
Vers l'apprentissage de requêtes

A. Lemay

GRAPPA - Lille 3
INRIA MOSTRARE

1. Apprentissage de "DTD" (automates d'arbres)
2. Apprentissage de requête (transducteurs d'arbres)

Apprentissage de 'DTD' (au sens large)

- DTD
- XML-scheme
- ...

Apprentissage de 'DTD' (au sens large)

- DTD
- XML-scheme
- ...

⇒ Langages d'arbres rationnels (représentables par automates d'arbres)

Apprentissage de 'DTD' (au sens large)

- DTD
- XML-scheme
- ...

⇒ Langages d'arbres rationnels (représentables par automates d'arbres)

Cadre : apprentissage par données positives

Classes de langages “apprenables” par exemples positifs

1. dans les mots.

- Langages réguliers : NON !
- langages **réversibles** de mots : OUI

2. en arité bornés.

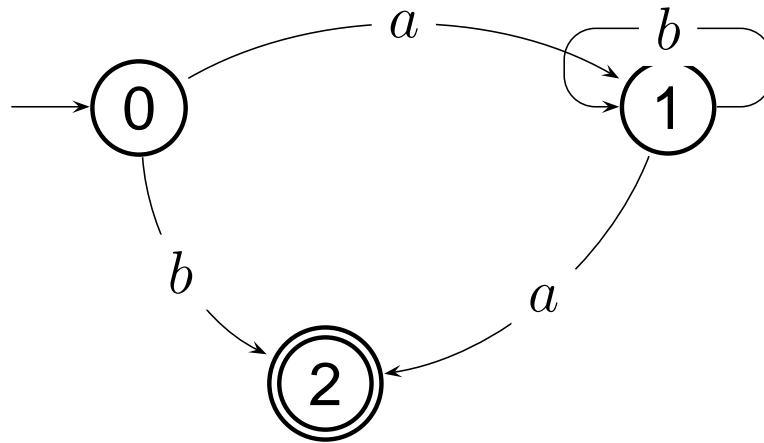
Langages réversibles d'arbres

3. en arité non bornée.

langages dont l'arbre de construction est réversible
(ex : stepwise reversible)

Automates 0-Reversibles et ZR

Automate 0-réversible :



⇒ automate **déterministe** dont le **miroir** est déterministe.

Langage 0-réversible = reconnu par un automate 0-réversible

⇒ Identifiable à la limite par exemples positifs (ZR [Angluin 82])

ZR : sort toujours un langage réversible (compatible avec l'échantillon, le plus 'gros' possible)

MOSTRARE

- Automate ascendant déterministe

1 terme \Leftrightarrow 1 état

$\forall \triangle_t, \exists! q$ tel que $\triangle_t \rightarrow q$

- Automate ascendant déterministe

1 terme \Leftrightarrow 1 état

$\forall \triangle_t, \exists! q$ tel que $\triangle_t \rightarrow q$

- Automate descendant déterministe pour contextes

1 contexte \Leftrightarrow 1 état

$\forall \triangle_c, \exists! q$ tel que $\triangle_c \rightarrow q$

- Automate ascendant déterministe

1 terme \Leftrightarrow 1 état

$\forall \triangle_t, \exists! q$ tel que $\triangle_t \rightarrow q$

- Automate descendant déterministe pour contextes

1 contexte \Leftrightarrow 1 état

$\forall \triangle_c, \exists! q$ tel que $\triangle_c \rightarrow q$

- Automate réversible : Automate **déterministe** ascendant et **déterministe pour contextes**

- Automate ascendant déterministe

1 terme \Leftrightarrow 1 état

$$\forall \triangle_t, \exists! q \text{ tel que } \triangle_t \rightarrow q$$

- Automate descendant déterministe pour contextes

1 contexte \Leftrightarrow 1 état

$$\forall \triangle_c, \exists! q \text{ tel que } \triangle_c \rightarrow q$$

- Automate réversible : Automate **déterministe** ascendant et **déterministe pour contextes**
- Identifiables à la limite par exemples positifs

[Sakakibara 92], [Besombes - Marion 02]

Langages d'arbres d'arité non bornée

langages dont l'arbre de construction est réversible

Ex : step-wise réversible

règles de la forme :

- $q_1 \oplus q_2 \rightarrow q_3$ **et** $q'_1 \oplus q_2 \rightarrow q_3 \Rightarrow q_1 = q'_1$
- $q_1 \oplus q_2 \rightarrow q_3$ **et** $q_1 \oplus q'_2 \rightarrow q_3 \Rightarrow q_2 = q'_2$
- $q_1 \oplus q_2 \rightarrow q_3$ **et** $q_1 \oplus q_2 \rightarrow q'_3 \Rightarrow q_3 = q'_3$

Langages d'arbres d'arité non bornée

langages dont l'arbre de construction est réversible

Ex : step-wise réversible

règles de la forme :

- $q_1 \oplus q_2 \rightarrow q_3$ et $q'_1 \oplus q_2 \rightarrow q_3 \Rightarrow q_1 = q'_1$
- $q_1 \oplus q_2 \rightarrow q_3$ et $q_1 \oplus q'_2 \rightarrow q_3 \Rightarrow q_2 = q'_2$
- $q_1 \oplus q_2 \rightarrow q_3$ et $q_1 \oplus q_2 \rightarrow q'_3 \Rightarrow q_3 = q'_3$

La classe des langages stepwise-reversibles est (probablement) apprenables par exemples positifs.

- extension aux (multi)-algèbre
- problème des reversibles : très contraignant ?
- utilisation des AFER ?

