
Toward learning rational queries

A. Lemay







GRAPPA - Lille 3
INRIA MOSTRARE

1. Query : a definition
2. Link rational query \leftrightarrow tree transducer
3. Learning rational queries

Query : a definition

220 NEC items in All eBay

[All Items](#) **All items including Gallery preview** [Gallery Items](#)

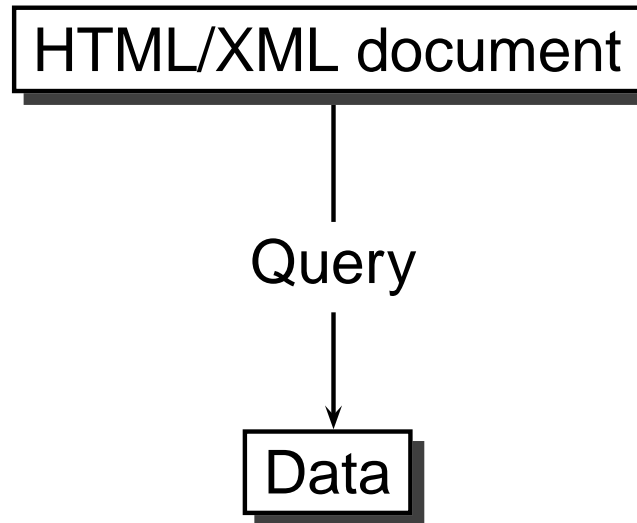
Status	Featured Items - Current	Price	Bids	Ends PDT
	 Qty 10 NEC Versa LX PII 300MHz/128M warranty	\$4400.00	-	Jul-17 09:04
	 NEC Pentium 133 Laptop Versa 6030H	\$51.00	14	Jul-13 07:14
	2-NEC Versa V/75~ ~complete w/ HD & AC (dutch)	\$100.00	2	Jul-14 07:12
	NEC Versa Laptop P-133/24MB/1.4GB/33.6kbp/TFT 	\$112.50	6	Jul-13 19:45

To find out how to be listed in this section and seen by thousands, please visit this link [Featured Auctions](#)

Query the web : Extract information on web pages / on XML document...

MOSTRARE

A query on HTML/XML documents

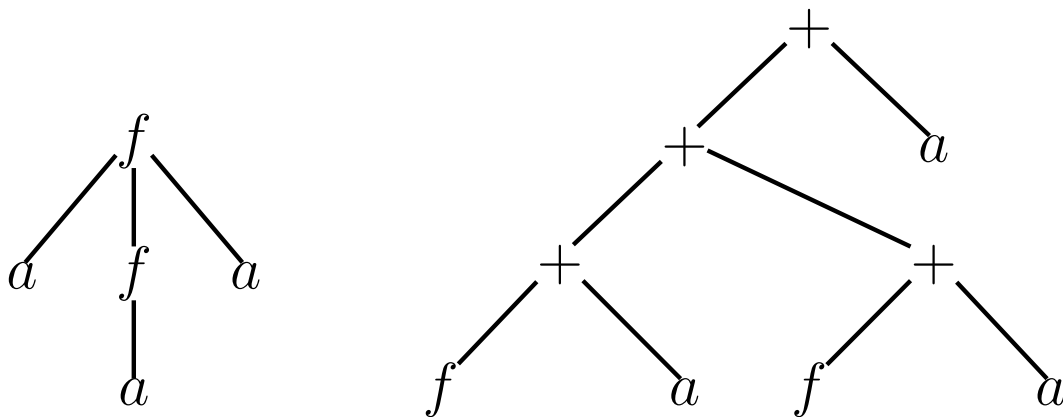


- An XML document : a tree on an unranked alphabet
- query : rational queries using step-wise automata

Example : (trees with a leaf of height exactly 2)

$A = \langle \Sigma, Q, F = \{q_0\}, \Delta \rangle$. $\Delta =$

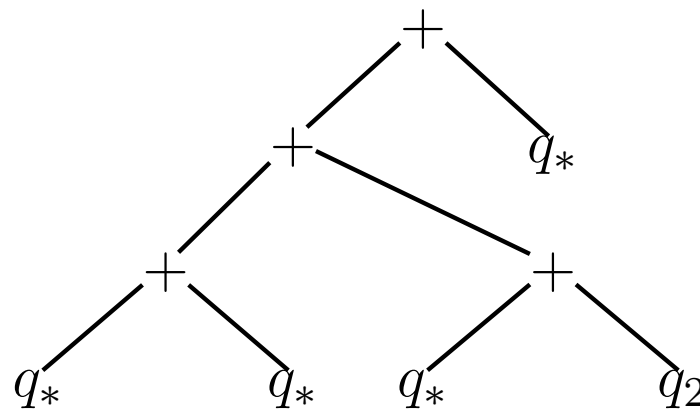
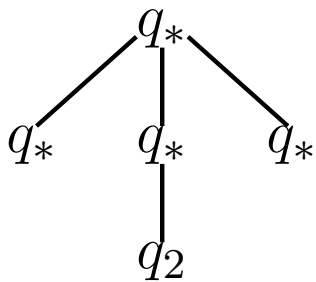
- $a \rightarrow q_2 ; a \rightarrow q_*, f \rightarrow q_*$
- $q_* + q_* \rightarrow q_* ; q_* + q_2 \rightarrow q_1 ; q_* + q_1 \rightarrow q_0 ; q_1 + q_* \rightarrow q_1 ; q_0 + q_* \rightarrow q_0$



Example : (trees with a leaf of height exactly 2)

$A = \langle \Sigma, Q, F = \{q_0\}, \Delta \rangle$. $\Delta =$

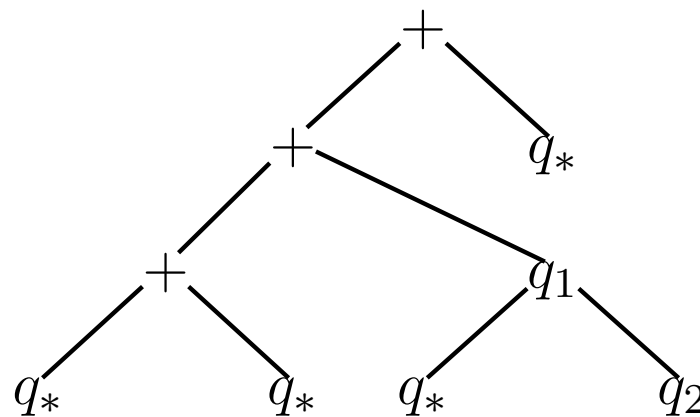
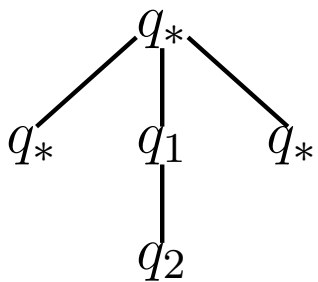
- $a \rightarrow q_2$; $a \rightarrow q_*$, $f \rightarrow q_*$
- $q_* + q_* \rightarrow q_*$; $q_* + q_2 \rightarrow q_1$; $q_* + q_1 \rightarrow q_0$; $q_1 + q_* \rightarrow q_1$; $q_0 + q_* \rightarrow q_0$



Example : (trees with a leaf of height exactly 2)

