From Two-way to One-way Finite State Transducers: a Shepherdson's Approach

Benjamin Monmege, Pierre-Alain Reynier
and **Jean-Marc Talbot**

LIF, Aix-Marseille

DELTA Kick-off - Feb. 9-10

Outline

1. From two-way to universal transducer
2. Normalizing universal transducer
3. Deciding one-wayness of universal transducers

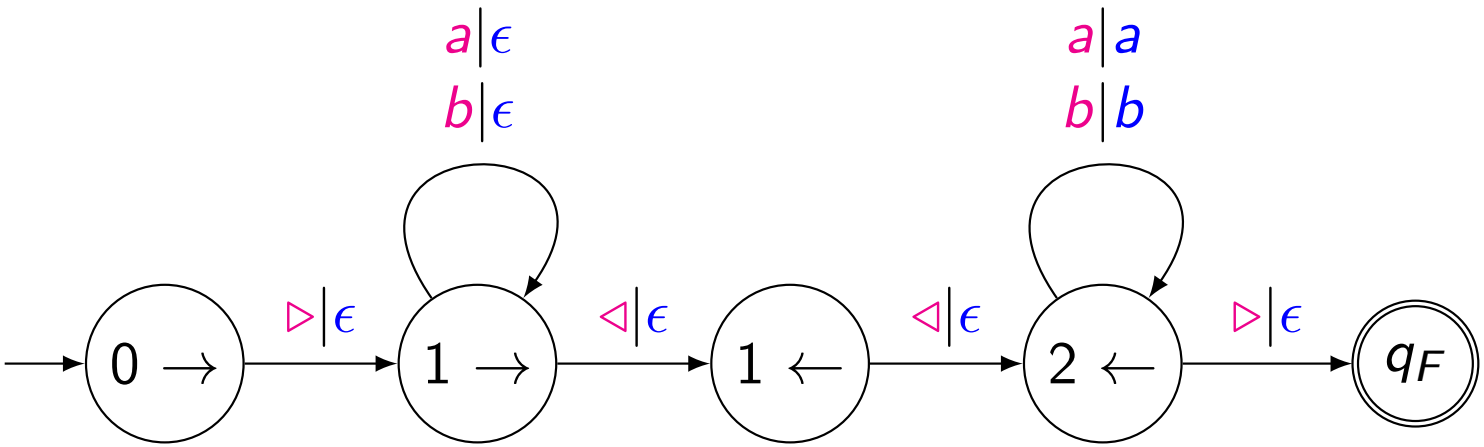
Outline

1 - From two-way to universal transducer

Deterministic Two-way Transducers: Example 1

Reverse : $\triangleright\Sigma^*\triangleleft \rightarrow \Sigma^*$

$\triangleright a_1 \ldots a_n\triangleleft \rightarrow a_n \ldots a_1$

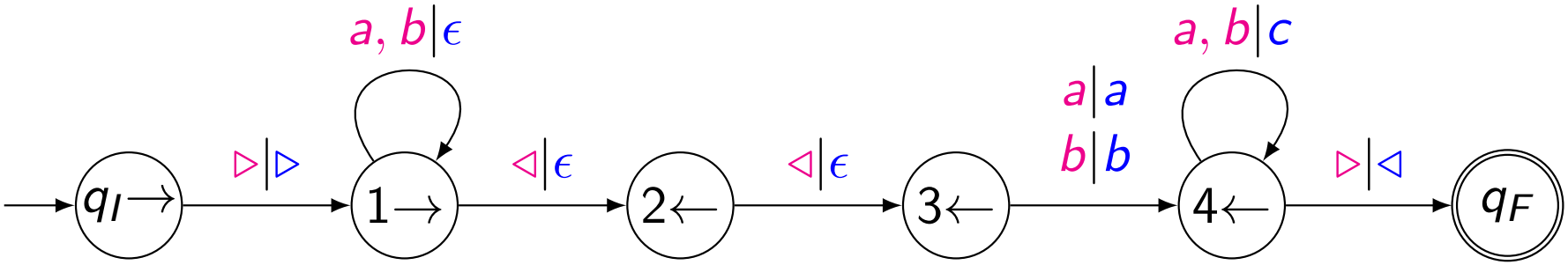


Deterministic Two-way Transducers: Example 2

LastFirst : $\triangleright\{a,b\}^+\triangleleft \rightarrow \triangleright\{a,b,c\}^+\triangleleft$

$\triangleright u\sigma\triangleleft \rightarrow \triangleright\sigma c^{|u|}\triangleleft$

with $\sigma \in \{a,b\}$



The One-wayness Problem of D2FST

Definition

A D2FST is *one-way definable* if there exists a equivalent one-way (functional) transducer.

One-way : directions restricted to $\rightarrow$

Functional =

deterministic

+

look-ahead (finite information on the suffix of the input)

Note that

- Reverse is not one-way definable
- LastFirst is one-way definable

The One-wayness Problem of D2FST

Definition

A D2FST is *one-way definable* if there exists a equivalent one-way (functional) transducer.

One-way : directions restricted to $\rightarrow$

Functional $=$

deterministic

$+$

look-ahead (finite information on the suffix of the input)

Note that

- Reverse is not one-way definable
- LastFirst is one-way definable

The One-wayness Problem of D2FST

Definition

A D2FST is *one-way definable* if there exists a equivalent one-way (functional) transducer.

One-way : directions restricted to $\rightarrow$

Functional $=$

deterministic

$+$

look-ahead (finite information on the suffix of the input)

Note that

- Reverse is not one-way definable
- LastFirst is one-way definable

Universal Transducers

Definition

Universal transducer = word-to-word finite state transducers that are deterministic for some (co-deterministic) look-ahead automaton

aka Deterministic top-down tree-to-string transducers with look-ahead and monadic inputs (see Maneth-Engelfriet)

Transition rules for some universal transducer $\mathcal{G}$ with look-ahead automaton B:

$$q(\sigma) \xrightarrow{p} v_0 q_1 v_1 \dots q_n v_n$$

$\sigma \in \Sigma$, $v_0, \dots v_n \in \Delta^*$, States $q, q_1 \dots q_n$ of $\mathcal{G}$, a state p from B

+ some initialization rules $\tau_I : p \to v_0 q_1 v_1 \dots q_n v_n$.

Universal Transducers

Definition

Universal transducer = word-to-word finite state transducers that are deterministic for some (co-deterministic) look-ahead automaton

aka Deterministic top-down tree-to-string transducers with look-ahead and monadic inputs (see Maneth-Engelfriet)

Transition rules for some universal transducer $\mathcal{G}$ with look-ahead automaton B:

$$q(\sigma) \xrightarrow{p} v_0 q_1 v_1 \dots q_n v_n$$

$\sigma \in \Sigma$, $v_0, \dots v_n \in \Delta^*$, States $q, q_1 \dots q_n$ of $\mathcal{G}$, a state p from B

+ some initialization rules $\tau_I : p \to v_0 q_1 v_1 \dots q_n v_n$.