Une requête : Un couple (A, q) (A : automate d'arbre)

Ex : $A = \{\Sigma, \{q_0, q_1, q_f\}, \{q_f\}, \Delta\}$ avec Δ :

$a \quad \rightarrow \quad q_0 \mid q_1$

$f(q_0, q_1) \quad \rightarrow \quad q_f$

$f(q_0, q_f) \quad \rightarrow \quad q_f$

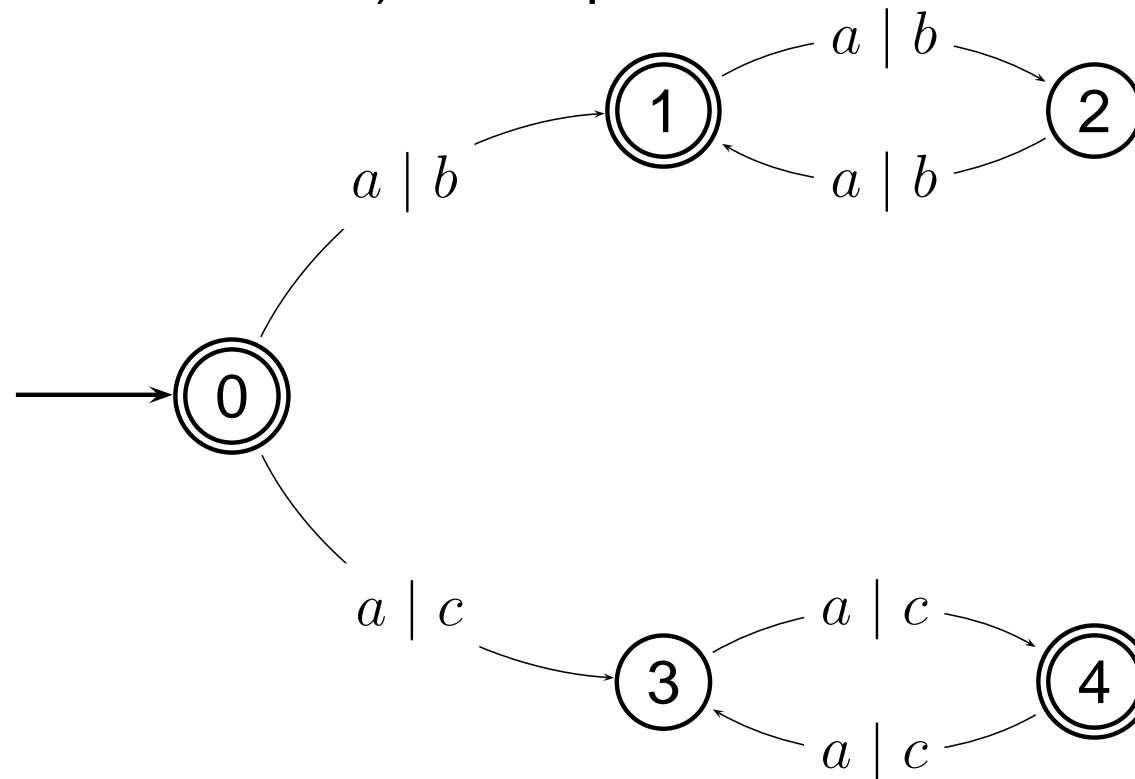
$f(q_f, q_f) \quad \rightarrow \quad q_f$

$f(q_f, q_f) \quad \rightarrow \quad q_f$

$\Rightarrow (A, q_0)$: donne toutes les premiers fils qui sont des feuilles

Une requête : une fonction rationnelle 'relabelling'

fonction rationnelle (dans les mots). Exemple :



Transductions rationnelles : résultats

- fonction rationnelle = transduction rationnelle *fonctionnelle*

Transductions rationnelles : résultats

- fonction rationnelle = transduction rationnelle *fonctionnelle*
- transducteur séquentiel : “déterministe sur l’entrée”

Transductions rationnelles : résultats

- fonction rationnelle = transduction rationnelle *fonctionnelle*
- transducteur séquentiel : “déterministe sur l’entrée”
- Une fonction rationnelle n’est pas toujours séquentielle

Transductions rationnelles : résultats

- fonction rationnelle = transduction rationnelle *fonctionnelle*
- transducteur séquentiel : “déterministe sur l’entrée”
- Une fonction rationnelle n’est pas toujours séquentielle
- $FRAT = SEQ_{\rightarrow} \circ SEQ_{\leftarrow}$

- transducteur d'arbre de $\mathcal{T}(\Sigma) \rightarrow \mathcal{T}(\Sigma')$: règles de la forme
 $f(q_1(x_1), \dots, q_n(x_n)) \rightarrow q(t(x_1, \dots, x_n))$ avec $t \in \mathcal{T}(\Sigma' \cup \{x_1, \dots, x_n\})$.

- transducteur d'arbre de $\mathcal{T}(\Sigma) \rightarrow \mathcal{T}(\Sigma')$: règles de la forme $f(q_1(x_1), \dots, q_n(x_n)) \rightarrow q(t(x_1, \dots, x_n))$ avec $t \in \mathcal{T}(\Sigma' \cup \{x_1, \dots, x_n\})$.
- transducteur d'arbre *Relabelling* : règles de la forme $f(q_1(x_1), \dots, q_n(x_n)) \rightarrow q(g(x_1, \dots, x_n))$ avec $g \in \Sigma'$.