$A = \langle \Sigma, Q, F = \{q_0\}, \Delta \rangle$. $\Delta =$

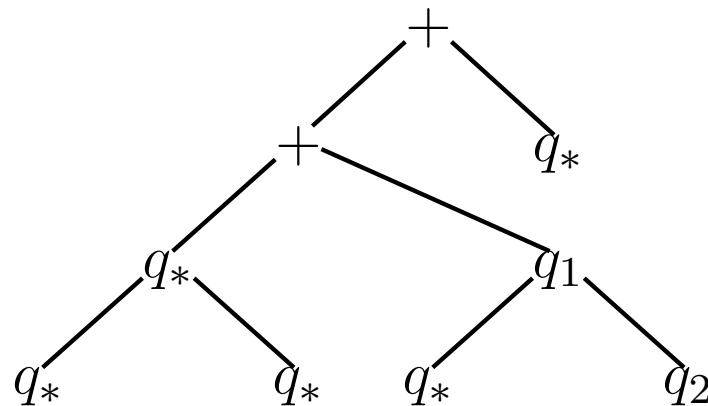
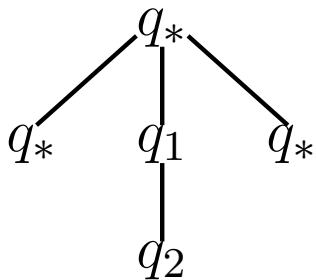
- $a \rightarrow q_2$; $a \rightarrow q_*$, $f \rightarrow q_*$
- $q_* + q_* \rightarrow q_*$; $q_* + q_2 \rightarrow q_1$; $q_* + q_1 \rightarrow q_0$; $q_1 + q_* \rightarrow q_1$; $q_0 + q_* \rightarrow q_0$



Example : (trees with a leaf of height exactly 2)

$A = \langle \Sigma, Q, F = \{q_0\}, \Delta \rangle$. $\Delta =$

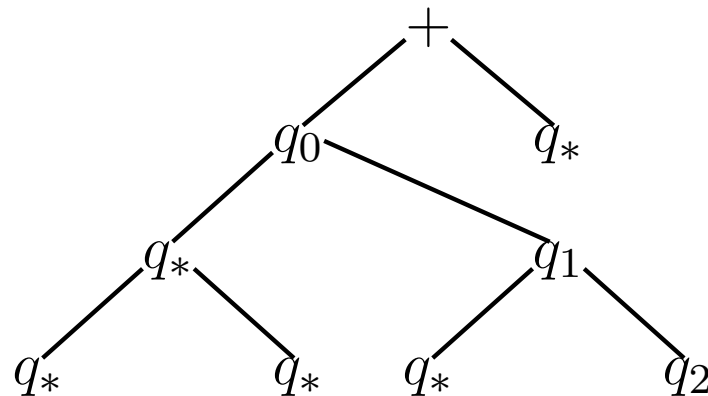
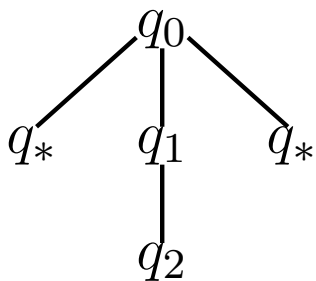
- $a \rightarrow q_2$; $a \rightarrow q_*$, $f \rightarrow q_*$
- $q_* + q_* \rightarrow q_*$; $q_* + q_2 \rightarrow q_1$; $q_* + q_1 \rightarrow q_0$; $q_1 + q_* \rightarrow q_1$; $q_0 + q_* \rightarrow q_0$



Example : (trees with a leaf of height exactly 2)

$A = \langle \Sigma, Q, F = \{q_0\}, \Delta \rangle$. $\Delta =$

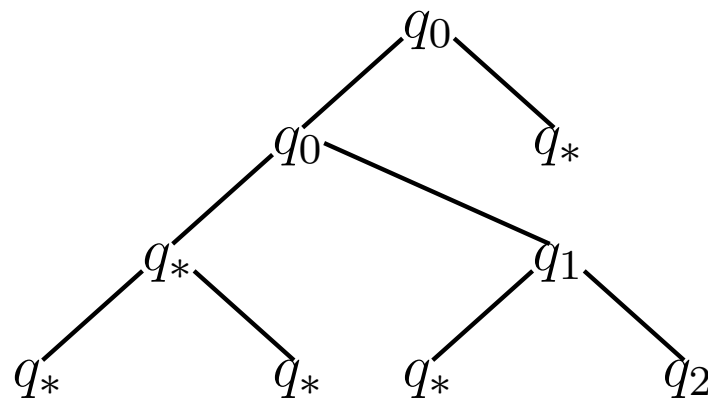
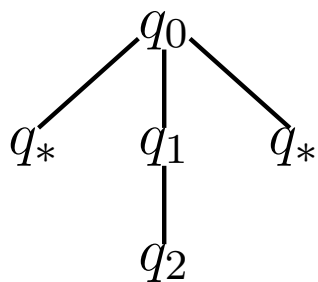
- $a \rightarrow q_2$; $a \rightarrow q_*$, $f \rightarrow q_*$
- $q_* + q_* \rightarrow q_*$; $q_* + q_2 \rightarrow q_1$; $q_* + q_1 \rightarrow q_0$; $q_1 + q_* \rightarrow q_1$; $q_0 + q_* \rightarrow q_0$



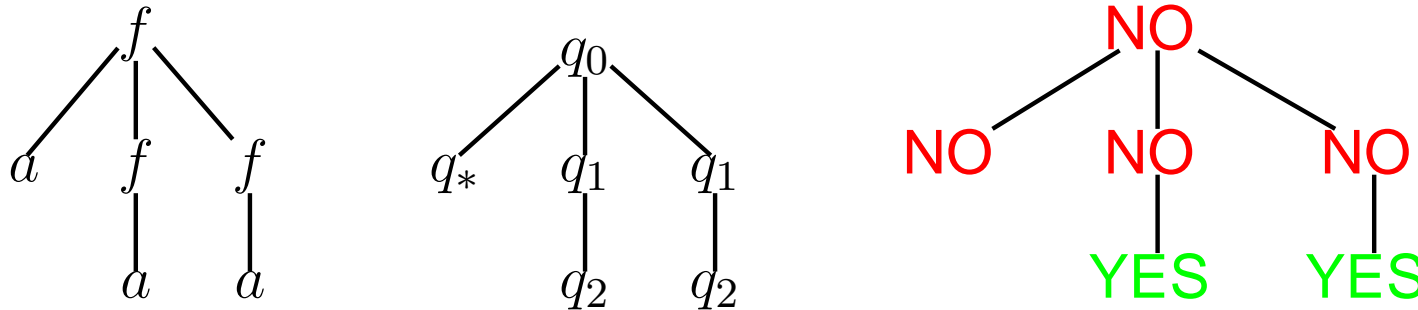
Example : (trees with a leaf of height exactly 2)

$A = \langle \Sigma, Q, F = \{q_0\}, \Delta \rangle$. $\Delta =$

- $a \rightarrow q_2$; $a \rightarrow q_*$, $f \rightarrow q_*$
- $q_* + q_* \rightarrow q_*$; $q_* + q_2 \rightarrow q_1$; $q_* + q_1 \rightarrow q_0$; $q_1 + q_* \rightarrow q_1$; $q_0 + q_* \rightarrow q_0$

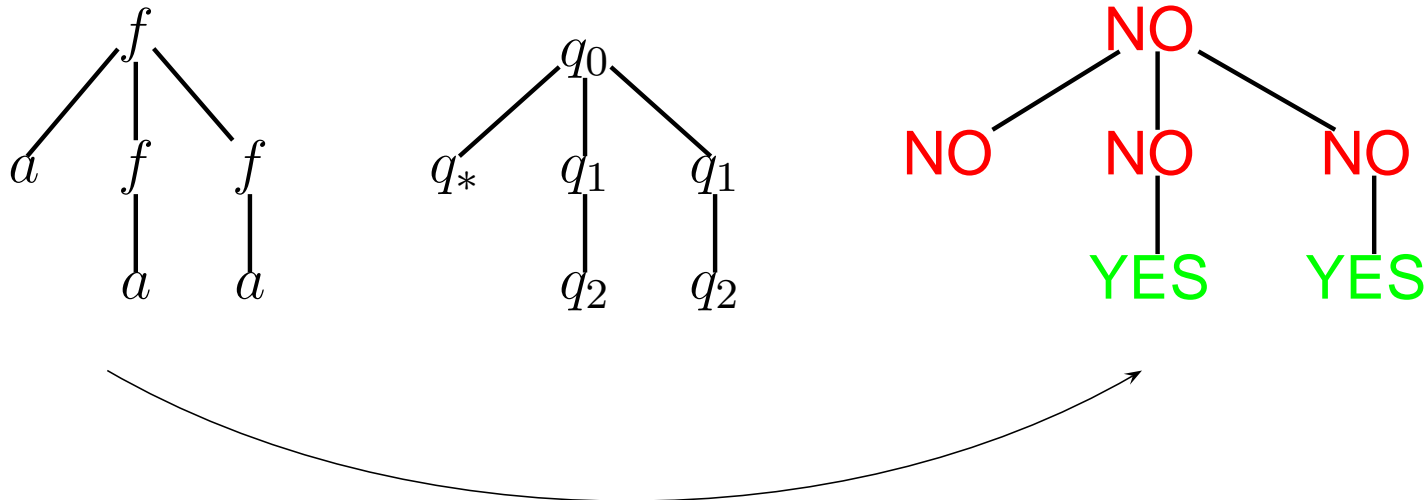


A *rational* query : a stepwise automaton + a set of states. Ex : $\langle A, \{q_2\} \rangle$



