forme d'un système de contraintes dans un espace de recherche implicite

Fusions utilisées

- Fusion pour déterminisation



$\forall q \in Q, \forall a \in \Sigma, \forall s_1, s_2 \in \delta(q, a), \text{Fusionner}(s_1, s_2)$

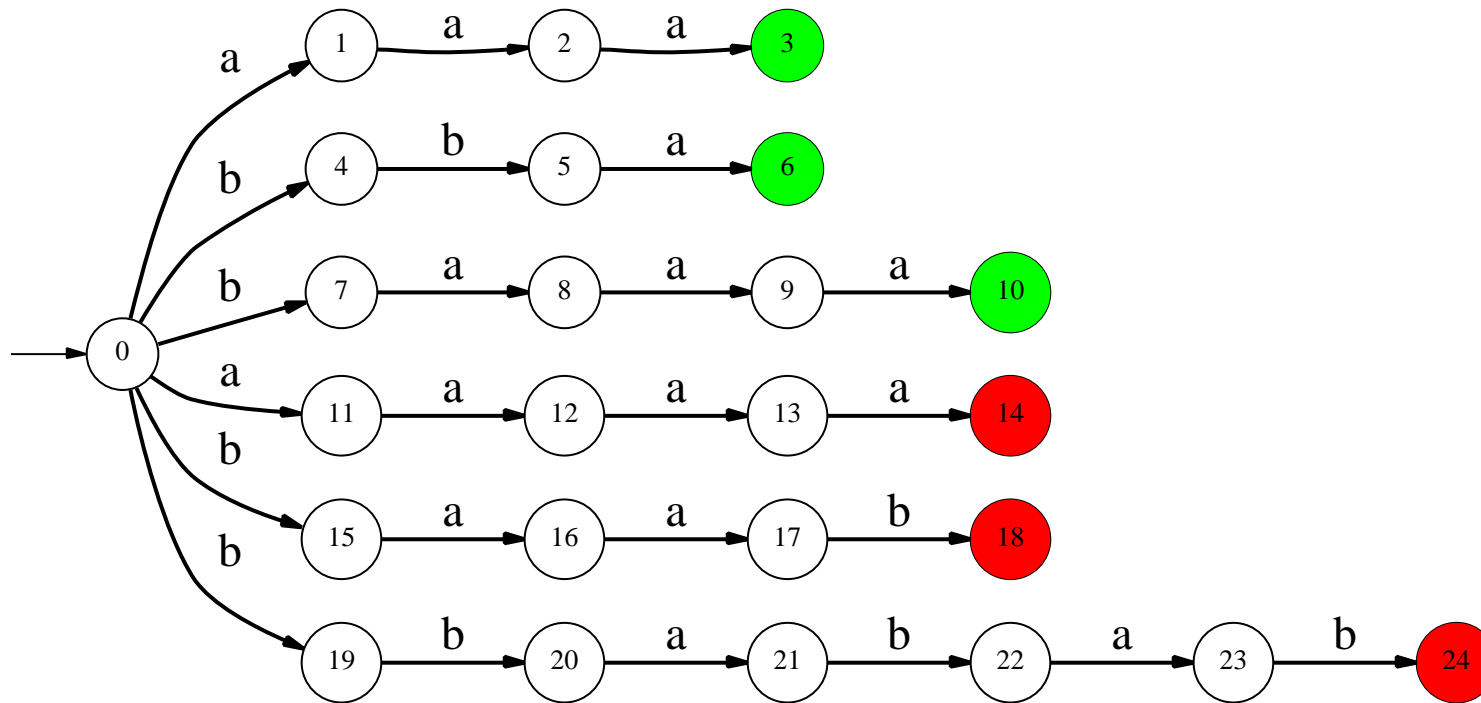
→ automate déterministe dans le treillis,
éventuellement plus général