- transducteur d'arbre de $\mathcal{T}(\Sigma) \rightarrow \mathcal{T}(\Sigma')$: règles de la forme $f(q_1(x_1), \dots, q_n(x_n)) \rightarrow q(t(x_1, \dots, x_n))$ avec $t \in \mathcal{T}(\Sigma' \cup \{x_1, \dots, x_n\})$.
- transducteur d'arbre *Relabelling* : règles de la forme $f(q_1(x_1), \dots, q_n(x_n)) \rightarrow q(g(x_1, \dots, x_n))$ avec $g \in \Sigma'$.
- relabelling = préservant la structure

- transducteur d'arbre de $\mathcal{T}(\Sigma) \rightarrow \mathcal{T}(\Sigma')$: règles de la forme $f(q_1(x_1), \dots, q_n(x_n)) \rightarrow q(t(x_1, \dots, x_n))$ avec $t \in \mathcal{T}(\Sigma' \cup \{x_1, \dots, x_n\})$.
- transducteur d'arbre *Relabelling* : règles de la forme $f(q_1(x_1), \dots, q_n(x_n)) \rightarrow q(g(x_1, \dots, x_n))$ avec $g \in \Sigma'$.
- relabelling = préservant la structure

Ex : $A = \{\Sigma, \Sigma' = \{0, 1\}, Q, q_f, \Delta\}$ avec Δ :

$$a \quad \rightarrow \quad q_0(1) \mid q_1(0)$$

$$f(q_0, q_1) \quad \rightarrow \quad q_f(0)$$

$$f(q_0, q_f) \quad \rightarrow \quad q_f(0)$$

$$f(q_f, q_f) \quad \rightarrow \quad q_f(0)$$

$$f(q_f, q_f) \quad \rightarrow \quad q_f(0)$$

Transducteur qui étiquette par un 1 tous les premiers fils feuilles

$$F_{RELAB} = DB_{RELAB} \circ DT_{RELAB}$$

“Equivalence” entre :

- *requêtes* et
- *transducteurs d'arbres fonctionels relabelling*

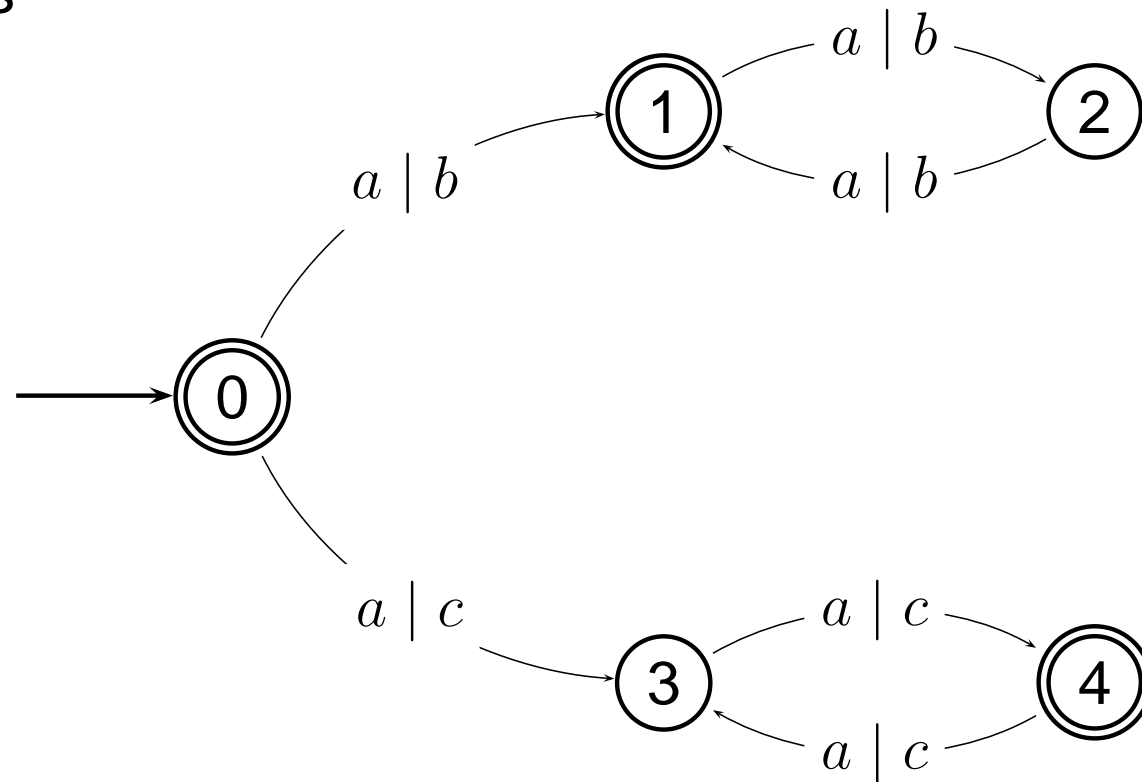
“Equivalence” entre :

- *requêtes* et
- *transducteurs d'arbres fonctionels relabelling*

En tout cas, ce serait pas mal...

Résultat : toutes les fonctions peuvent être mis sous la forme d'un transducteur déterministes sur les couples entrée sortie.

- Vrai dans les mots



- Vrai dans les arbres ? (pour les relabelling)

Algorithme (dans les mots)

Dans les mots : fonction rationnelle préservant les longueurs.