Universal Transducers

Definition

Universal transducer = word-to-word finite state transducers that are deterministic for some (co-deterministic) look-ahead automaton

aka Deterministic top-down tree-to-string transducers with look-ahead and monadic inputs (see Maneth-Engelfriet)

Transition rules for some universal transducer $\mathcal{G}$ with look-ahead automaton B:

$$q(\sigma) \xrightarrow{p} v_0 q_1 v_1 \dots q_n v_n$$

$\sigma \in \Sigma$, $v_0, \dots v_n \in \Delta^*$, States $q, q_1 \dots q_n$ of $\mathcal{G}$, a state p from B

+ some initialization rules $\tau_I : p \to v_0 q_1 v_1 \dots q_n v_n$.

Universal Transducers

Definition

Universal transducer = word-to-word finite state transducers that are deterministic for some (co-deterministic) look-ahead automaton

aka Deterministic top-down tree-to-string transducers with look-ahead and monadic inputs (see Maneth-Engelfriet)

Transition rules for some universal transducer $\mathcal{G}$ with look-ahead automaton B:

$$q(\sigma) \xrightarrow{p} v_0 q_1 v_1 \dots q_n v_n$$

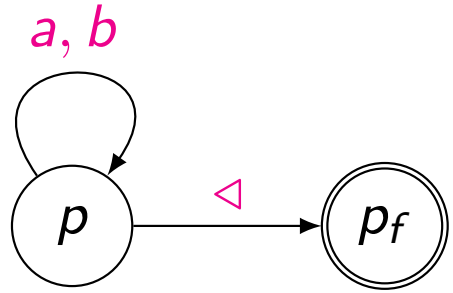
$\sigma \in \Sigma$, $v_0, \dots v_n \in \Delta^*$, States $q, q_1 \dots q_n$ of $\mathcal{G}$, a state p from B

+ some initialization rules $\tau_I : p \to v_0 q_1 v_1 \dots q_n v_n$.

Universal Transducers: Semantics by Example

Reverse :

- Look-ahead automaton



- A universal transducer $\mathcal{G}$

$$\tau_l(p) = q$$

$$q(a) \xrightarrow{p} q\,a$$
$$q(b) \xrightarrow{p} q\,b$$
$$q(\triangleleft) \longrightarrow \epsilon$$

Semantics: $\mathcal{G}(\triangleright aabba\triangleleft) = [\![q]\!]_p(aabba\triangleleft) = abbaa$

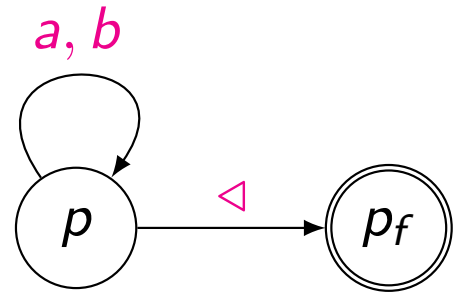
$$(aabba\triangleleft, q, p) \vdash (abba\triangleleft, qa, p) \vdash (bba\triangleleft, qaa, p) \vdash (ba\triangleleft, qbaa, p)$$
$$\vdash (a\triangleleft, qbbaa, p) \vdash (\triangleleft, qabbaa, p) \vdash (\epsilon, abbaa, p_F)$$

At each step, the suffix of the input word belongs to $L(p)$

Universal Transducers: Semantics by Example

Reverse :

- Look-ahead automaton



- A universal transducer $\mathcal{G}$

$$\tau_l(p) = q$$

$$q(a) \xrightarrow{p} q\,a$$
$$q(b) \xrightarrow{p} q\,b$$
$$q(\triangleleft) \longrightarrow \epsilon$$

Semantics: $\mathcal{G}(\triangleright aabba\triangleleft) = [\![q]\!]_p(aabba\triangleleft) = abbaa$

$$(aabba\triangleleft, q, p) \vdash (abba\triangleleft, qa, p) \vdash (bba\triangleleft, qaa, p) \vdash (ba\triangleleft, qbaa, p)$$

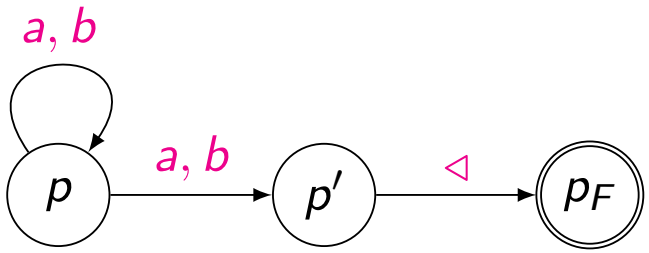
$$\vdash (a\triangleleft, qbbaa, p) \vdash (\triangleleft, qabbaa, p) \vdash (\epsilon, abbaa, p_F)$$

At each step, the suffix of the input word belongs to $L(p)$

Universal Transducers: Semantics by Example (II)

LastFirst : $\triangleright\{a,b\}^+\triangleleft \rightarrow \triangleright\{a,b,c\}^+\triangleleft$ with $\sigma \in \{a,b\}$

$\triangleright u\sigma\triangleleft \rightarrow \triangleright\sigma c^{|u|}\triangleleft$



$$\tau_I(p) = \triangleright q \triangleleft$$

$$q(a) \xrightarrow{p} q\,c \qquad q(b) \xrightarrow{p} q\,c$$
$$q(a) \xrightarrow{p'} q'\,a \qquad q(b) \xrightarrow{p'} q'\,b$$
$$q'(\triangleleft) \longrightarrow \epsilon$$

$$[\![q]\!]_p = \{[\![q]\!]_p(u) \mid u \in L(p)\} = ac^* + bc^* \qquad [\![q']\!]_{p'} = \epsilon$$

From D2FST to Universal Transducers

The (co-deterministic) look-ahead construction

Left-left traversals on u in $A = (q, q')$ if u admits a left-left computation in the D2FST A from $(q, \rightarrow)$ to $(q', \leftarrow)$

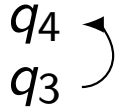
$\mathbb{T}$: the set of the finite set of all left-left traversals

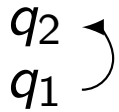
- The set of states of the look-ahead automaton is $\mathbb{T}$
- Transitions are defined from $\mathbb{T} \times \Sigma \times \mathbb{T}$