- Algebraic view of queries
- Works the same if A is a classical tree automaton (on $\Sigma \cup +$) on the construction tree
- Equivalent to a monadic DATALOG query
- Equivalent to MSO formula

A *rational* query : a stepwise automaton + a set of states. Ex : $\langle A, \{q_2\} \rangle$



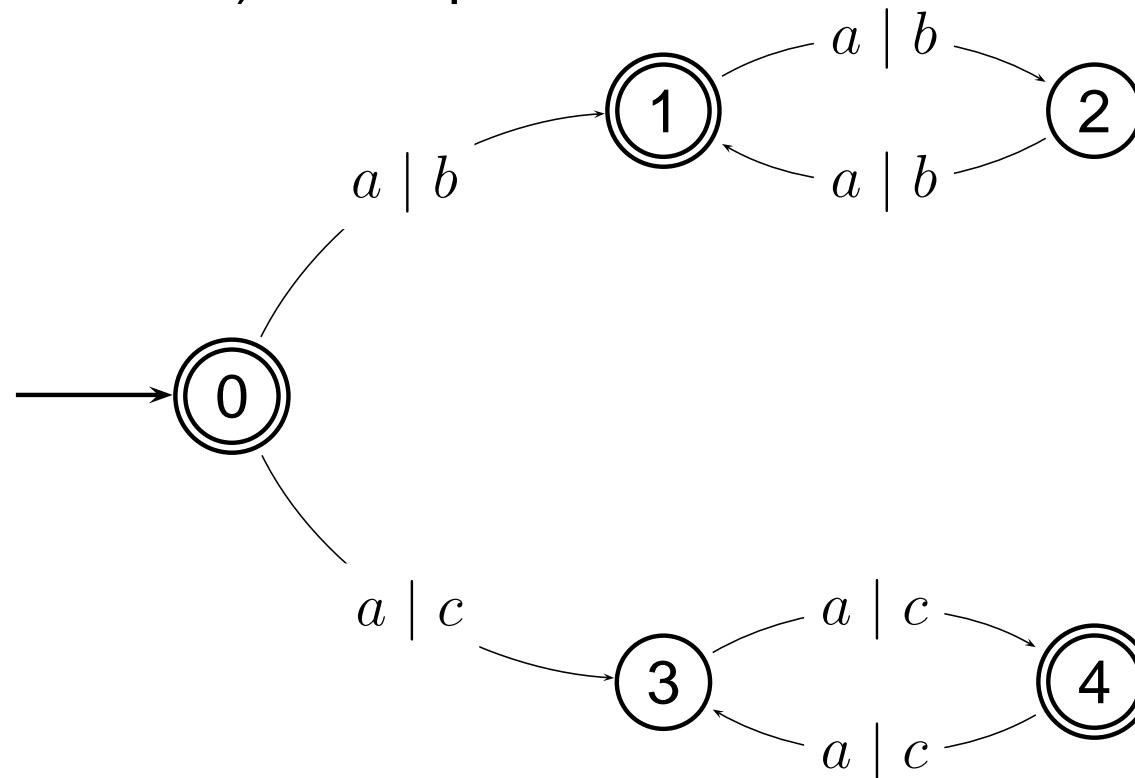
Transduction :

- relabelling (structure preserved)
- fonctionnal (**YES** for each node that can be output by the query)
- rational (can be done by a tree transducer)

rational query \leftrightarrow tree transducer

Relabelling rational function

Rational function (on words). Example :



- Rational transduction : word transformation using a rational transducer
- Functional : One input has exactly one output
- relabelling (here length-preserving) : each letter is transformed in exactly one letter

- *Relabelling tree transducer* : rules like

$f(q_1(x_1), \dots, q_n(x_n)) \rightarrow q(g(x_1, \dots, x_n))$ with $g \in \Sigma'$.

Ex : $A = \{\Sigma = \{a_0, f_2\}, \Sigma' = \{YES, NO\}, Q, q_f, \Delta\}$ avec Δ :

$a \rightarrow q_0(YES) \mid q_1(NO)$

$f(q_0, q_1) \rightarrow q_f(NO)$

$f(q_0, q_f) \rightarrow q_f(NO)$

$f(q_f, q_f) \rightarrow q_f(NO)$

Tree transducer that labels **YES** all the first child that are a leaf

“Equivalence” between :

- *rational queries* and
- *rational relabelling functional tree transducers*

Proof based on the fact that :

- each rational query can be done in two “deterministic” run ($\uparrow + \downarrow$)
- each relabelling rational function can be decomposed in two “deterministic” relabelling tree transducer

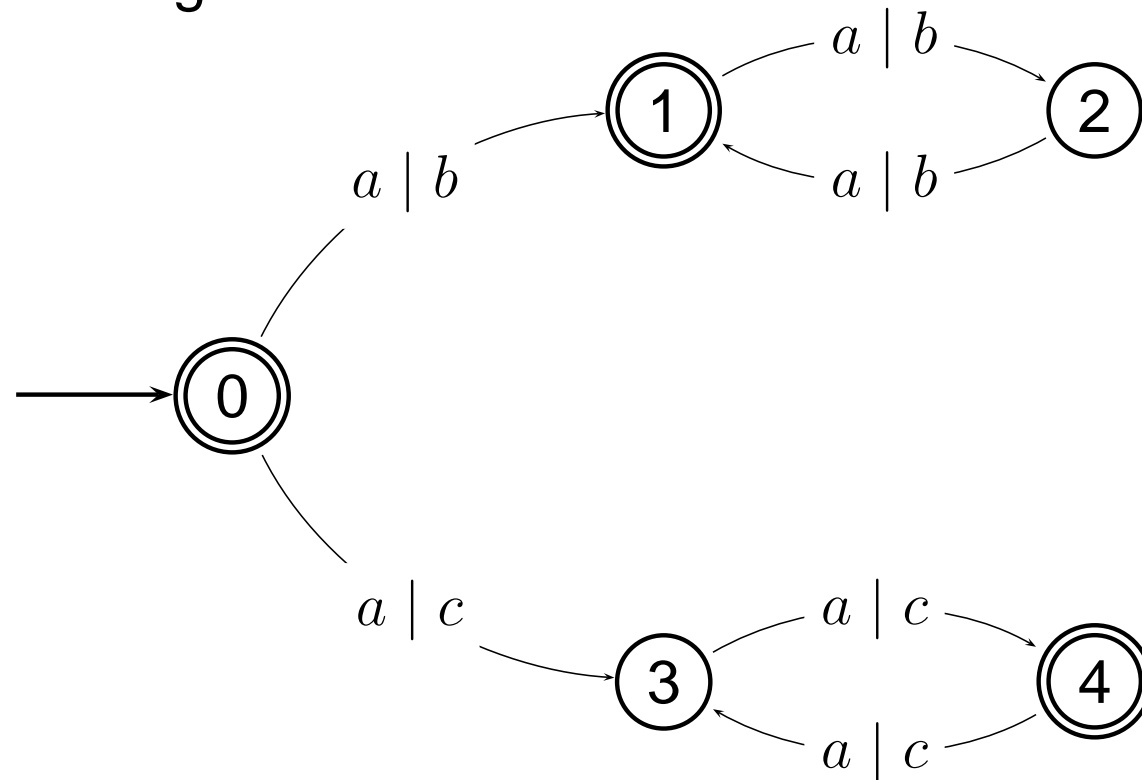
$$F_{RELAB} = D \uparrow_{RELAB} \circ D \downarrow_{RELAB}$$