Échantillon (positif)
(couple entrées/sorties)

Algorithme

Transducteur

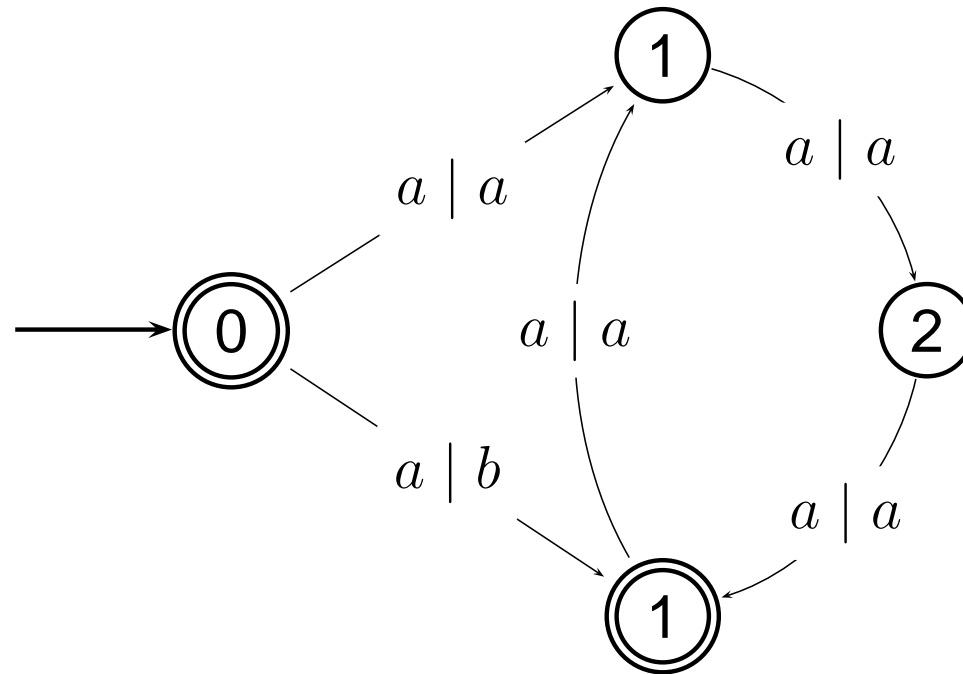
Domaine
(langage support)

Algorithme : variante de RPNI

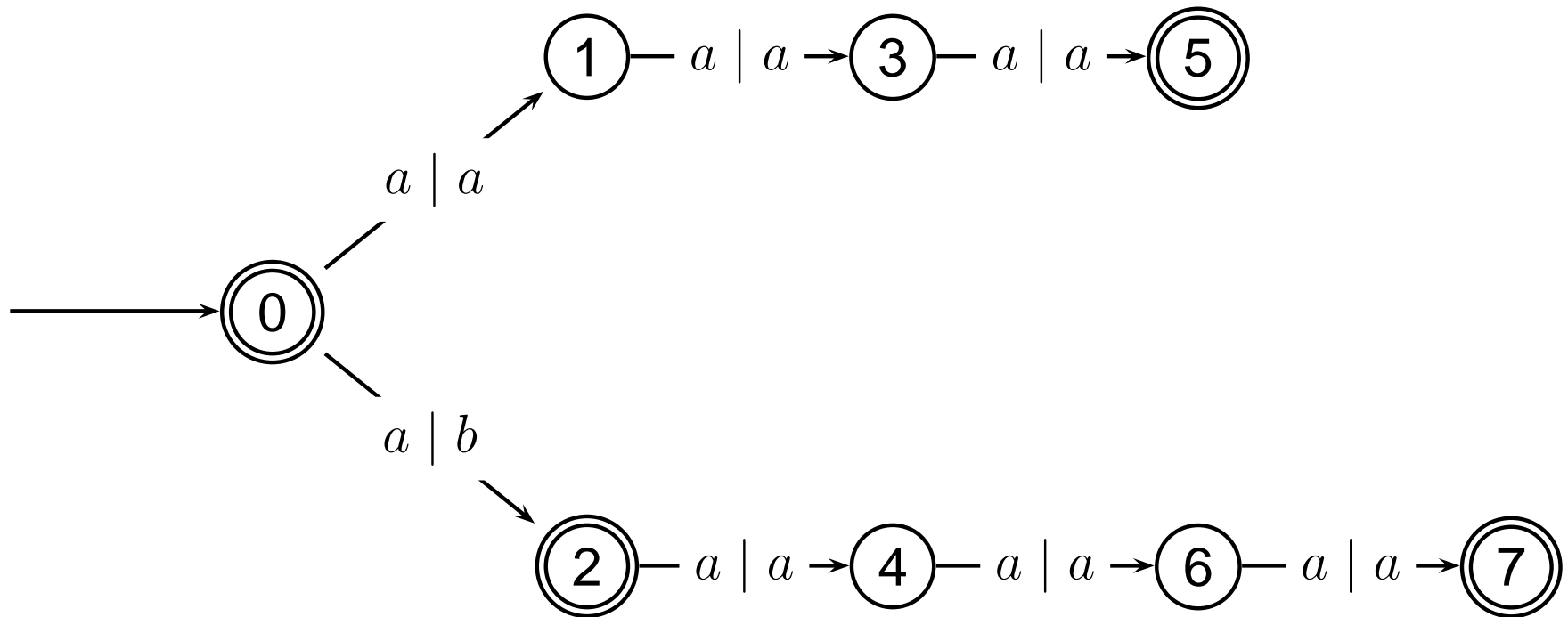
Apprentissage de fonctions rationnelles préservant les longueurs