From D2FST to Universal Transducers

The (co-deterministic) look-ahead construction

Traversals $T = \{(q_1, q_2), (q_3, q_4)\}$





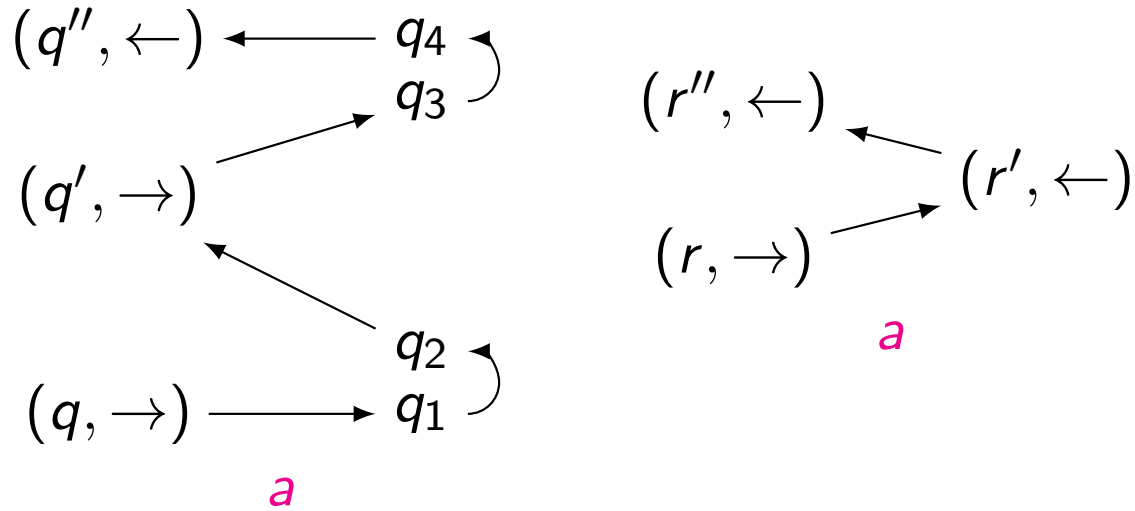
Transitions for $a = \left\{ \begin{array}{ll} (q, \rightarrow, a, q_1, \rightarrow, v_0) & (q_2, \leftarrow, a, q', \rightarrow, v_1) \\ (q', \rightarrow, a, q_3, \rightarrow, v_2) & (q_4, \leftarrow, a, q'', \rightarrow, v_3) \\ \\ (r, \rightarrow, a, r', \leftarrow, v_4) & (r', \leftarrow, a, r'', \leftarrow v_5) \end{array} \right.$

New traversal $T' = \{(q, q''), (r, r'')\}$ and a transition (T', a, T) in the look-ahead.

From D2FST to Universal Transducers

The (co-deterministic) look-ahead construction

Traversals $T = \{(q_1, q_2), (q_3, q_4)\}$



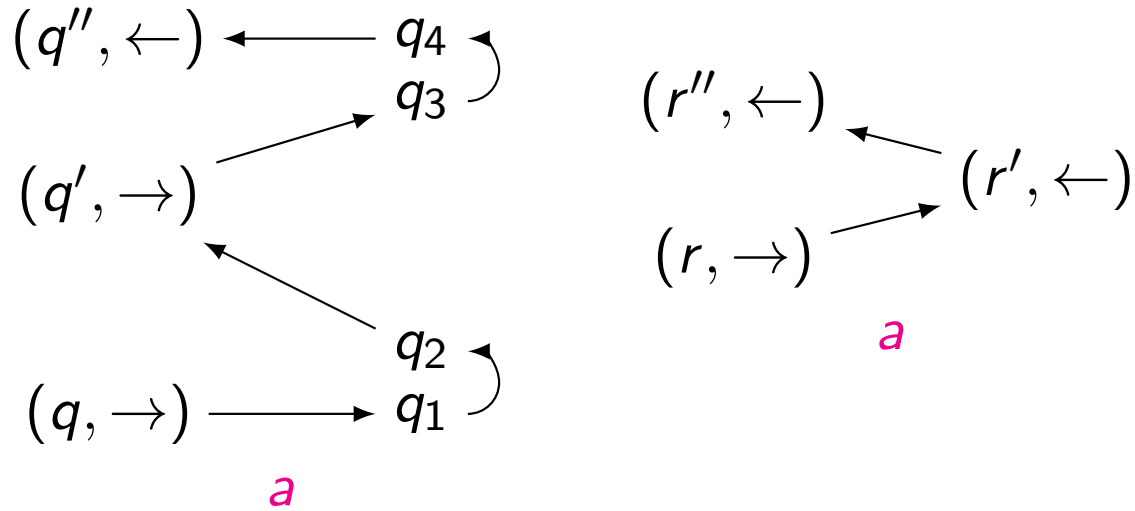
Transitions for $a = \left\{ \begin{array}{ll} (q, \rightarrow, a, q_1, \rightarrow, v_0) & (q_2, \leftarrow, a, q', \rightarrow, v_1) \\ (q', \rightarrow, a, q_3, \rightarrow, v_2) & (q_4, \leftarrow, a, q'', \rightarrow, v_3) \\ \\ (r, \rightarrow, a, r', \leftarrow, v_4) & (r', \leftarrow, a, r'', \leftarrow v_5) \end{array} \right.$

New traversal $T' = \{(q, q''), (r, r'')\}$ and a transition (T', a, T) in the look-ahead.

From D2FST to Universal Transducers

The (co-deterministic) look-ahead construction

Traversals $T = \{(q_1, q_2), (q_3, q_4)\}$



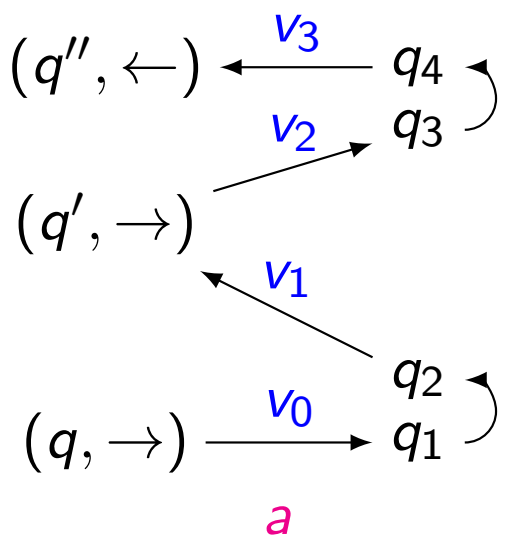
Transitions for $a = \left\{ \begin{array}{ll} (q, \rightarrow, a, q_1, \rightarrow, v_0) & (q_2, \leftarrow, a, q', \rightarrow, v_1) \\ (q', \rightarrow, a, q_3, \rightarrow, v_2) & (q_4, \leftarrow, a, q'', \rightarrow, v_3) \\ \\ (r, \rightarrow, a, r', \leftarrow, v_4) & (r', \leftarrow, a, r'', \leftarrow v_5) \end{array} \right.$

New traversal $T' = \{(q, q''), (r, r'')\}$ and a transition (T', a, T) in the look-ahead.