Learning rational query

Result :

All rational function can be described by a transducer **deterministic** on couple input/output, **unambiguous** on input

- True in “length-preserving” word transducer



- Also true for relabelling rational tree function

Learning algorithm (in words)

In words : length preserving rational function :

Positive sample
(couple input/output)

Domain
(support language)

```
graph LR; A[Algorithm] --> T[Transducer]; P[Positive sample (couple input/output)] --> A; D[Domain (support language)] --> A;
```

Algorithm

Transducer

- Algorithm : (slight) variant of RPNI ([Oncina - Garcia 92])

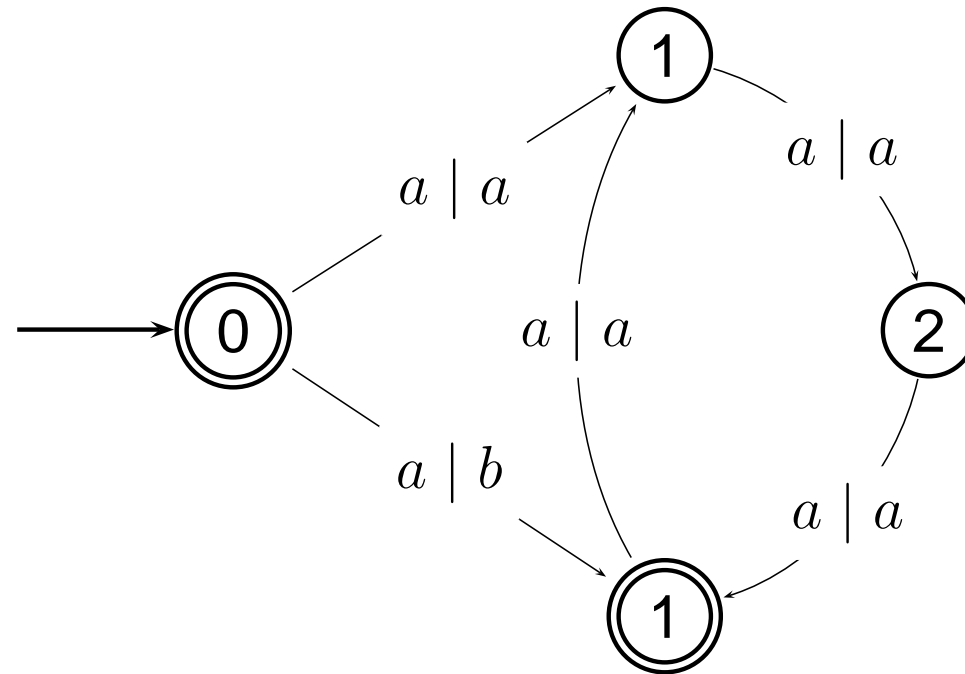
No negative examples but :

- functionality : badly labelled are negative ($(u, v) \in S \Rightarrow (u, v') \notin L$)
- inputs out of domain are negative

MOSTRARE

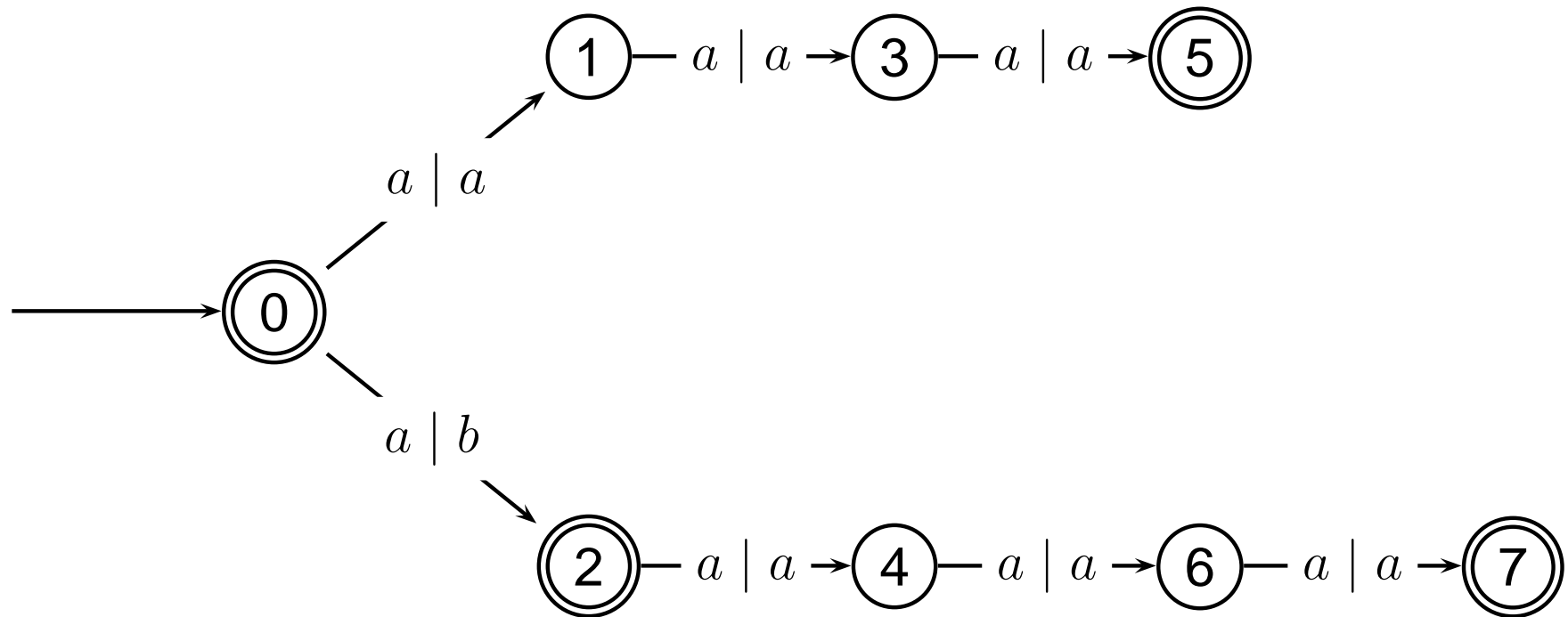
Target : $a^{3n} \rightarrow a^{3n}$
 $a^{3n+1} \rightarrow ba^{3n}$