- Fusion déterministe

Fusion d'états + pour déterminisation

Fusion pour déterminisation

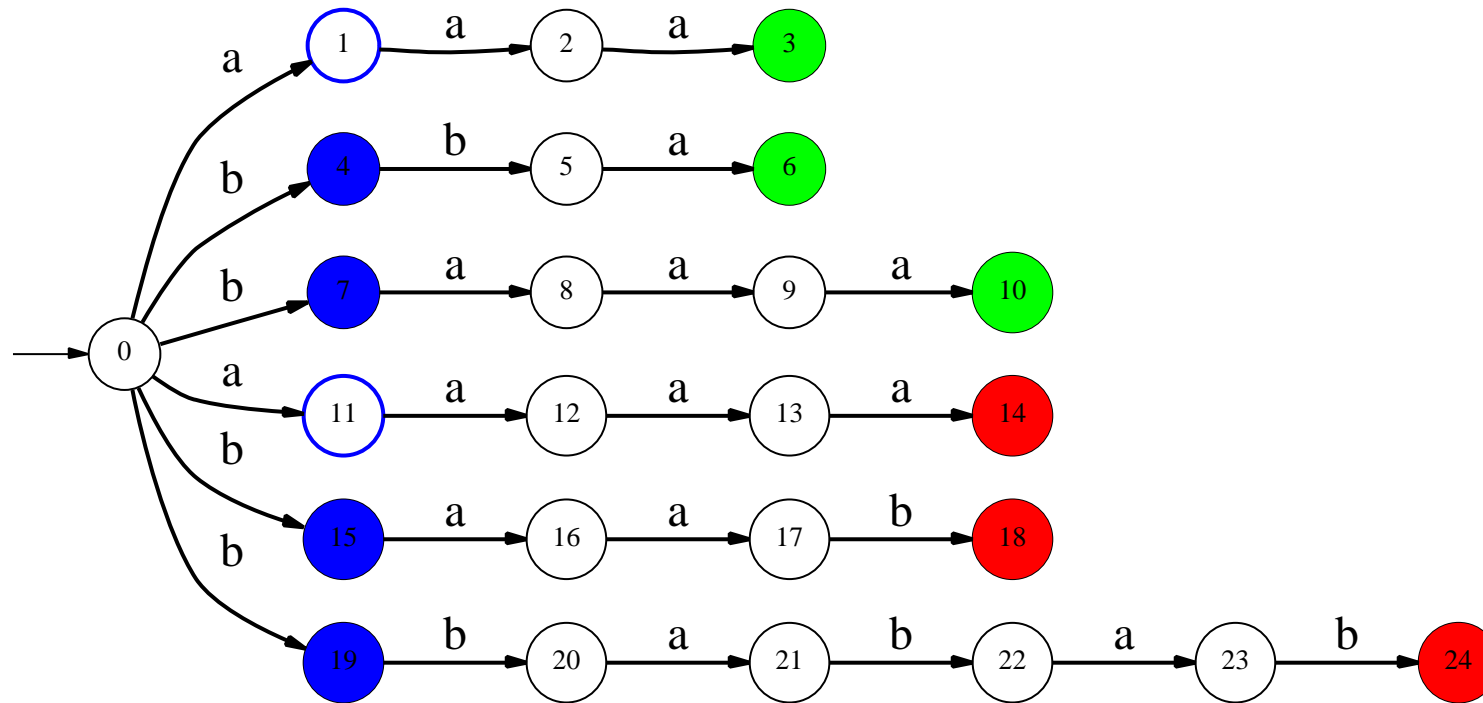
Sur $\text{MCA}(\mathcal{I})$:



$\mathcal{I} : I_+ = \{aaa, bba, baaa\}, I_- = \{aaaa, baab, bbabab\}$

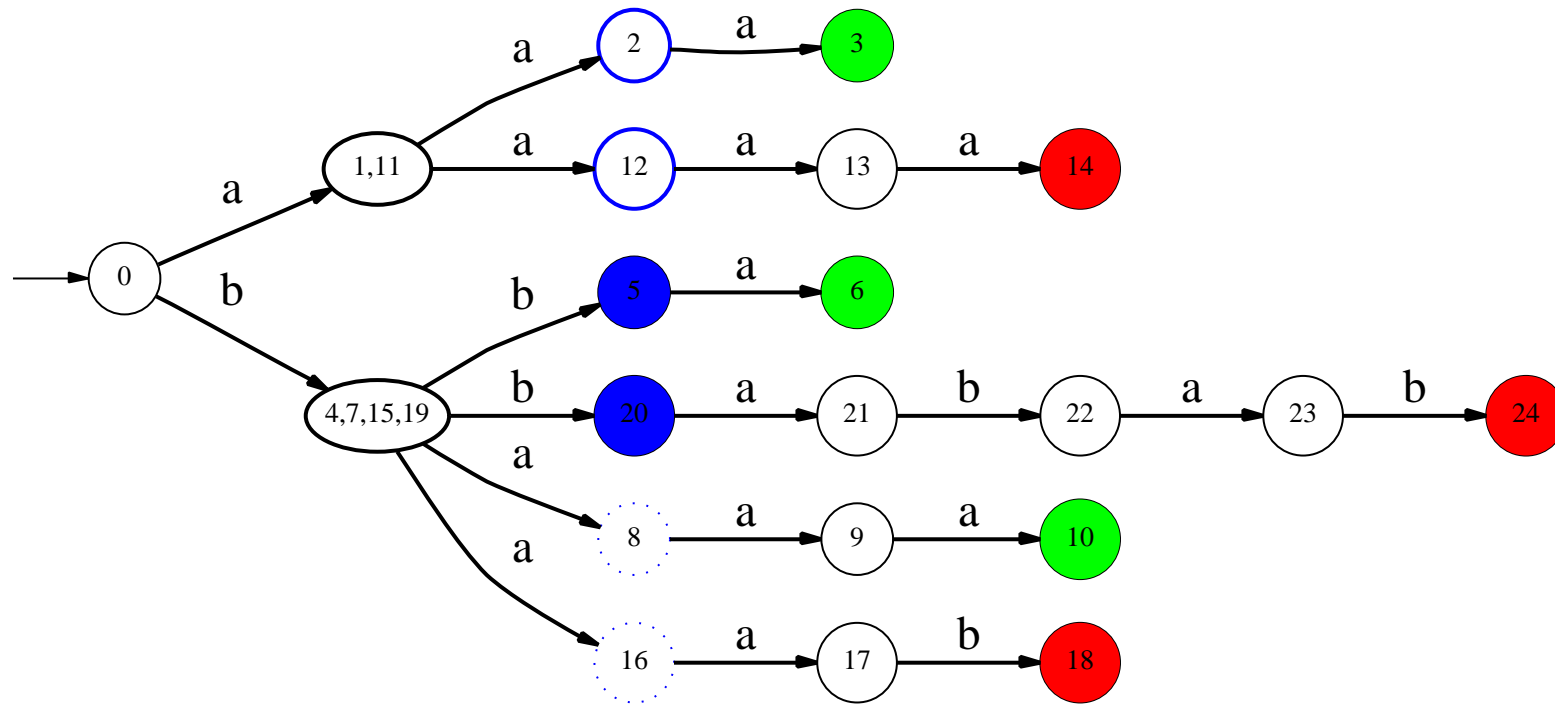
Fusion pour déterminisation

Sur $\text{MCA}(\mathcal{I})$:



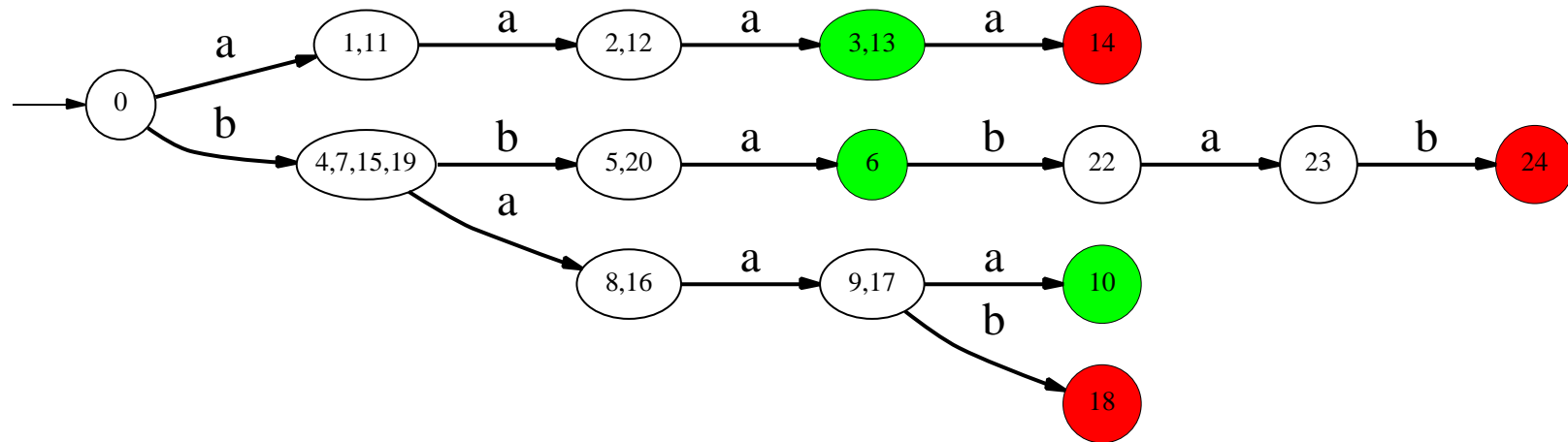
Fusion pour déterminisation

Sur $MCA(\mathcal{I})$:



Fusion pour déterminisation

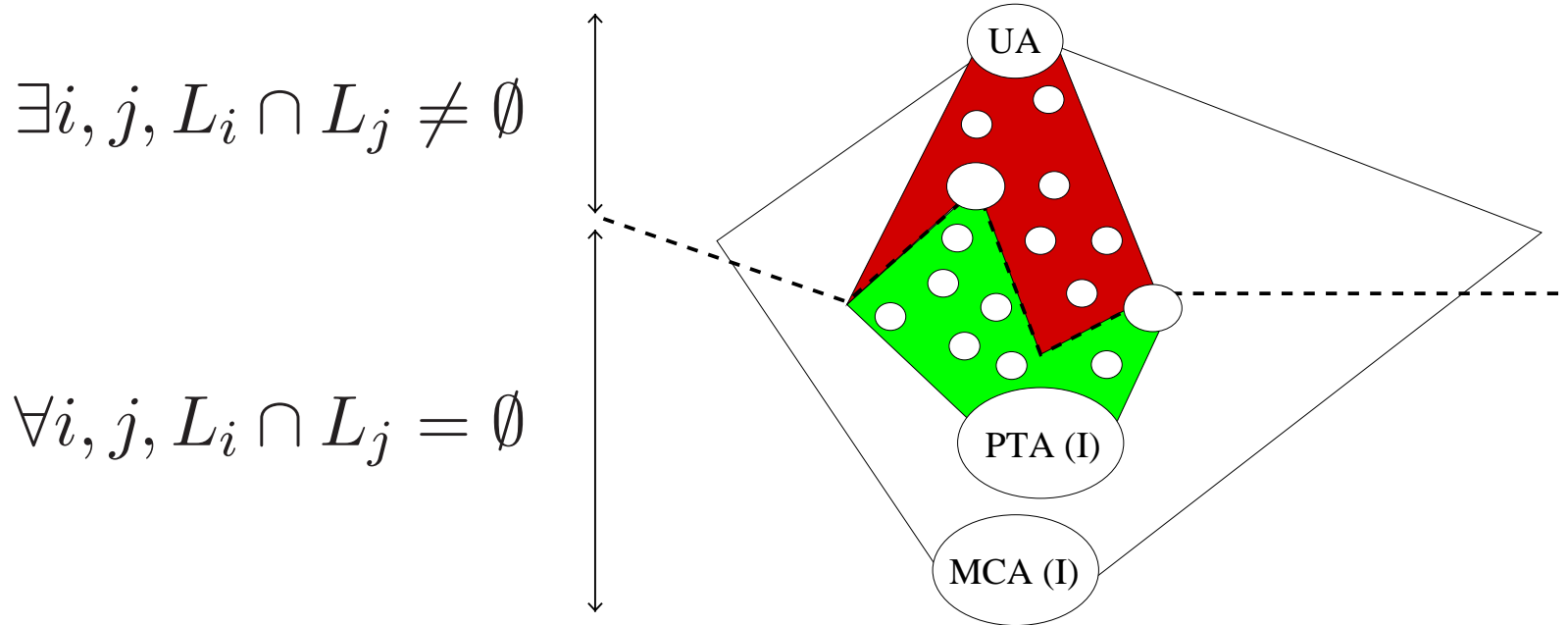
Sur $\text{MCA}(\mathcal{I})$:



= $\text{PTA}(\mathcal{I})$ (*Prefix Tree Acceptor*)

Tout k -DFA pour lequel \mathcal{I} est structurellement complet peut être obtenu par fusions d'états de $\text{PTA}(\mathcal{I})$

Espace de recherche



Pour étudier plus particulièrement la dynamique des fusions et des incompatibilités, passage de l'espace des automates dérivés de $PTA(\mathcal{I})$ à l'espace des fusions sur $PTA(\mathcal{I})$...

Espace des fusions

- Attributs
 - Paires d'états d'un automate A

	q_1	q_2	q_3	q_4
q_1	(\simeq)	\neq	\simeq	?
q_2		(\simeq)	\neq	\neq
q_3			(\simeq)	?
q_4				(\simeq)

Affectation

- Valeurs
 - $q_1 \simeq q_2$: q_1 et q_2 sont fusionnés
 - $q_1 \neq q_2$: q_1 et q_2 différenciés

→ caractérisation des k -DFA univoques sur
l'espace des fusions de $PTA(\mathcal{I}) \dots$

Systeme de contraintes sur $\text{PTA}(\mathcal{I})$

$$\forall q_1, q_2, q_3 \in Q, \forall a \in \Sigma, \forall s_1, s_2 \in \delta(q_1, a) \times \delta(q_2, a)$$