From D2FST to Universal Transducers

The transition rules construction

We produce a transition rule for each $a \in \Sigma$, for each pair (q, q'') and each look-ahead T such that $(q, q'') \in T'$ and (T', a, T) is a look-ahead rule. E.g.
$T = \{(q_1, q_2), (q_3, q_4)\}$

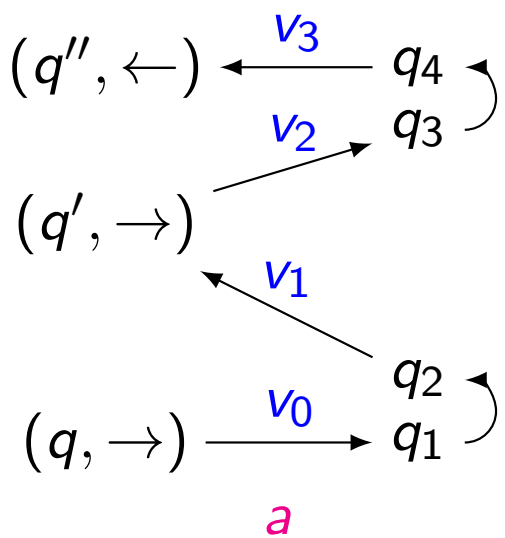


$(q, q'')(a) \xrightarrow{T} v_0(q_1, q_2)v_1 v_2(q_2, q_3)v_3$

From D2FST to Universal Transducers

The transition rules construction

We produce a transition rule for each $a \in \Sigma$, for each pair (q, q'') and each look-ahead T such that $(q, q'') \in T'$ and (T', a, T) is a look-ahead rule. E.g.
$T = \{(q_1, q_2), (q_3, q_4)\}$



$(q, q'')(a) \xrightarrow{T} v_0(q_1, q_2)v_1 v_2(q_2, q_3)v_3$

Outline

2 - Normalizing universal transducer

Classifying Ranges of states under some look-ahead

A language $[\![q]\!]_p$ is

periodic if for $v \in \Sigma^*$,

$$[\![q]\!]_p \subseteq \{v\}^*$$

the shortest of such v is the period π of $[\![q]\!]_p$

almost-periodic if for some $\beta, \pi \in \Sigma^*$,

$$[\![q]\!]_p \subseteq \beta.\{\pi\}^*$$

semi-periodic if for some $k \in \mathbb{N}$, some $\{(\beta_i, \pi_i) \mid i \in [1..k]\}$,

$$[\![q]\!]_p \subseteq \bigcup_k \beta_i.\{\pi_i\}^*$$

strongly aperiodic (and thus, infinite) if it is not semi-periodic

The Role of Normalization of Universal Transducers

A universal transducer $\mathcal{G}$ with look-ahead automaton B

What ?

Find an equivalent universal transducer $\mathcal{G}'$ s.t. for all q, p,

$[\![q]\!]_p$ *is either infinite periodic or strongly aperiodic*

How ?

Essentially modify the look-ahead automaton B (and $\mathcal{G}$ slightly)

Normalization : Example 1

Reverse

$$\tau_I(p) = q$$

$$q(a) \xrightarrow{p} q\,a$$
$$q(b) \xrightarrow{p} q\,b$$
$$q(\triangleleft) \longrightarrow \epsilon$$

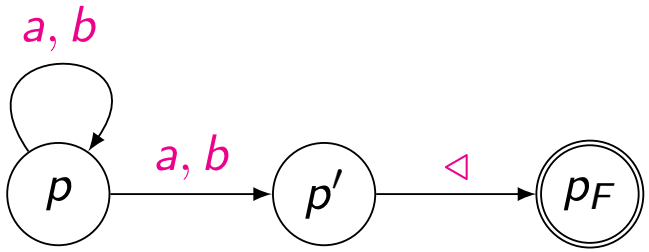
$[\![q]\!]_p = \{a, b\}^*$ is strongly aperiodic

This universal transducer is normalized

Normalization : Example 2

LastFirst : $\triangleright\{a,b\}^+\triangleleft \rightarrow \triangleright\{a,b,c\}^+\triangleleft$

$\triangleright u\sigma\triangleleft \rightarrow \triangleright\sigma c^{|u|}\triangleleft$ with $\sigma \in \{a,b\}$



$\tau(p) = q$

$q(a) \xrightarrow{p} q\,c$ $\quad$ $q(b) \xrightarrow{p} q\,c$

$q(a) \xrightarrow{p'} q'\,a$ $\quad$ $q(b) \xrightarrow{p'} q'\,b$

$q'(\triangleleft) \longrightarrow \epsilon$

$[\![q]\!]_p = ac^* + bc^*$ is semi-periodic

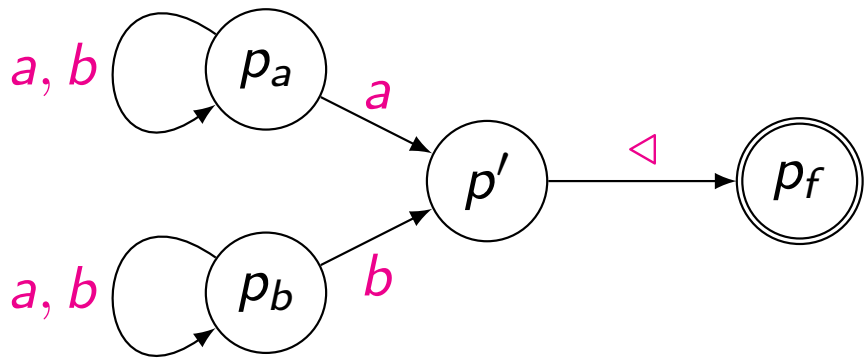
Normalization : Example 2 - Step 1

Removing semi-periodic states :

Look-ahead refinement by splitting states according to the sum of almost-periodic languages

$[\![q]\!]_p = ac^* + bc^*$

p is splitted into p_a and p_b



$\tau_I(p_a) = \triangleright\, q \triangleleft$
$\tau_I(p_b) = \triangleright\, q \triangleleft$

$q(a) \xrightarrow{p_a} q\, c \qquad q(b) \xrightarrow{p_a} q\, c \qquad q'(\triangleleft) \longrightarrow \epsilon$
$q(a) \xrightarrow{p_b} q\, c \qquad q(b) \xrightarrow{p_b} q\, c$
$q(a) \xrightarrow{p'} q'\, a \qquad q(b) \xrightarrow{p'} q'\, b$