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

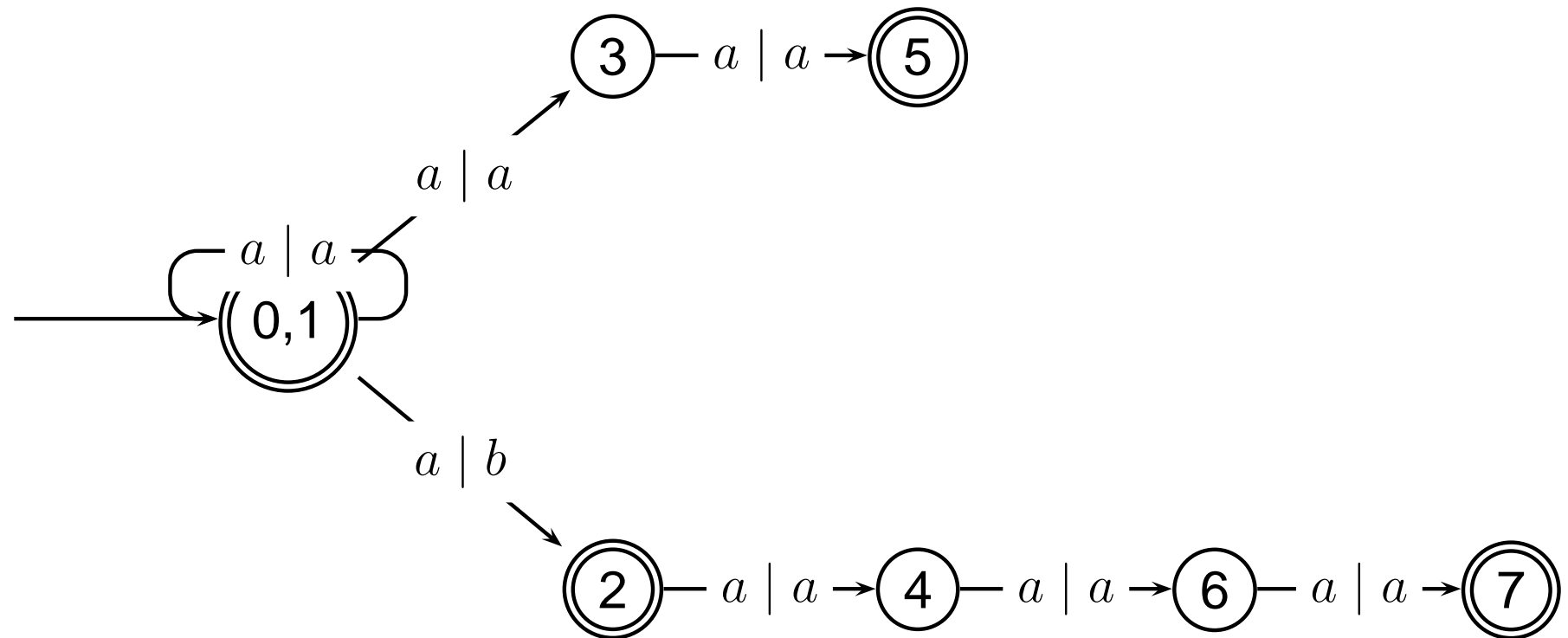


Example

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

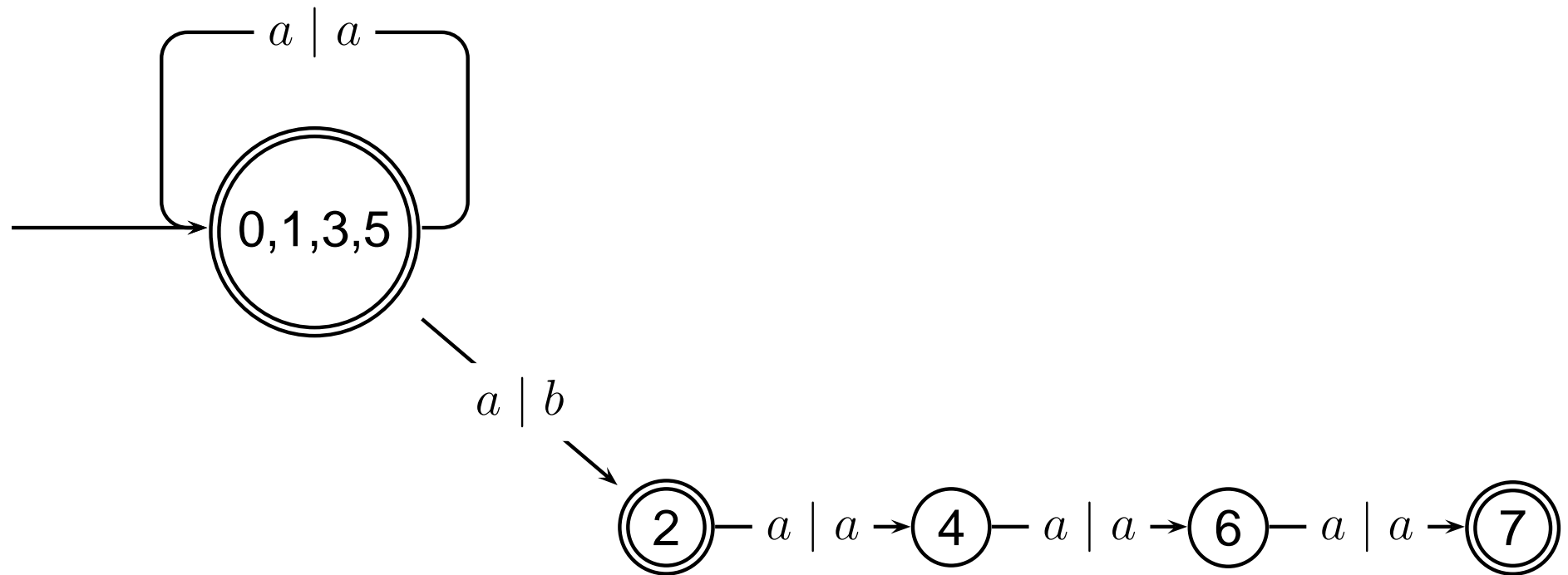


- Sample : $S = \{(\varepsilon, \varepsilon), (aaa, aaa), (a, b), (aaaa, baaa)\}$
- Domain : $a^{3n} + a^{3n+1}$



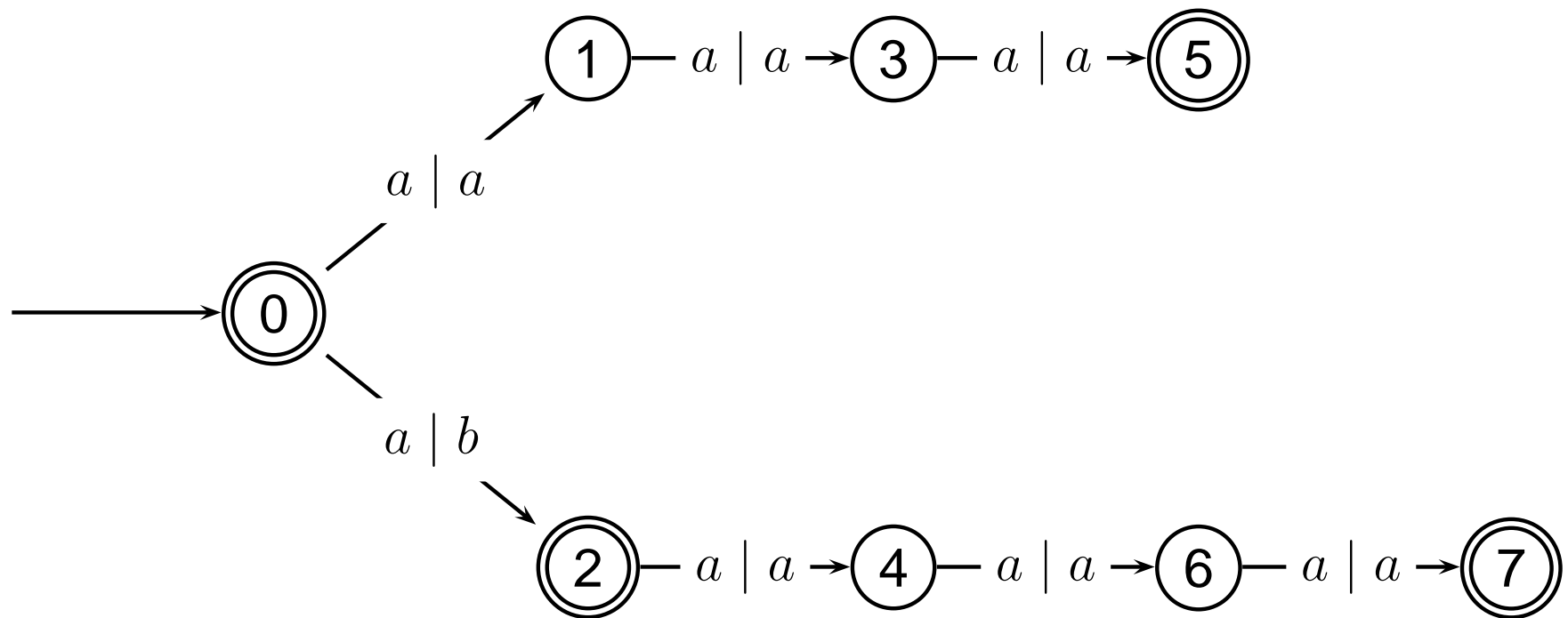
Fusion 0 – 1.