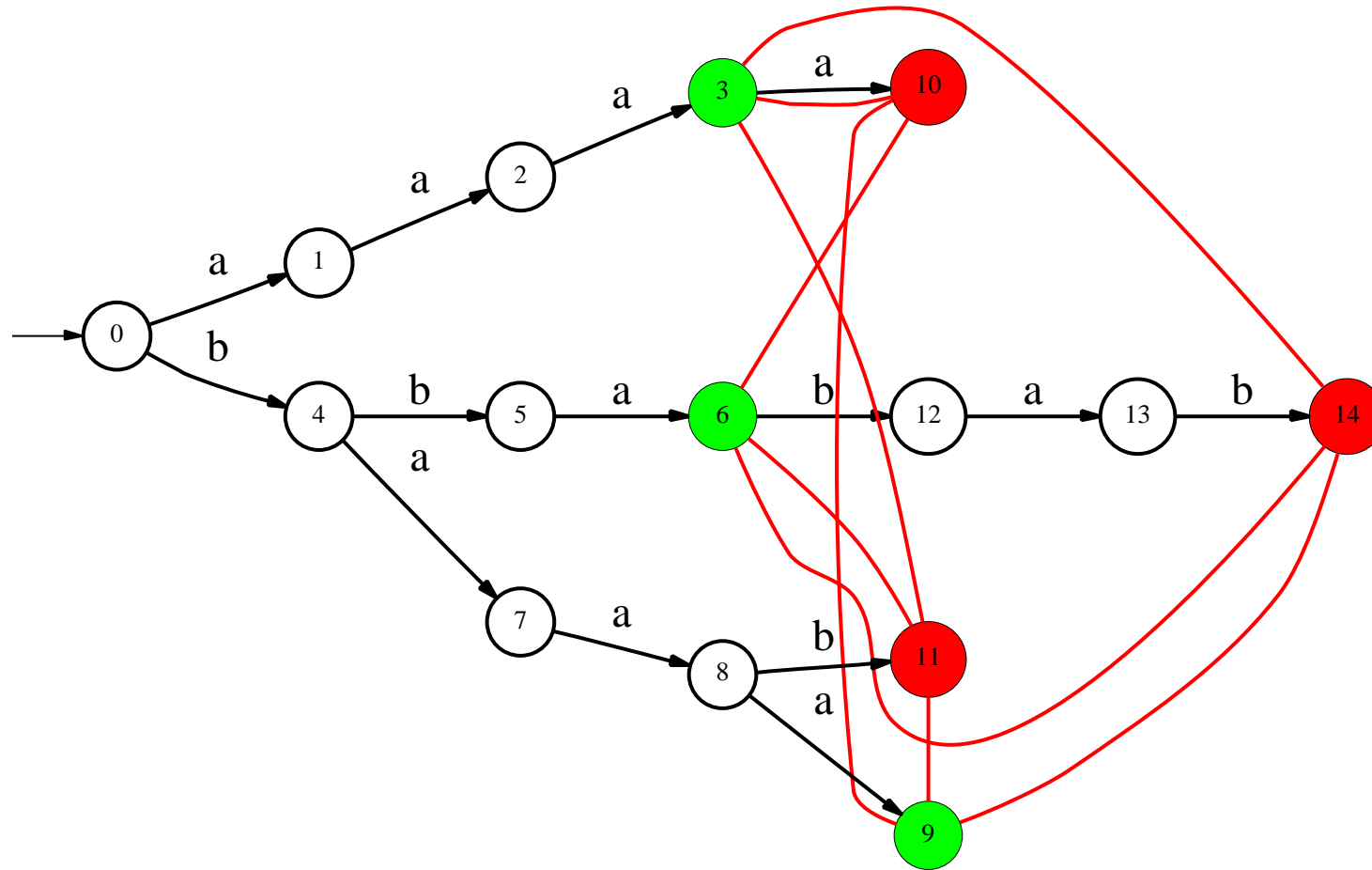
$$\left\{ \begin{array}{l} \text{Transitivité :} \quad q_1 \simeq q_2 \wedge q_2 \simeq q_3 \Rightarrow q_1 \simeq q_3 \\ \text{Unicité :} \quad q_1 \simeq q_2 \Rightarrow \rho(q_1) \sim \rho(q_2) \\ \text{Déterminisme :} \quad q_1 \simeq q_2 \Rightarrow s_1 \simeq s_2 \end{array} \right.$$