$[\![q]\!]_{p_a} = ac^* \qquad [\![q]\!]_{p_a} = bc^*$

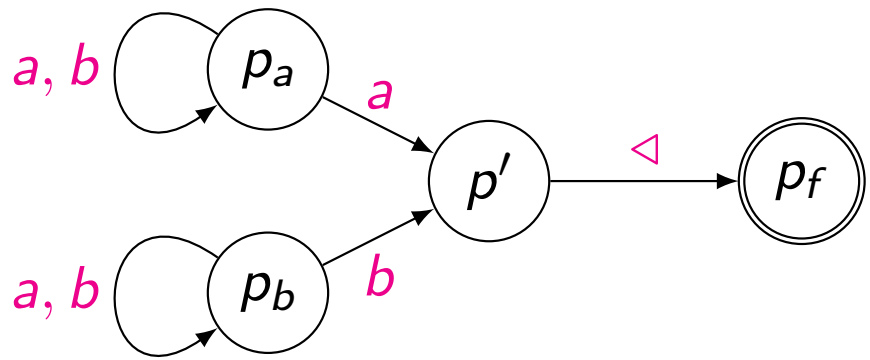
Normalization : Example 2 - Step 1

Removing semi-periodic states :

Look-ahead refinement by splitting states according to the sum of almost-periodic languages

$[\![q]\!]_p = ac^* + bc^*$

p is splitted into p_a and p_b



$\tau_I(p_a) = \triangleright\, q \triangleleft$
$\tau_I(p_b) = \triangleright\, q \triangleleft$

$q(a) \xrightarrow{p_a} q\,c \qquad q(b) \xrightarrow{p_a} q\,c \qquad q'(\triangleleft) \longrightarrow \epsilon$
$q(a) \xrightarrow{p_b} q\,c \qquad q(b) \xrightarrow{p_b} q\,c$
$q(a) \xrightarrow{p'} q'\,a \qquad q(b) \xrightarrow{p'} q'\,b$

$[\![q]\!]_{p_a} = ac^* \qquad [\![q]\!]_{p_a} = bc^*$

Normalization : Example 2 - Step 2

Removing almost-periodic states :

$\sim$ earliest output of longest common prefix (lcp)

- identify the first letter $\sigma_{q,p}$ of the lcp's of all $[\![q]\!]_p$
- refine the look-ahead and then modify the transition rules to output $\sigma_{q,p}$ as soon as possible
- reiterate (lcp of the $[\![q]\!]_p$ are shorter)

LastFirst :

$$\tau_l(p_a) = \triangleright\, q \triangleleft$$
$$\tau_l(p_b) = \triangleright\, q \triangleleft$$

$$q(a) \xrightarrow{p_a} q\, c \qquad q(b) \xrightarrow{p_a} q\, c$$
$$q(a) \xrightarrow{p_b} q\, c \qquad q(b) \xrightarrow{p_b} q\, c$$
$$q(a) \xrightarrow{p'} q'\, a \qquad q(b) \xrightarrow{p'} q'\, b$$
$$q'(\triangleleft) \longrightarrow \epsilon$$

Normalization : Example 2 - Step 2

Removing almost-periodic states :

$\sim$ earliest output of longest common prefix (lcp)

- identify the first letter $\sigma_{q,p}$ of the lcp's of all $[\![q]\!]_p$
- refine the look-ahead and then modify the transition rules to output $\sigma_{q,p}$ as soon as possible
- reiterate (lcp of the $[\![q]\!]_p$ are shorter)

LastFirst : Look-ahead is unchanged

$$\tau_I(p_a) = \triangleright\, a\, q \triangleleft$$
$$\tau_I(p_b) = \triangleright\, b\, q \triangleleft$$

$$q(a) \xrightarrow{p_a} q\,c \qquad q(b) \xrightarrow{p_a} q\,c$$
$$q(a) \xrightarrow{p_b} q\,c \qquad q(b) \xrightarrow{p_b} q\,c$$
$$q(a) \xrightarrow{p'} \epsilon \qquad q(b) \xrightarrow{p'} \epsilon$$

$[\![q]\!]_{p_a}, [\![q]\!]_{p_b}$ are periodic of period c

Outline

3 - Deciding one-wayness of universal transducers

Pushing Words through Languages / States

Ideas borrowed from [Laurence, Lemay, Niehren, Staworko and Tommasi: "Learning Sequential Tree-to-Word Transducers"]

For a word v and a language L,

$$\text{Push}(L, v) = L' \quad \text{if } L.v = v.L'$$

- L is periodic of period π then $v = \pi^k \pi'$ for $k \in \mathbb{N}$ and $\pi = \pi'\pi''$ and $L' = (\pi')^{-1}(L.\pi')$.
- L is strongly aperiodic then the set of v's is finite and prefix-closed and $L' = v^{-1}(L.v)$

Pushing Words through Languages / States

Ideas borrowed from [Laurence, Lemay, Niehren, Staworko and Tommasi: "Learning Sequential Tree-to-Word Transducers"]

For a word v and a language L,

$$\mathrm{Push}(L, v) = L' \quad \text{if } L.v = v.L'$$

- L is periodic of period π then $v = \pi^k \pi'$ for $k \in \mathbb{N}$ and $\pi = \pi'\pi''$ and $L' = (\pi')^{-1}(L.\pi')$.
- L is strongly aperiodic then the set of v's is finite and prefix-closed and $L' = v^{-1}(L.v)$

$$\mathrm{Push}(\{\epsilon, ab, ababab, abababababab\}, aba) = \{\epsilon, ba, bababa, bababababa\}$$

$$\mathrm{Push}((ab)^*, a) = (ba)^*$$

$$\mathrm{Push}(a\Sigma^*, a) = \Sigma^* a$$

Pushing Words through Languages / States

Ideas borrowed from [Laurence, Lemay, Niehren, Staworko and Tommasi: "Learning Sequential Tree-to-Word Transducers"]

For a word v and a language L,

$$\mathrm{Push}(L, v) = L' \quad \text{if } L.v = v.L'$$

- L is periodic of period π then $v = \pi^k \pi'$ for $k \in \mathbb{N}$ and $\pi = \pi'\pi''$ and $L' = (\pi')^{-1}(L.\pi')$.
- L is strongly aperiodic then the set of v's is finite and prefix-closed and $L' = v^{-1}(L.v)$

Pushing Words through States under Look-ahead

Annotating states with (part of) words that are pushed through

Define syntactically $Push([\![q]\!]_p, v)$ by means of a (finitely many) new states $q_{|\bar{v}}$ together with its transition rules such that $Push([\![q]\!]_p, v) = [\![q_{|\bar{v}}]\!]_p$

The Algorithm in a Nutshell

Compute from a universal transducer $\mathcal{G}$ with m copies a one-way deterministic transducer A with look-ahead.