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



Deterministic merge rejected ! (non fonctionnal + out of domain)

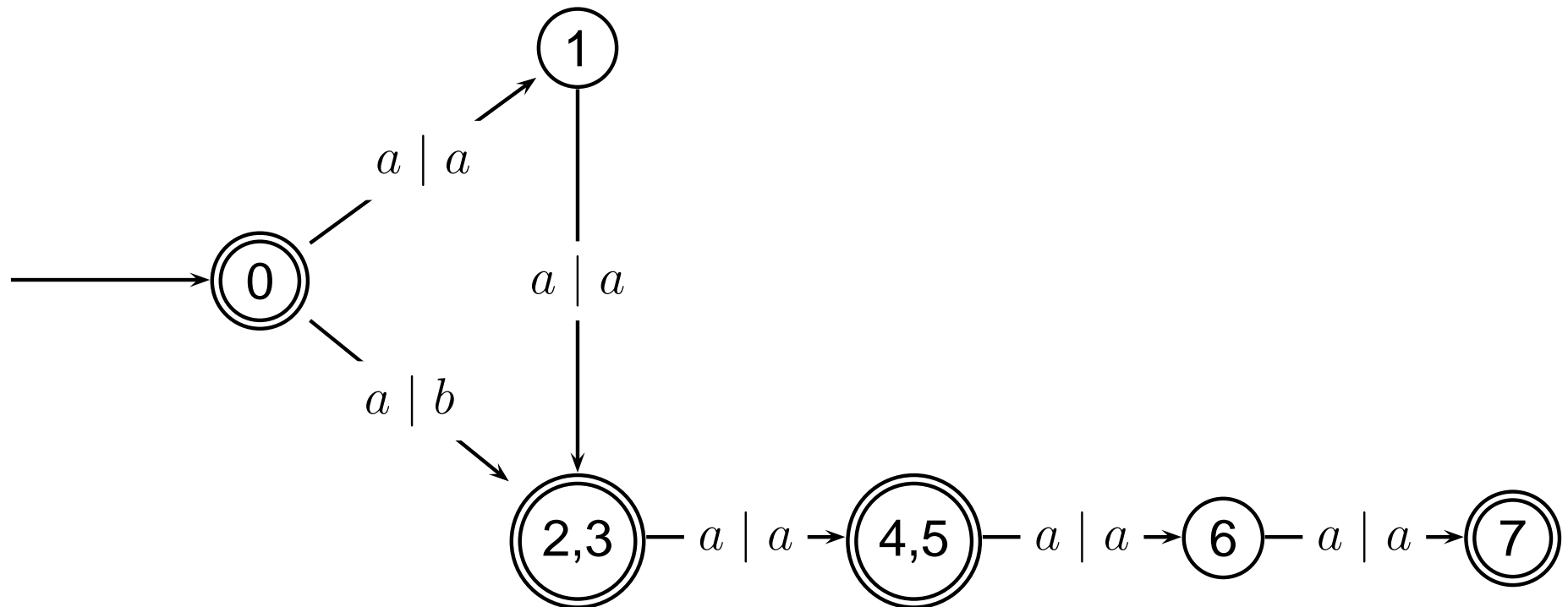
- Sample : $S = \{(\varepsilon, \varepsilon), (aaa, aaa), (a, b), (aaaa, baaa)\}$
- Domain : $a^{3n} + a^{3n+1}$



Next merge : 1 – 3

Example

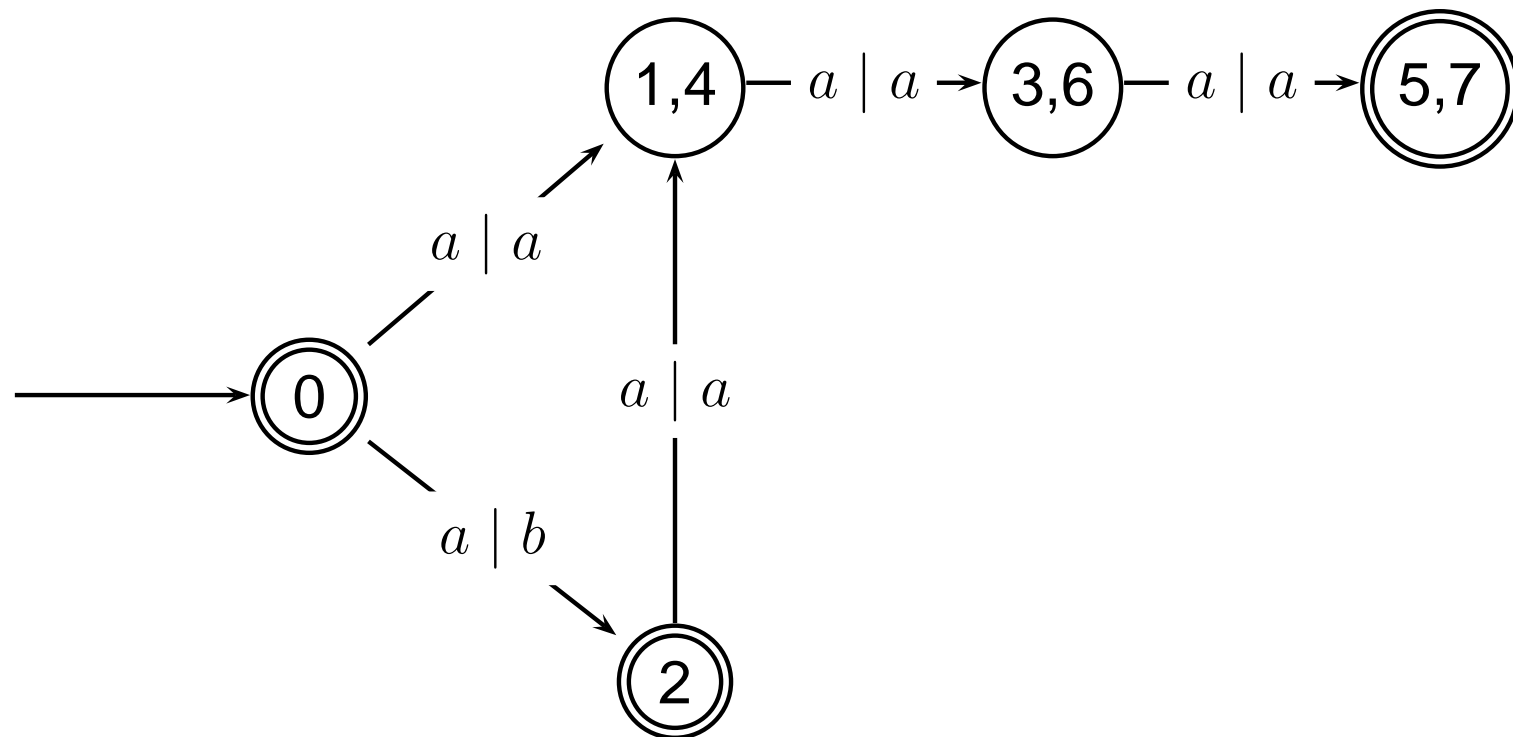
- Sample : $S = \{(\varepsilon, \varepsilon), (aaa, aaa), (a, b), (aaaa, baaa)\}$
- Domain : $a^{3n} + a^{3n+1}$



FAILED (not functional / out of domain)