Cible : $a^{3n} \rightarrow a^{3n}$
 $a^{3n+1} \rightarrow ba^{3n}$

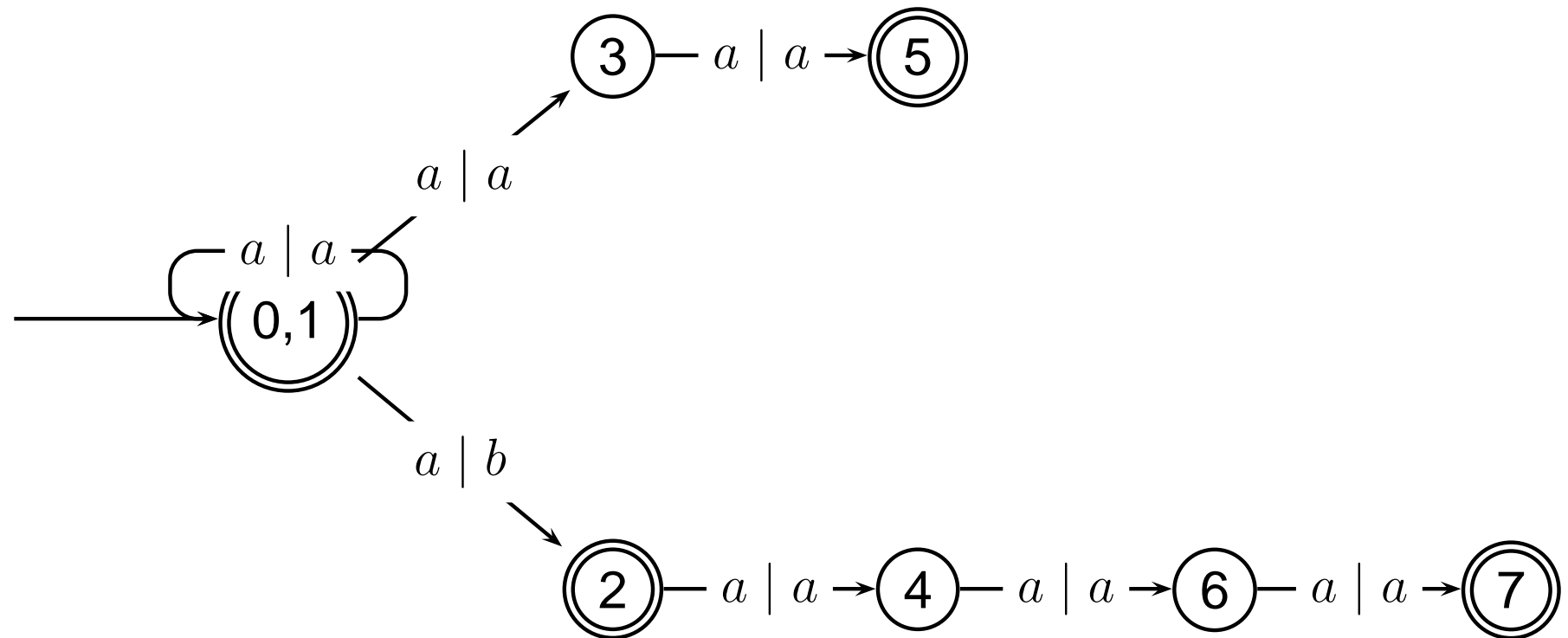
Domaine : $a^{3n} + a^{3n+1}$



- Echantillon : $S = \{(\varepsilon, \varepsilon), (aaa, aaa), (a, b), (aaaa, baaa)\}$
- Domaine : $a^{3n} + a^{3n+1}$

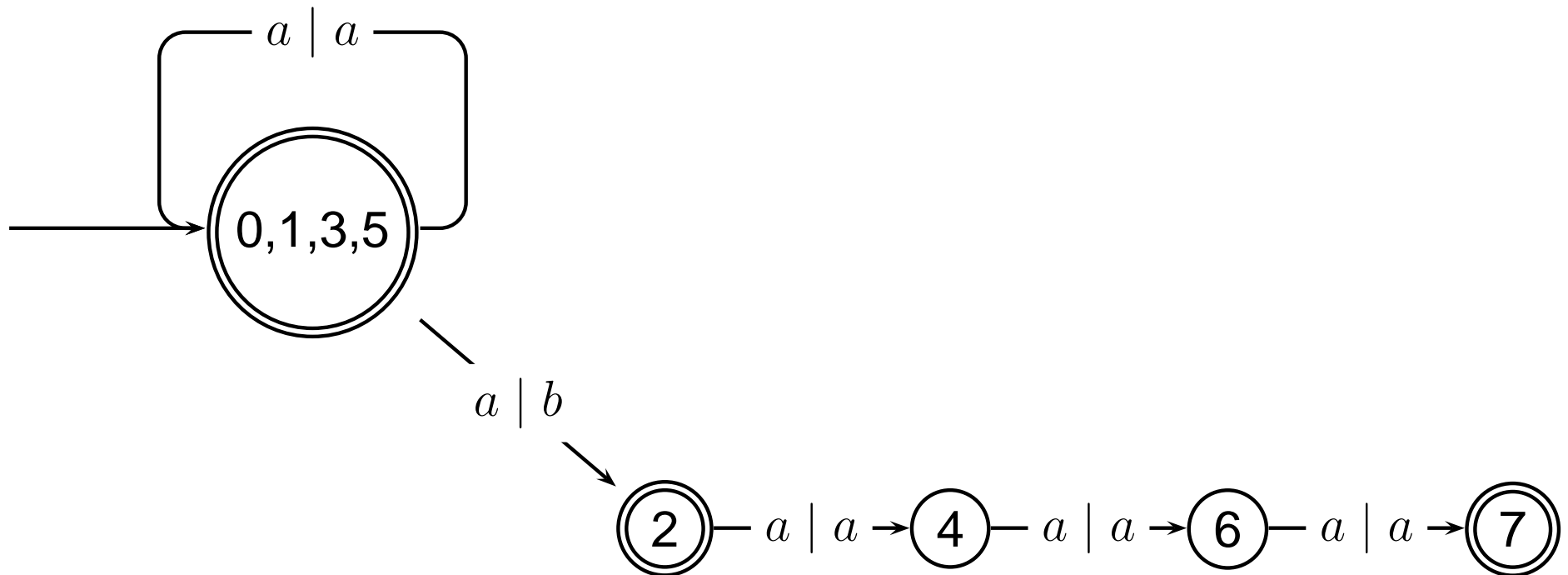


- Echantillon : $S = \{(\varepsilon, \varepsilon), (aaa, aaa), (a, b), (aaaa, baaa)\}$
- Domaine : $a^{3n} + a^{3n+1}$