Ingredient:

- The look-ahead of A is the look-ahead of $\mathcal{G}$
- States from A are of the form $[q_1 v_1 \dots q_n v_n]$ with $n \leq m$ and such that
  - the q_i's are states of $\mathcal{G}$
  - the v_i's remain small
- compute new states with the semantics of universal transducers and Pushing

Small ?
Not larger than $F(|\mathcal{G}|)$

Running the Algorithm on Examples (I)

Reverse

$$\tau_I(p) = q$$

$$q(a) \xrightarrow{p} q\,a$$
$$q(b) \xrightarrow{p} q\,b$$
$$q(\triangleleft) \longrightarrow \epsilon$$

$(s_I, \rightarrow, \triangleright, p, [q], \rightarrow, \epsilon)$
$([q], \rightarrow, a, p, [qa], \rightarrow, \epsilon)$
$([qa], \rightarrow, a, p, [qaa], \rightarrow, \epsilon)$
$\ldots$
$([qa \ldots a], \rightarrow, a, p, [qa \ldots aa], \rightarrow, \epsilon)$

$[qa \ldots aa]$ larger than $F(|\mathcal{G}|) \Rightarrow$ not one-way definable

Running the Algorithm on Examples (I)

Reverse

$$\tau_I(p) = q$$

$$q(a) \xrightarrow{p} q\,a$$
$$q(b) \xrightarrow{p} q\,b$$
$$q(\triangleleft) \longrightarrow \epsilon$$

$(s_I, \rightarrow, \triangleright, p, [q], \rightarrow, \epsilon)$
$([q], \rightarrow, a, p, [qa], \rightarrow, \epsilon)$
$([qa], \rightarrow, a, p, [qaa], \rightarrow, \epsilon)$
$\ldots$
$([qa \ldots a], \rightarrow, a, p, [qa \ldots aa], \rightarrow, \epsilon)$

$[qa \ldots aa]$ larger than $F(|\mathcal{G}|) \Rightarrow$ not one-way definable

Running the Algorithm on Examples (I)

Reverse

$$\tau_I(p) = q$$

$$q(a) \xrightarrow{p} q\,a$$
$$q(b) \xrightarrow{p} q\,b$$
$$q(\triangleleft) \longrightarrow \epsilon$$

$(s_I, \rightarrow, \triangleright, p, [q], \rightarrow, \epsilon)$
$([q], \rightarrow, a, p, [qa], \rightarrow, \epsilon)$
$([qa], \rightarrow, a, p, [qaa], \rightarrow, \epsilon)$
$\ldots$
$([qa \ldots a], \rightarrow, a, p, [qa \ldots aa], \rightarrow, \epsilon)$

$[qa \ldots aa]$ larger than $F(|\mathcal{G}|) \Rightarrow$ not one-way definable

Running the Algorithm on Examples (I)

Reverse

$$\tau_I(p) = q$$

$$q(a) \xrightarrow{p} q\,a$$
$$q(b) \xrightarrow{p} q\,b$$
$$q(\triangleleft) \longrightarrow \epsilon$$

$(s_I, \rightarrow, \triangleright, p, [q], \rightarrow, \epsilon)$
$([q], \rightarrow, a, p, [qa], \rightarrow, \epsilon)$
$([qa], \rightarrow, a, p, [qaa], \rightarrow, \epsilon)$
$\ldots$
$([qa \ldots a], \rightarrow, a, p, [qa \ldots aa], \rightarrow, \epsilon)$

$[qa \ldots aa]$ larger than $F(|\mathcal{G}|) \Rightarrow$ not one-way definable

Running the Algorithm on Examples (II)

LastFirst : $\triangleright\{a,b\}^+\triangleleft \to \triangleright\{a,b,c\}^+\triangleleft$ with $\sigma \in \{a,b\}$

$\triangleright u\sigma\triangleleft \to \triangleright\sigma c^{|u|}\triangleleft$

$\tau_I(p_a) = \triangleright a\, q\triangleleft$

$\tau_I(p_b) = \triangleright b\, q\triangleleft$

$q(a) \xrightarrow{p_a} q\,c$ $\quad$ $q(b) \xrightarrow{p_a} q\,c$

$q(a) \xrightarrow{p_b} q\,c$ $\quad$ $q(b) \xrightarrow{p_b} q\,c$

$q(a) \xrightarrow{p'} \epsilon$ $\quad$ $q(b) \xrightarrow{p'} \epsilon$

$[\![q]\!]_{p_a}, [\![q]\!]_{p_b}$ are periodic of period c $\quad$ $\begin{cases} \mathrm{Push}(q,p_a,c) = cq \\ \mathrm{Push}(q,p_b,c) = cq \end{cases}$

$(s_I, \to, \triangleright, p_a, [q\triangleleft], \to, \triangleright a)$

using $q(a) \xrightarrow{p_a} q\,c$ and pushing, $[qc\triangleleft]$ becomes $c[q\triangleleft]$

$([q\triangleleft], \to, a, p_a, [q\triangleleft], \to, c)$ $\quad$ $([q\triangleleft], \to, b, p_a, [q\triangleleft], \to, c)$

$([q\triangleleft], \to, a, p_b, [q\triangleleft], \to, c)$ $\quad$ $([q\triangleleft], \to, b, p_b, [q\triangleleft], \to, c)$

$([q\triangleleft], \to, a, p', s_F, \to, \triangleleft)$ $\quad$ $([q\triangleleft], \to, b, p', s_F, \to, \triangleleft)$

Running the Algorithm on Examples (II)

LastFirst : $\triangleright\{a,b\}^+\triangleleft \rightarrow \triangleright\{a,b,c\}^+\triangleleft$
$\triangleright u\sigma\triangleleft \rightarrow \triangleright\sigma c^{|u|}\triangleleft$
with $\sigma \in \{a,b\}$

$\tau_I(p_a) = \triangleright a\, q\triangleleft$
$\tau_I(p_b) = \triangleright b\, q\triangleleft$

$q(a) \xrightarrow{p_a} q\,c$ $\quad$ $q(b) \xrightarrow{p_a} q\,c$
$q(a) \xrightarrow{p_b} q\,c$ $\quad$ $q(b) \xrightarrow{p_b} q\,c$
$q(a) \xrightarrow{p'} \epsilon$ $\quad$ $q(b) \xrightarrow{p'} \epsilon$

$[\![q]\!]_{p_a}, [\![q]\!]_{p_b}$ are periodic of period c $\quad$ $\begin{cases} \text{Push}(q,p_a,c) = cq \\ \text{Push}(q,p_b,c) = cq \end{cases}$

$(s_I, \rightarrow, \triangleright, p_a, [q\triangleleft], \rightarrow, \triangleright a)$
using $q(a) \xrightarrow{p_a} q\,c$ and pushing, $[qc\triangleleft]$ becomes $c[q\triangleleft]$
$([q\triangleleft], \rightarrow, a, p_a, [q\triangleleft], \rightarrow, c)$ $\quad$ $([q\triangleleft], \rightarrow, b, p_a, [q\triangleleft], \rightarrow, c)$
$([q\triangleleft], \rightarrow, a, p_b, [q\triangleleft], \rightarrow, c)$ $\quad$ $([q\triangleleft], \rightarrow, b, p_b, [q\triangleleft], \rightarrow, c)$
$([q\triangleleft], \rightarrow, a, p', s_F, \rightarrow, \triangleleft)$ $\quad$ $([q\triangleleft], \rightarrow, b, p', s_F, \rightarrow, \triangleleft)$