MOSTRARE

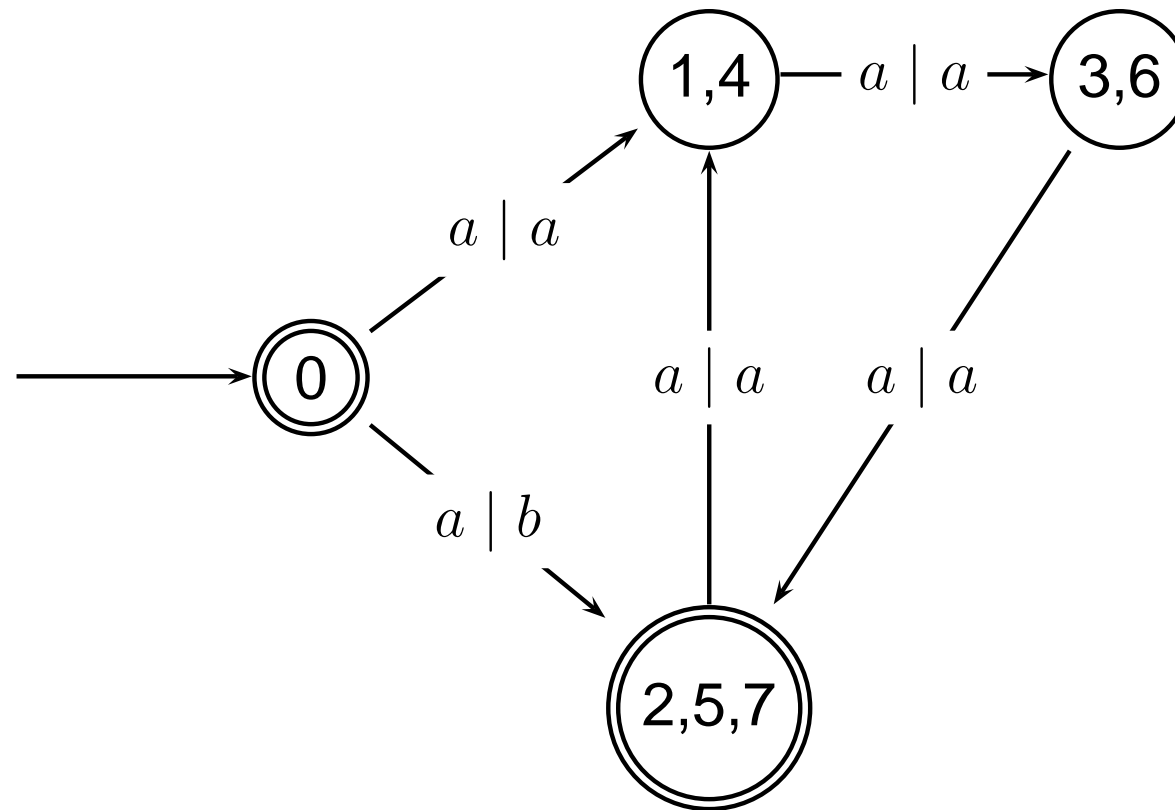
- Sample : $S = \{(\varepsilon, \varepsilon), (aaa, aaa), (a, b), (aaaa, baaa)\}$
- Domain : $a^{3n} + a^{3n+1}$



Merge 1 – 4 **Ok!**

MOSTRARE

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



Merge 2 – (5, 7) **Ok!**

MOSTRARE

Input : labelled sample and domain

1. prefix tree building (deterministic on couple input/output)
2. width first search of all couple of state. 'Deterministically' Merge them if
 - the current automaton is still fonctionnal
 - the support language is still in the domain

Output : a relabelling rational function

- Domain ?

Not necessary ? (without domain, some negatives are unavailable, but transducer learned *should* behave the same...)

- Functionality ?

Here : same as unambiguity on input.

Use of **Common Prefix** and **Common Suffix** relations ([Coste-Fredouille 2000])

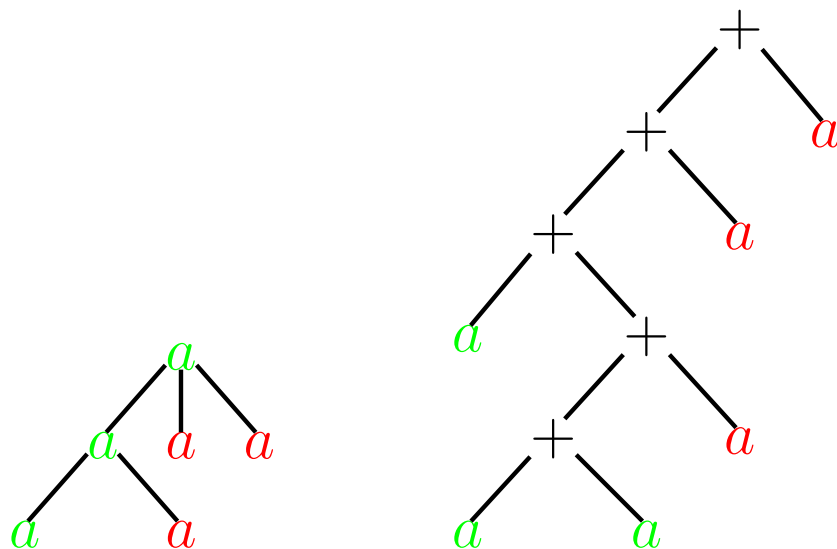
- Tree transducer ?

Extension of the algorithm (like [Oncinca-Garcia 93])

Target transducer :

label **YES** the first leaf and all its ancestor

- $a \rightarrow 0(\mathbf{YES})$, $a \rightarrow 1(\mathbf{NO})$
- $0 + 0 \rightarrow 2$, $0 + 2 \rightarrow 2$, $1 + 1 \rightarrow 1$, $2 + 1 \rightarrow 2$



(Note : here, only nodes are labeled)

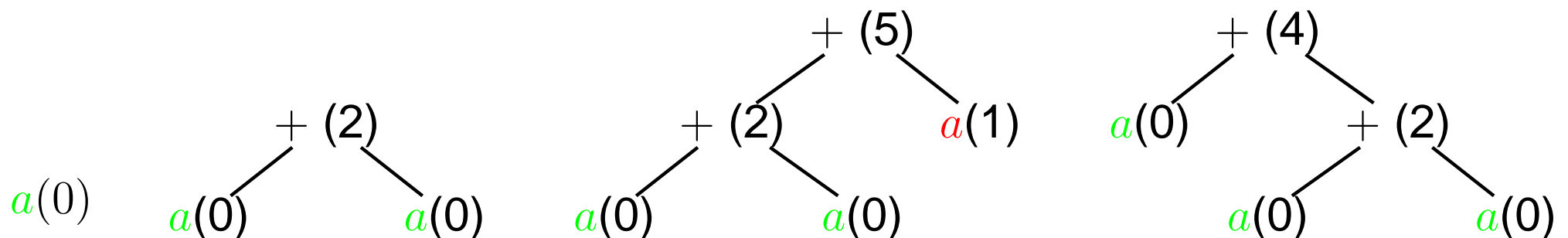
Characteristical sample :

a , $a(a)$, $a(a,a)$, $a(a(a))$, $a(a,a(a))$, $a(a(a), a)$, $a(a, a,a)$

1. Construction of the “Prefix Tree” :

- $a \rightarrow 0$ (YES), $a \rightarrow 1$ (NO)
- $0 + 0 \rightarrow 2$, $0 + 2 \rightarrow 4$, $1 + 1 \rightarrow 3$, $2 + 1 \rightarrow 5$, $2 + 3 \rightarrow 6$, $4 + 1 \rightarrow 8$, $5 + 1 \rightarrow 7$,

Example :



One state : One labelled subtree

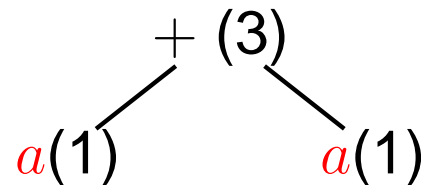
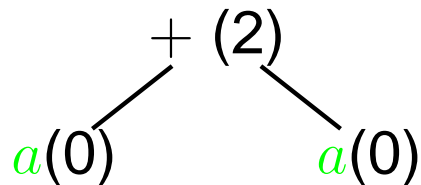
2. Construction of sets **Common Subtree** and **Common Context**

Common subtree : states that can be reached by the same tree labelled differently

$a(0)$

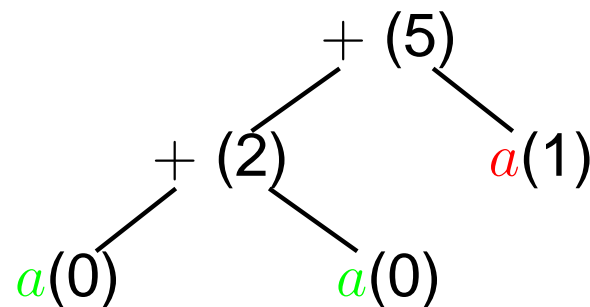
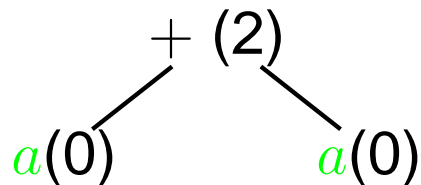
$a(1)$

$(0, 1) \in \text{Common Subtree}$



$(2, 3) \in \text{Common Subtree}$

Common context : states that can be reached by the same context (eventually) labelled differently



$(0, 2) \in \text{Common Context } (a(\cdot, a))$

...