Contraposée

$$\left\{ \begin{array}{l} \text{Transitivité :} \quad q_1 \not\simeq q_3 \Rightarrow q_1 \not\simeq q_2 \vee q_2 \not\simeq q_3 \\ \text{Unicité :} \quad \rho(q_1) \not\sim \rho(q_2) \Rightarrow q_1 \not\simeq q_2 \\ \text{Déterminisme :} \quad s_1 \not\simeq s_2 \Rightarrow q_1 \not\simeq q_2 \end{array} \right.$$

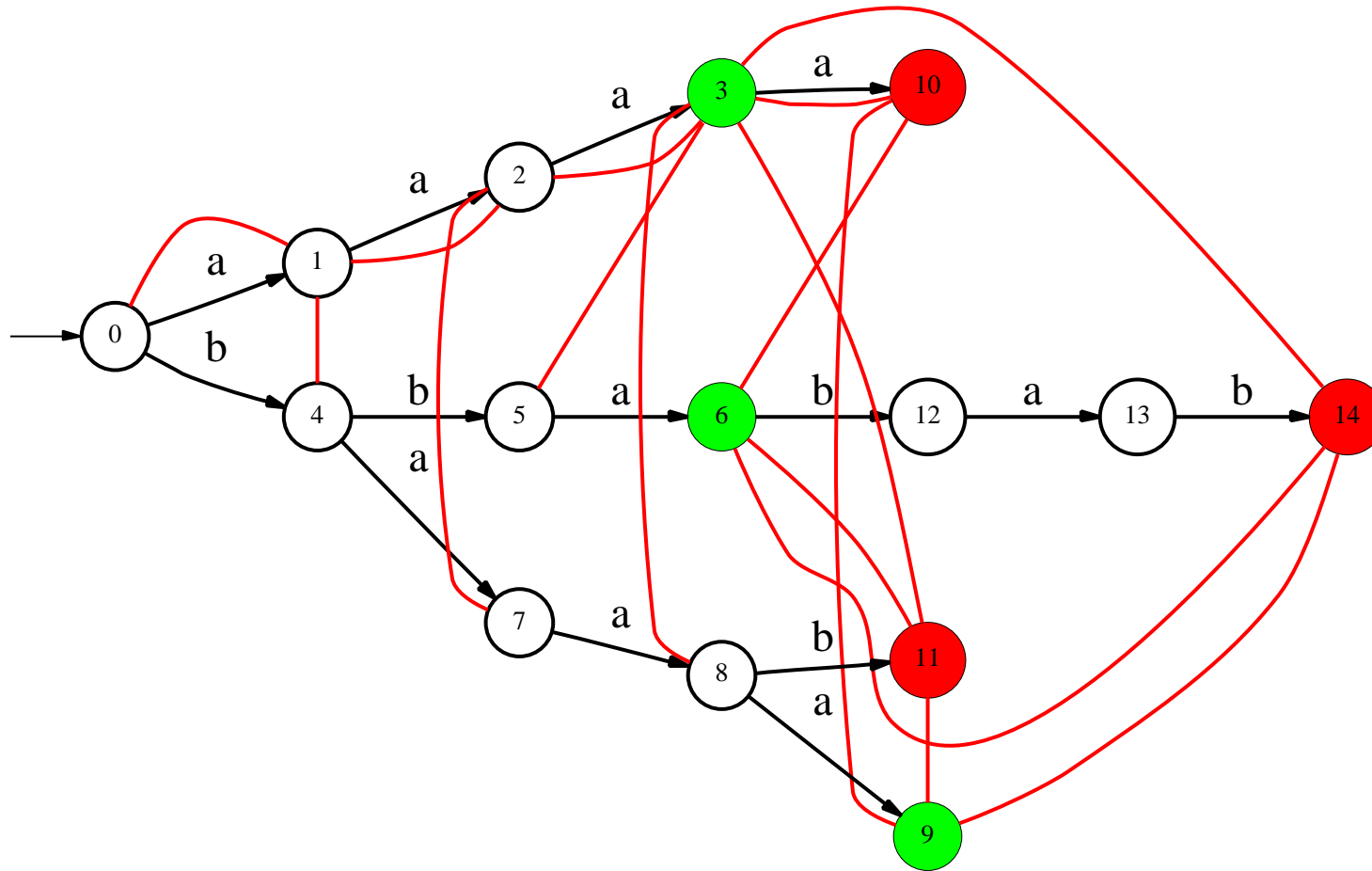
Différenciations initiales

Unicité



Différenciations initiales

Déterminisme



Enumération des solutions

EnumSol()

si Affectation Consistante **alors**

si Affectation Complète **alors**

Solution

sinon

Choix d'une paire d'états : q_1, q_2

Affecte et propage $q_1 \simeq q_2$

EnumSol()