Running the Algorithm on Examples (II)

LastFirst : $\triangleright\{a,b\}^+\triangleleft \rightarrow \triangleright\{a,b,c\}^+\triangleleft$
$\triangleright u\sigma\triangleleft \rightarrow \triangleright\sigma c^{|u|}\triangleleft$
with $\sigma \in \{a,b\}$

$\tau_I(p_a) = \triangleright a\, q\triangleleft$
$\tau_I(p_b) = \triangleright b\, q\triangleleft$

$q(a) \xrightarrow{p_a} q\,c$ $\quad$ $q(b) \xrightarrow{p_a} q\,c$
$q(a) \xrightarrow{p_b} q\,c$ $\quad$ $q(b) \xrightarrow{p_b} q\,c$
$q(a) \xrightarrow{p'} \epsilon$ $\quad$ $q(b) \xrightarrow{p'} \epsilon$

$[\![q]\!]_{p_a}, [\![q]\!]_{p_b}$ are periodic of period c $\quad$ $\begin{cases} \text{Push}(q,p_a,c) = cq \\ \text{Push}(q,p_b,c) = cq \end{cases}$

$(s_I, \rightarrow, \triangleright, p_a, [q\triangleleft], \rightarrow, \triangleright a)$
using $q(a) \xrightarrow{p_a} q\,c$ and pushing, $[qc\triangleleft]$ becomes $c[q\triangleleft]$
$([q\triangleleft], \rightarrow, a, p_a, [q\triangleleft], \rightarrow, c)$ $\quad$ $([q\triangleleft], \rightarrow, b, p_a, [q\triangleleft], \rightarrow, c)$
$([q\triangleleft], \rightarrow, a, p_b, [q\triangleleft], \rightarrow, c)$ $\quad$ $([q\triangleleft], \rightarrow, b, p_b, [q\triangleleft], \rightarrow, c)$
$([q\triangleleft], \rightarrow, a, p', s_F, \rightarrow, \triangleleft)$ $\quad$ $([q\triangleleft], \rightarrow, b, p', s_F, \rightarrow, \triangleleft)$

Running the Algorithm on Examples (II)

LastFirst : $\triangleright\{a,b\}^+\triangleleft \rightarrow \triangleright\{a,b,c\}^+\triangleleft$
$\triangleright u\sigma\triangleleft \rightarrow \triangleright\sigma c^{|u|}\triangleleft$
with $\sigma \in \{a,b\}$

$\tau_I(p_a) = \triangleright a\, q\triangleleft$
$\tau_I(p_b) = \triangleright b\, q\triangleleft$

$q(a) \xrightarrow{p_a} q\,c$ $\quad$ $q(b) \xrightarrow{p_a} q\,c$
$q(a) \xrightarrow{p_b} q\,c$ $\quad$ $q(b) \xrightarrow{p_b} q\,c$
$q(a) \xrightarrow{p'} \epsilon$ $\quad$ $q(b) \xrightarrow{p'} \epsilon$

$[\![q]\!]_{p_a}, [\![q]\!]_{p_b}$ are periodic of period c $\quad$ $\begin{cases} \mathrm{Push}(q,p_a,c) = cq \\ \mathrm{Push}(q,p_b,c) = cq \end{cases}$

$(s_I, \rightarrow, \triangleright, p_a, [q\triangleleft], \rightarrow, \triangleright a)$
using $q(a) \xrightarrow{p_a} q\,c$ and pushing, $[qc\triangleleft]$ becomes $c[q\triangleleft]$
$([q\triangleleft], \rightarrow, a, p_a, [q\triangleleft], \rightarrow, c)$ $\quad$ $([q\triangleleft], \rightarrow, b, p_a, [q\triangleleft], \rightarrow, c)$
$([q\triangleleft], \rightarrow, a, p_b, [q\triangleleft], \rightarrow, c)$ $\quad$ $([q\triangleleft], \rightarrow, b, p_b, [q\triangleleft], \rightarrow, c)$
$([q\triangleleft], \rightarrow, a, p', s_F, \rightarrow, \triangleleft)$ $\quad$ $([q\triangleleft], \rightarrow, b, p', s_F, \rightarrow, \triangleleft)$

To Sum up

If successful, A is of size at most 4-exponential in the size of the input D2FST.

Guessing a failure requires only 3-exponential space.

Next ?

- On going : the second step of normalization is useless $\Rightarrow$ drop one exponential
- Educated guess : pushing only powers of (complete) periods through periodic language
- (False ?) hope : technics could be adapted to (normalized) functional 2FST.

To Sum up

If successful, A is of size at most 4-exponential in the size of the input D2FST.

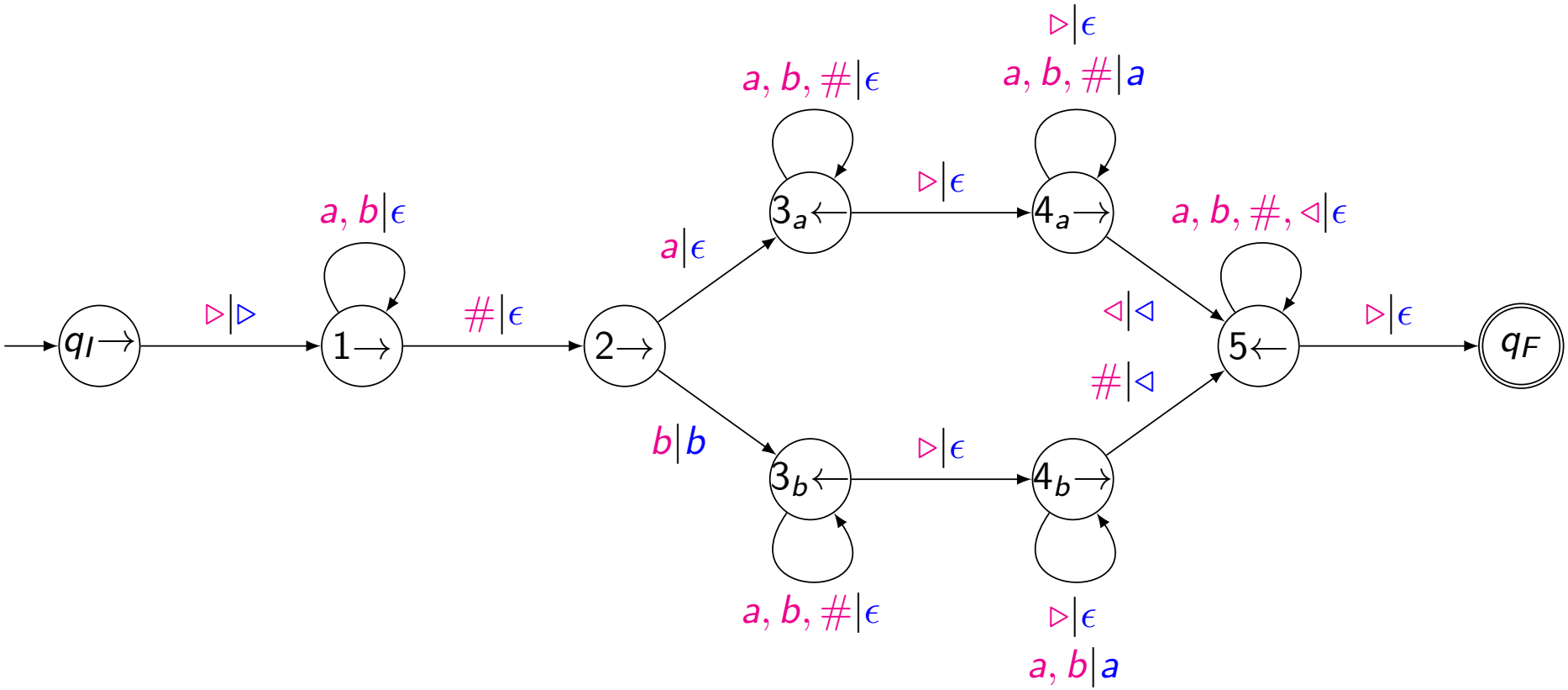
Guessing a failure requires only 3-exponential space.