Ideas :

- elements of Common Subtree unmergeable !

Ex : $(0, 1) \in \text{Common Subtree } a \rightarrow 0(\text{YES}), a \rightarrow 1(\text{NO})$

Algorithm (prototype)

Ideas :

- elements of Common Subtree unmergeable !

Ex : $(0, 1) \in \text{Common Subtree } a \rightarrow 0(\text{YES}), a \rightarrow 1(\text{NO})$

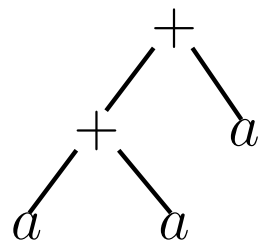
- if $(q, q') \in \text{Common Subtree}$ and $(q, q') \in \text{Common Subtree}$: ambiguity (i.e. lack of functionality !)

Ex : after merging 0 and 2

$$0 + 0 \rightarrow 2$$

$$2 + 1 \rightarrow 5$$

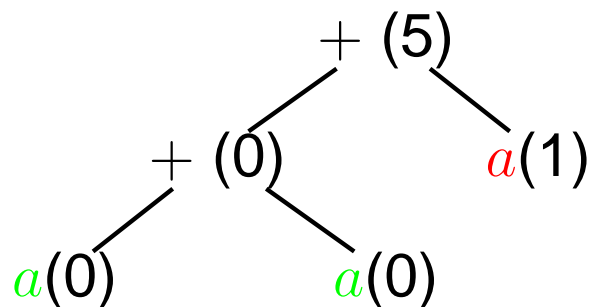
...



$$0 + 0 \rightarrow 0$$

$$0 + 1 \rightarrow 5$$

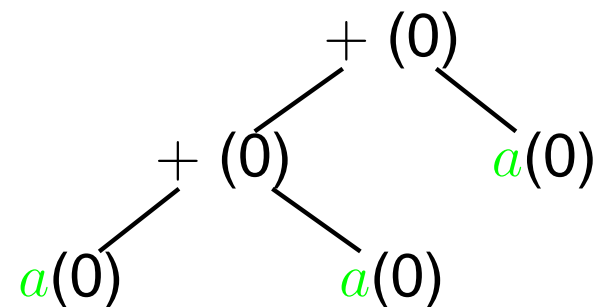
...



Common Subtree

$(0, 5)$

...



Common Context

$(0, 5)$

...

- Merge $(0, 1)$? No ! $(0, 1)$ in Common Subtree

- Merge $(0, 1)$? No ! $(0, 1)$ in Common Subtree
- Merge $(0, 2)$? No ! it leads to $(0, 5) \in \text{Common Prefix} \cap \text{Common Suffix}$

- Merge (0, 1) ? No ! (0, 1) in Common Subtree
- Merge (0, 2) ? No ! it leads to (0, 5) \in Common Prefix \cap Common Suffix
- ... Merge (1, 3) ? YES ! no problem...

$$\begin{array}{ccc} 1 + 1 \rightarrow 3 & & 1 + 1 \rightarrow 1 \\ 2 + 1 \rightarrow 5 & \Rightarrow & 2 + 1 \rightarrow 5 \\ 2 + 3 \rightarrow 6 & & 2 + 1 \rightarrow 6 \\ \dots & & \dots \end{array} \Rightarrow \begin{array}{ccc} & & 1 + 1 \rightarrow 1 \\ & & 2 + 1 \rightarrow 5 \\ & & \dots \end{array}$$

Merge (1, 3) and (5, 6)

- Merge (0, 1) ? No ! (0, 1) in Common Subtree
- Merge (0, 2) ? No ! it leads to (0, 5) \in Common Prefix \cap Common Suffix
- ... Merge (1, 3) ? YES ! no problem...

$$\begin{array}{ccc} 1 + 1 \rightarrow 3 & & 1 + 1 \rightarrow 1 \\ 2 + 1 \rightarrow 5 & \Rightarrow & 2 + 1 \rightarrow 5 \\ 2 + 3 \rightarrow 6 & & 2 + 1 \rightarrow 6 \\ \dots & & \dots \end{array} \Rightarrow \begin{array}{ccc} & & 1 + 1 \rightarrow 1 \\ & & 2 + 1 \rightarrow 5 \\ & & \dots \end{array}$$

Merge (1, 3) and (5, 6)

- ... Merge (2, 4) (implies Merge (5, 8))

- Merge (0, 1) ? No ! (0, 1) in Common Subtree
- Merge (0, 2) ? No ! it leads to (0, 5) \in Common Prefix \cap Common Suffix
- ... Merge (1, 3) ? YES ! no problem...

$$\begin{array}{ccc} 1 + 1 \rightarrow 3 & & 1 + 1 \rightarrow 1 \\ 2 + 1 \rightarrow 5 & \Rightarrow & 2 + 1 \rightarrow 5 \\ 2 + 3 \rightarrow 6 & & 2 + 1 \rightarrow 6 \\ \dots & & \dots \end{array} \Rightarrow \begin{array}{ccc} & & 1 + 1 \rightarrow 1 \\ & & 2 + 1 \rightarrow 5 \\ & & \dots \end{array}$$

Merge (1, 3) and (5, 6)

- ... Merge (2, 4) (implies Merge (5, 8))
- then merge (5, 2) (implies (7, 2) and (8, 2))

Final Automaton

$$a \rightarrow 0(\text{YES}), a \rightarrow 1(\text{NO}), 0 + 0 \rightarrow 2, 0 + 2 \rightarrow 2, 1 + 1 \rightarrow 1, 2 + 1 \rightarrow 2$$

Input : Labeled sample

1. Build “prefix tree”
2. Build “Common Subtree” and “Common Context” relations
3. **Begin Loop** : Width search of couple of states (q, q') :

IF (q, q') are not in Common Subtree **Then**

- “deterministically” merge q and q'
- update “Common Subtree” and “Common Context” relations
- if there are no couple of states both in “Common Subtree” and “Common Context” then validate the merge

End If

End Loop

Output : a relabelling functional tree-transducer (the target ?)

MOSTRARE

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

rational queries are learnable from positive data.
(in a “reasonable” variant of the Gold Model*)

* :

- From positive labeled sample
- in the limit, only examples of the 'domain' will be observed
- in the limit, output query behave correctly on the domain

(hoped) **Results**

- equivalence rational queries / relabelling functional tree transducer
- relabelling functional tree transducer : unambiguous (on input)
“deterministic” (on input/output) transducer
- “deterministic” (on input/output) tree transducer are learnable from positive and negative data (RPNI)
- (?) relabelling functional tree transducer are learnable from positive data

Questions

- Is a knowledge on the domain really useful ? (personal opinion : No, but could help ?)
- complexity issues (esp. on size of characteristic sample)
- Number of examples necessary to learn a query ok ?
- Experimentation ?
- Adapt the learning protocol to reduce the number of example (queries to the user ?)