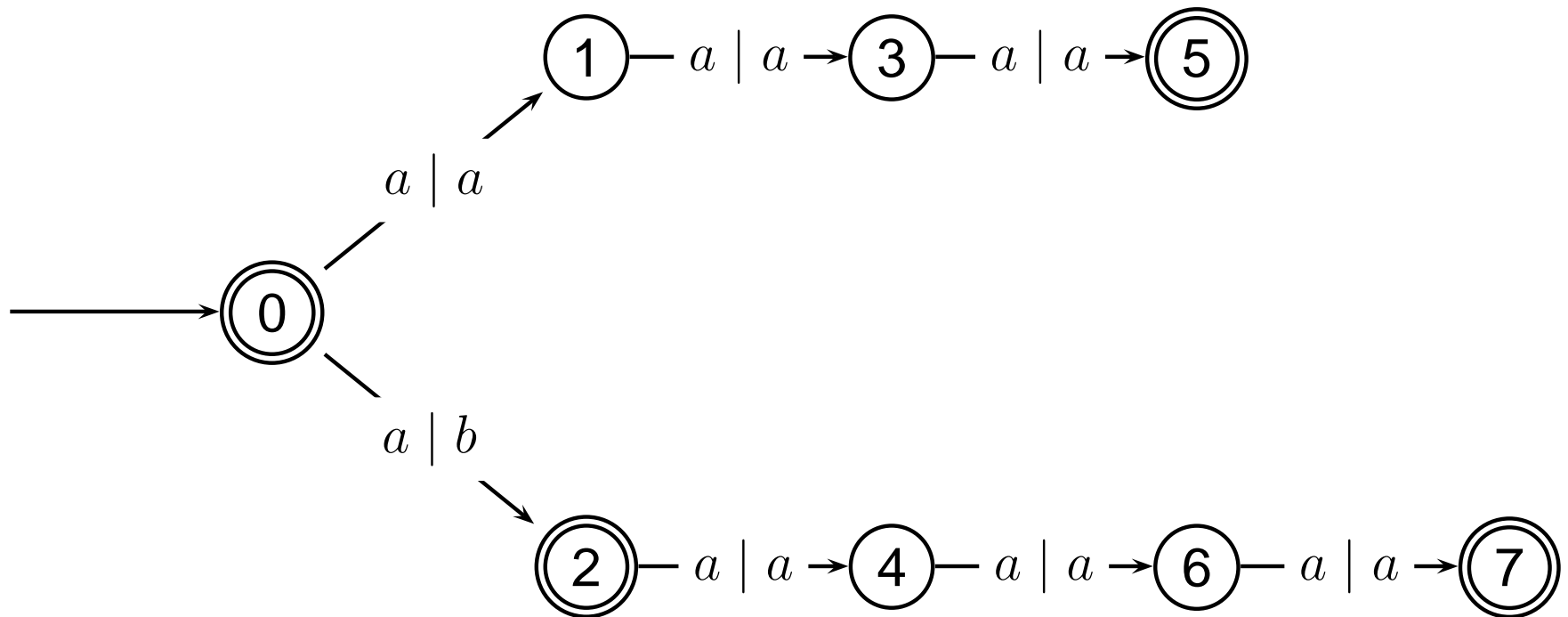
Fusion 0 – 1.

- Echantillon : $S = \{(\varepsilon, \varepsilon), (aaa, aaa), (a, b), (aaaa, baaa)\}$
- Domaine : $a^{3n} + a^{3n+1}$