Next ?

- On going : the second step of normalization is useless $\Rightarrow$ drop one exponential
- Educated guess : pushing only powers of (complete) periods through periodic language
- (False ?) hope : technics could be adapted to (normalized) functional 2FST.

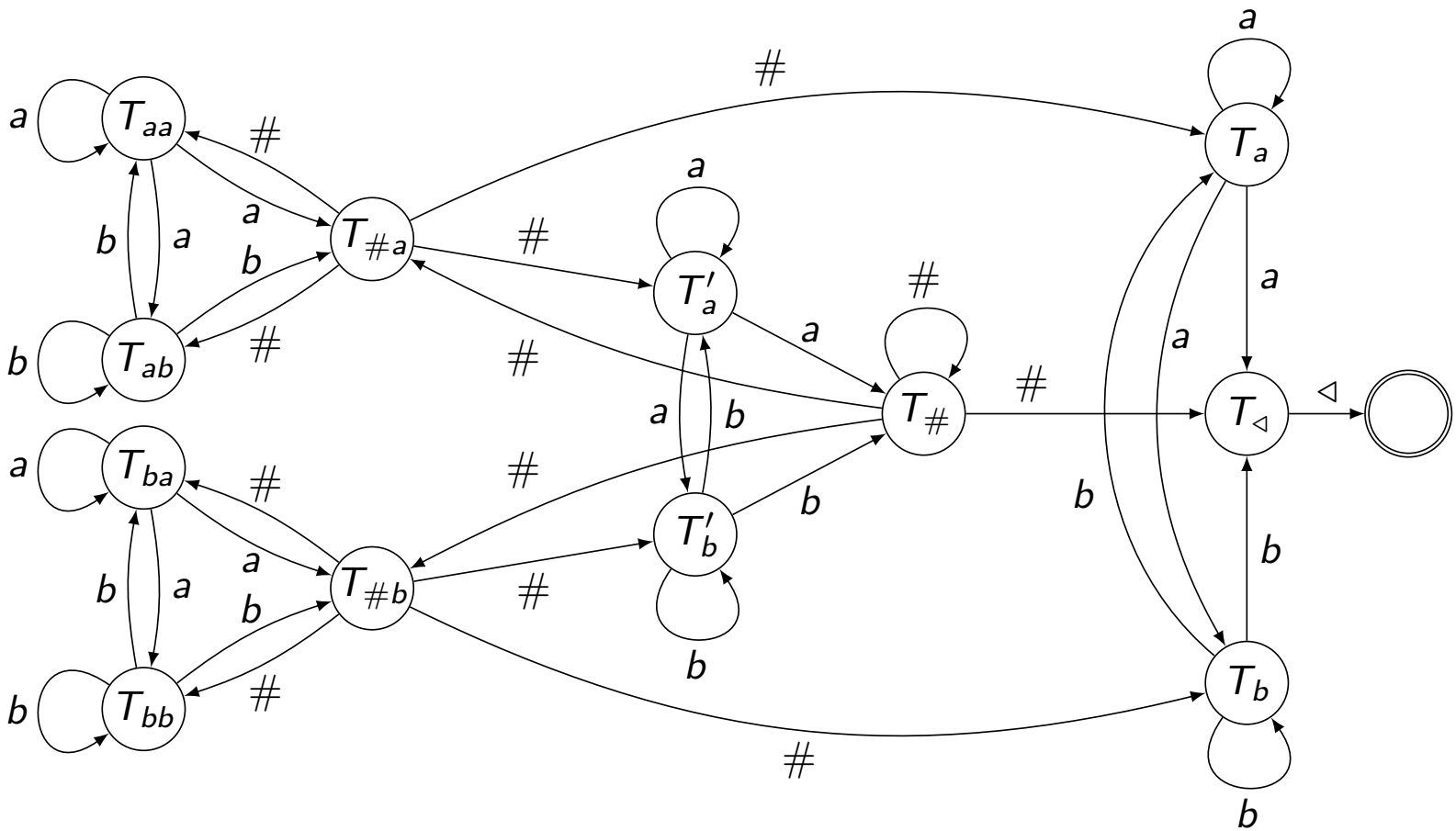
More Example

$[\![A]\!](\triangleright u_1 \# \sigma u_2 \triangleleft) = \triangleright a^{|u_1|+|u_2|+2} \triangleleft$ if $\sigma = a$, and
$[\![A]\!](\triangleright u_1 \# \sigma u_2 \triangleleft) = \triangleright b a^{|u_1|} \triangleleft$ if $\sigma = b$



More Example

The look-ahead automaton



More Example

$$(s_I, \rightarrow, \triangleright, T_a, (1,4)\triangleleft^{T_a}, \rightarrow, \triangleright a)$$
$$(s_I, \rightarrow, \triangleright, T_b, (1,4)\triangleleft^{T_b}, \rightarrow, \triangleright b)$$

$$((1,4)\triangleleft^{T_a}, \rightarrow, \sigma, T_a, (1,4)\triangleleft^{T_a}, \rightarrow, c) \qquad \text{for } \sigma \in \{a, b\}$$
$$((1,4)\triangleleft^{T_b}, \rightarrow, \sigma, T_b, (1,4)\triangleleft^{T_b}, \rightarrow, c) \qquad \text{for } \sigma \in \{a, b\}$$
$$((1,4)\triangleleft^{T_a}, \rightarrow, a, T_\triangleleft, \epsilon, \rightarrow, \triangleleft)$$
$$((1,4)\triangleleft^{T_b}, \rightarrow, b, T_\triangleleft, \epsilon, \rightarrow, \triangleleft)$$