Affecte et propage $q_1 \not\simeq q_2$

EnumSol()

II.C. Expérimentations

Implémentation et validation des idées présentées, constitution d'une bibliothèque orientée objet (C++) pour l'inférence de machines à états finies

- ▷ Recherche exacte d'automates minimaux
- ▷ Deux schémas génériques du type *Branch and Bound*
- ▷ De nombreuses variantes possibles

Recherche de k -DFA univoque minimal

- Problème NP-Complet
- Algorithmes gloutons insuffisants
- Déterminer le domaine d'application des méthodes exactes
- Base d'étude pour les autres méthodes

Algorithme par fusion et différenciation

Cisma() /* Compatible Incompatible sma */

si Affectation Consistante **alors**

si Affectation Complète **alors**

Solution : opt ← *taille solution*

sinon

si *Évaluation partielle* < *opt* **alors**

Choix d'une paire d'états : q_1, q_2

Affecte et propage $q_1 \simeq q_2$

Cisma()

Affecte et propage $q_1 \not\simeq q_2$

Cisma()

Algorithme par coloriage

Colorig(\mathcal{C}) /* \mathcal{C} : ensemble des états coloriés */

si Affectation Consistante **alors**

si Affectation Complète **alors**

 Solution : opt \leftarrow card(\mathcal{C})

sinon

si card(\mathcal{C}) < opt **alors**

 Choix d'un état de $Q - \mathcal{C}$: q

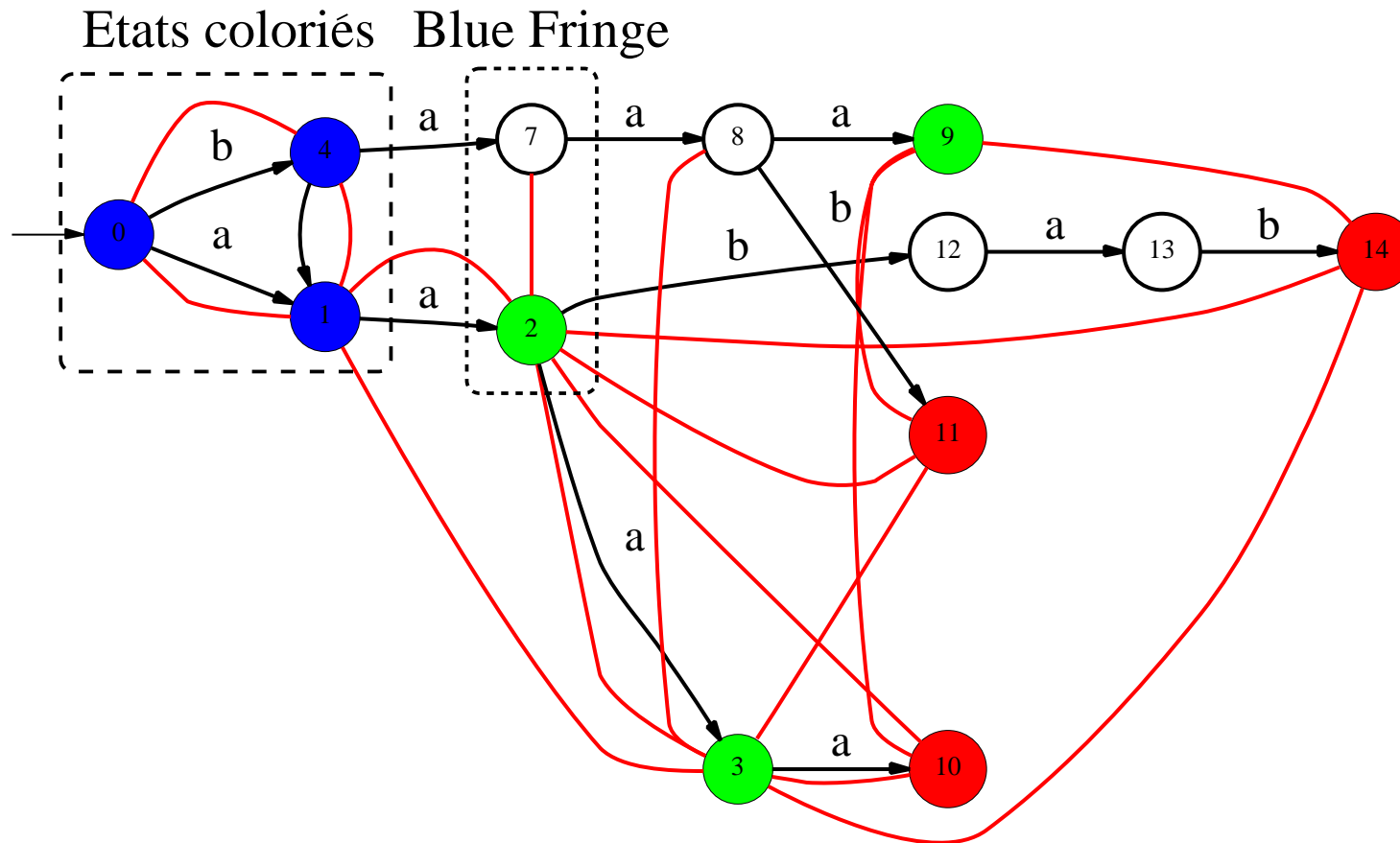
pour tout $q_c \in \mathcal{C}$ **faire**