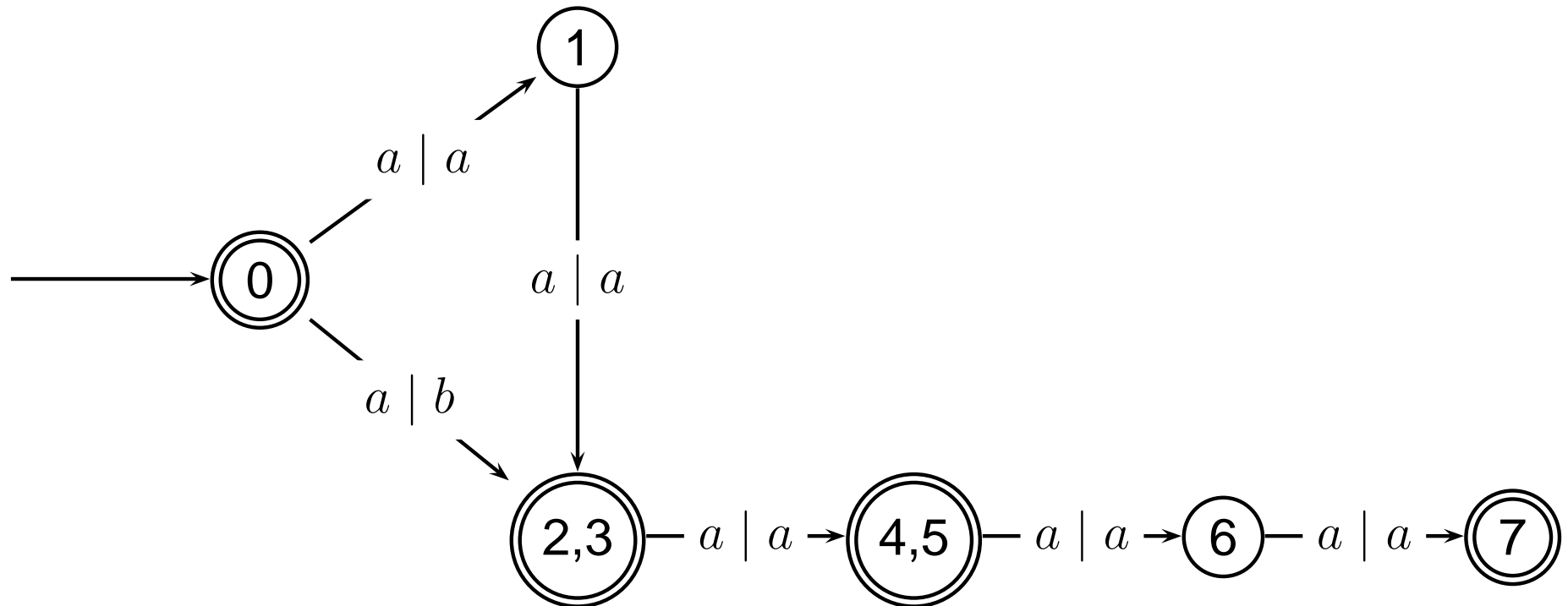
Fusion déterministe rejetée ! (non fonctionnel + sort du domaine)

- Echantillon : $S = \{(\varepsilon, \varepsilon), (aaa, aaa), (a, b), (aaaa, baaa)\}$
- Domaine : $a^{3n} + a^{3n+1}$



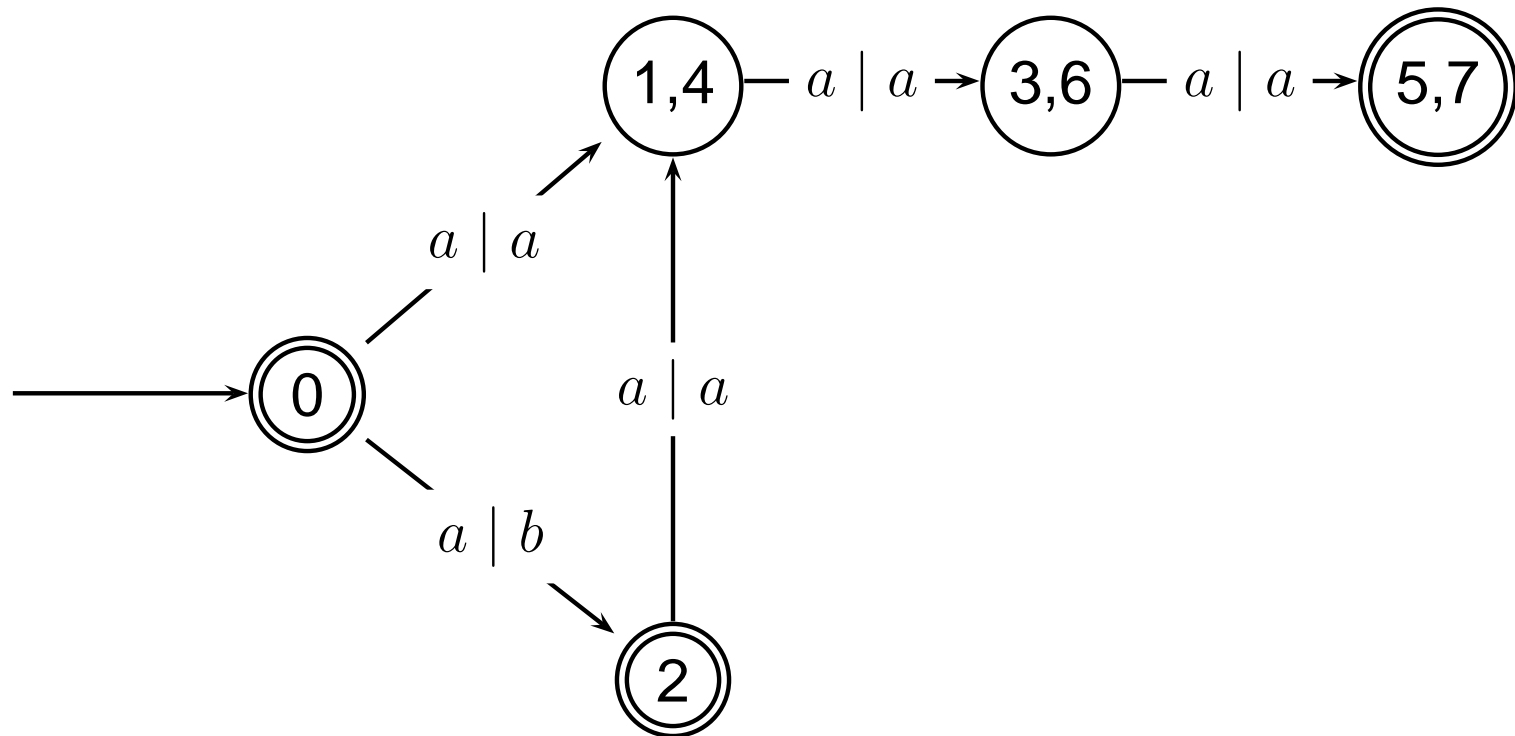
Prochaine fusion : 1 – 2

- Echantillon : $S = \{(\varepsilon, \varepsilon), (aaa, aaa), (a, b), (aaaa, baaa)\}$
- Domaine : $a^{3n} + a^{3n+1}$



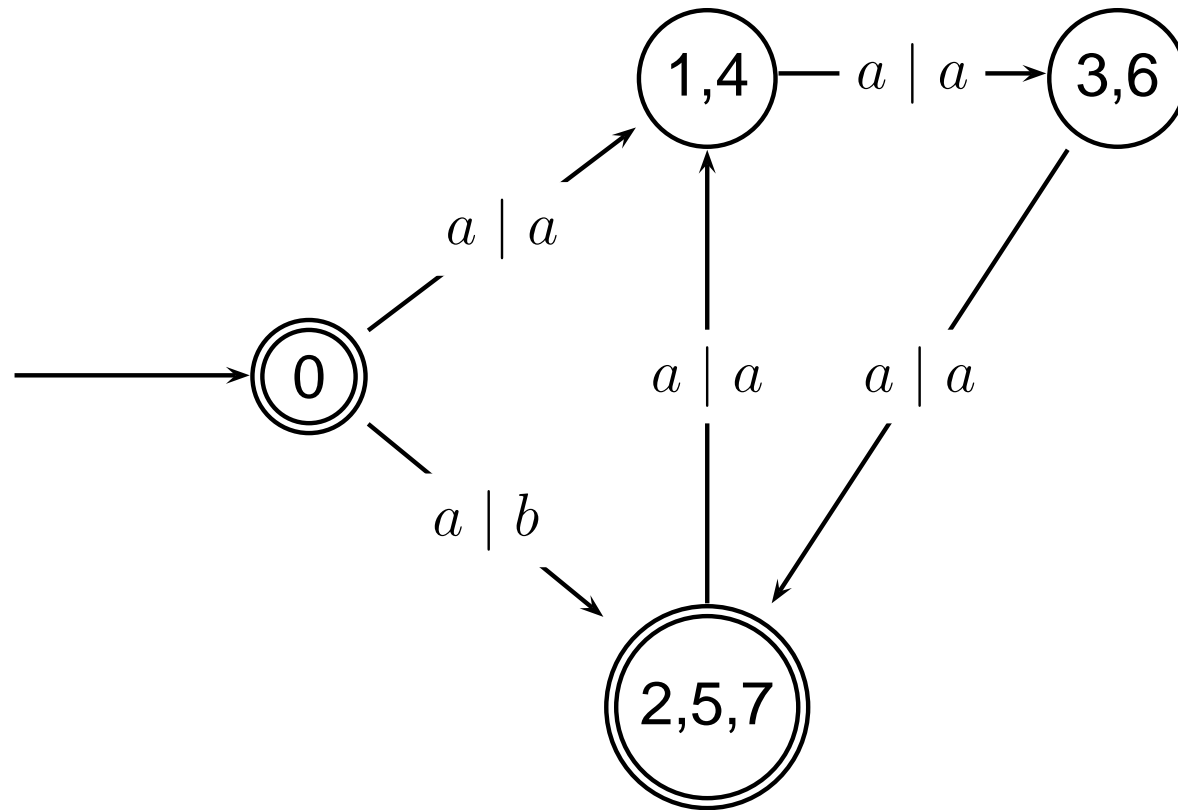
ECHEC (pas fonctionnel, et sort du domaine)

- Echantillon : $S = \{(\varepsilon, \varepsilon), (aaa, aaa), (a, b), (aaaa, baaa)\}$
- Domaine : $a^{3n} + a^{3n+1}$



Fusion 1 – 4 **Ok!**

- Echantillon : $S = \{(\varepsilon, \varepsilon), (aaa, aaa), (a, b), (aaaa, baaa)\}$
- Domaine : $a^{3n} + a^{3n+1}$



Fusion 2 – (5, 7) **Ok!**

MOSTRARE

Input : échantillon (arbre étiquetés) et domaine

1. construction de l'arbre préfixe (déterministe sur les couples entrées/sorties)
2. Tester toutes les fusions 'déterministes' possibles telles que :
 - l'automate courant reste fonctionnel
 - le langage 'support' reste dans le domaine

Sortie : Une relabelling rational function

relabelling rational functions are learnable from positive data and domain.
(in a “reasonable” variant of the Gold Model)

relabelling rational functions are learnable from positive data and domain.
(in a “reasonable” variant of the Gold Model)

J'aimerais bien en tout cas...

Résumé des questions principales

- équivalence requête / relabelling functions ?

Résumé des questions principales

- équivalence requête / relabelling functions ?
- toutes les 'relabelling functions' peuvent être mis sous la forme de transducteurs déterministes sur les couples entrées/sorties ?

Résumé des questions principales

- équivalence requête / relabelling functions ?
- toutes les 'relabelling functions' peuvent être mis sous la forme de transducteurs déterministes sur les couples entrées/sorties ?
- apprenabilité des relabelling rational functions ?

Résumé des questions principales

- équivalence requête / relabelling functions ?
- toutes les 'relabelling functions' peuvent être mis sous la forme de transducteurs déterministes sur les couples entrées/sorties ?
- apprenabilité des relabelling rational functions ?
- que se passe-t'il si le domaine est inconnu ?

Résumé des questions principales

- équivalence requête / relabelling functions ?
- toutes les 'relabelling functions' peuvent être mis sous la forme de transducteurs déterministes sur les couples entrées/sorties ?
- apprenabilité des relabelling rational functions ?
- que se passe-t'il si le domaine est inconnu ?
- ... ou *mal* connu ? (par exemple si on a un step-wise réversible plus important)