 Affecte et propage $q \simeq q_c$

 Colorig(\mathcal{C})

 Colorig($\mathcal{C} \cup q$)

Algorithme par coloriage et Blue Fringe



Facteurs de performance

<i>Objectif</i>	<i>Proposition</i>
Obtention rapide d'une bonne borne supérieure	score EDSM, Abbadingo I [Lang, Pearlmutter, Price 1998]
Détection rapide des inconsistances	Propagation suivant le système de contraintes
Bonne évaluation partielle	Taille de clique dans le graphe d'incompatibilité (Cisma) ou nombre d'états de \mathcal{C} (Colorig)

L'ordre de prise en compte des attributs est primordial pour les deux premiers objectifs.

Comparaison des ordonnancements

- Schéma Cisma : Choix de 2 états

Valeurs : \simeq puis \neq

- Extension des algorithmes gloutons

à la recherche complète :

RPNI [Oncina, Garcia 92], EDSM et Blue Fringe [Lang, Pearlmutter, Price 98]

- Stratégie de *hill-climbing* : favoriser le plus grand nombre de fusions

- Schéma Colorig : Choix de 1 état
Valeurs : ordonnancement suivant score EDSM
des états de \mathcal{C} puis promotion dans \mathcal{C}
 - Adaptation des algorithmes gloutons
RPNI, EDSM et Blue Fringe
 - Stratégies de saturation
ColorSat : état avec le plus grand nombre
de couleurs adjacentes cf. DSATUR [Brelaz
79] (diminution des choix possibles)
FnokSat : état avec le plus grand nombre
d'états adjacent (saturation brute de l'espace)

Jeux de tests disponibles

Échantillons générés aléatoirement

- **Dupont (1996)**
Petits automates mais représentatifs
des différents cas de figure
- **Abbadingo (1998)**
Problèmes de trop grande taille pour être
abordés par méthode exacte
- **Gowachin (1999)** (<http://www.irisa.fr/Gowachin/>)
Taille et nombre d'exemples paramétrable

Expérimentations

- Recherche complète
- Recherche avec une borne supérieure donnée par un algorithme glouton EDSM
- Vérification de la taille minimale (donnée)
- Recherche de toutes les solutions de taille minimale (donnée)

Mesure de performance effectuée
en nombre de fusions

Conclusions

- **Avantage à Colorig**
Concentration de la recherche
sur un nombre réduit d'états
- **Confirmation de EDSM couplée à Blue Fringe**
Bon compromis exploration-exploitation
Erreurs du début très coûteuses
- Lorsque la borne supérieure est bonne, notre
stratégie **FnokSat** de saturation des incom-
patibilités entre états est très efficace.

Benchmark [Oliveira, Silva 2000]

à paraître dans *Machine Learning Journal*

1520 problèmes, 2 à 18 états, échantillons d'apprentissage :
20 mots de 30 symboles

- Glouton EDSM obtient un automate minimal pour 964 problèmes, reste 556 problèmes
- Difficultés d'obtention d'une bonne borne supérieure
- Utilisation d'une borne supérieure croissante
 - BIC : 89 non résolus (16%)
 - FnokSat : 23 non résolus (4%)

Conclusions

- Bases de l'inférence d'automates classifieurs (univoques)
 - Formalisation
 - Étude de l'espace de recherche
 - Algorithmes efficaces
 - * Glouton : EDSM
 - * Recherche exacte : Colorig FnokSat
 - * Recherches heuristiques ?
 - divergence limitée,*
 - profondeur d'abord entrelacée*

Perspectives

- Application à d'autres machines
k-NFA, automates classifieurs de Mealy, Transducteurs
- Traiter des problèmes réels
 - Critères qualitatifs à optimiser
 - Considérer les symboles de transition
*Réintroduire la notion de valeur
sur les symboles ...*