Finite-State Transducers in Language and Speech Processing

報告人:郭榮芳 05/20/2003

1. M. Mohri, On some applications of Finite-state automata theory to natural language processing, J. Nature Language Eng. 2 (1996).
2. M. Mohri, Finite-state transducers in language and speech processing, Comput. Linguistics 23 (2) (1997).

Outline

- Introduction
- Sequential string-to-string transducers
- Power series and subsequential string-to-weight transducers
- Application to speech recognition

Introduction

- Finite-state machines have been used in many areas of computational linguistics. Their use can be justified by both linguistic and computational arguments.

Linguistically

- Finite automata are convenient since they allow one to describe easily most of the relevant local phenomena encountered in the empirical study of language.
- They often lead to a compact representation of lexical rules, or idioms and clich es, that appears as natural to linguists (Gross, 1989).

Linguistically(cont.)

- Graphic tools also allow one to visualize and modify automata.This helps in correcting and completing a grammar.
- Other more general phenomena such as parsing context-free grammars can also be dealt with using finite-state machines such as RTN's (Woods, 1970).

Computational

- The use of finite-state machines is mainly motivated by considerations of time and space efficiency.
- Time efficiency is usually achieved by using deterministic automata.
  - Deterministic automata
    - Have a deterministic input.
    - For every state,at most one transition labeled with a given element of the alphabet .
- The output of deterministic machines depends, in gen- eral linearly.

Computational(cont.)

- Space efficiency is achieved with classical minimization algorithms (Aho,Hopcroft, and Ullman, 1974) for deterministic automata.
- Applications such as compiler construction have shown deterministic finite automata to be very efficient in practice (Aho, Sethi, and Ullman, 1986).

Applications in natural language processing

- Lexical analyzers
- The compilation of morphological
- Phonological rules
- Speech processing

The idea of deterministic automata

- Produce output strings or weights in addition to accepting(deterministically) input.
- Time efficiency
- Space efficiency
- A large increase in the size of data.

- Limitations of the corresponding techniques, however, are very often pointed out more than their advantages.
- The reason for that is probably that recent work in this field are not yet described in computer science textbooks.
- Sequential finite-state transducers are now used in all areas of computational linguistics.

The case of string-to-string transducers.

- These transducers have been successfully used in the representation of large-scale dictionaries, computational morphology, and local grammars and syntax.
- We describe the theoretical bases for the use of these transducers.In particular, we recall classical theorems and give new ones characterizing these transducers.

The case of sequential string-to-weight transducers

- These transducers appear as very interesting in speech processing. Language models, phone lattices and word lattices.
  - Determinization
  - Minimization
  - Unambiguous
- Some applications in speech recognition.

Sequential string-to-string transducers

- Sequential string-to-string transducers are used in various areas of natural language processing.
- Both determinization (Mohri, 1994c) and minimization algorithms (Mohri,1994b) have been defined for the class of *p*-subsequential transducers which includes sequential string-to-string transducers.
- In this section the theoretical basis of the use of sequential transducers is described.
- Classical and new theorems help to indicate the usefulness of these devices as well as their characterization.

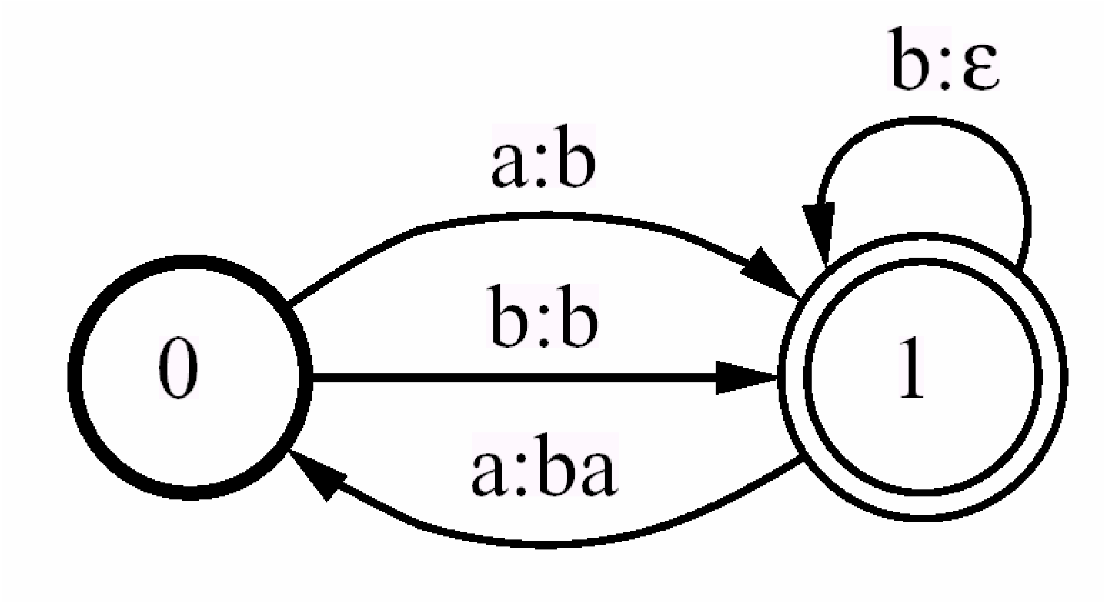
Sequential transducers

- **Sequential transducers**:
  - Sequential transducers has a deterministic input,namely at any state there is at most one transition labeled with a given element of the input alphabet.
  - Output labels might be strings, including the empty string ε .

Sequential transducers(cont.)

- Their use with a given input does not depend on the size of the transducer but only on that of the input.
- The total computational time is linear in the size of the input.

Example of a sequential transducer



Definition of Non-sequential transducer

$$T_1 = (V_1, I_1, F_1, A, B^*, \delta_1, \sigma_1)$$

– V_1 is the set of states,

– I_1 is the initial state,

– F_1 is the set of final states,

– A and B, finite sets corresponding respectively to the input and output alphabets of the transducer,

– δ_1, the state transition function which maps $V_1 \times A$ to 2^{V_1} ,

– σ_1, the output function which maps $V_1 \times A \times V_1$ to B* .

Definition of Subsequential transducer

$$T_2 = (V_2, i_2, F_2, A, B^*, \delta_2, \sigma_2, \phi_2)$$

- I_2 the unique initial state,
- δ_2, the state transition function which maps $V_2 \times A$ *to* V_2 ,
- σ_1, the output function which maps $V_1 \times A$ to B^* ,
- Φ_2, the final function maps F to B^*

Denote

- $x \wedge y$ is the longest common prefix of two strings x and y.
- $x^{-1}(xy)$ is the string y obtained by dividing (xy) at left by x.
- Subsets made of pairs (q,w) of a state q of T_1 and a string $w \in B^*$
- $J_1(a)=\{(q,w) \mid \delta_1(q,a)$ defined and $(q,w) \in q_2\}$
- $J_2(a)=\{(q,w,q') \mid \delta_1(q,a)$ defined and $(q,w) \in q_2$ and $q' \in \delta_1(q,a)\}$

DETERMINIZATION_TRANSDUCER(T_1, T_2)

$F_2 \leftarrow \emptyset$

$i_2 \leftarrow \bigcup_{i \in I_1} \{(i, \epsilon)\}$

$Q \leftarrow \{i_2\}$

while $Q \neq \emptyset$

    do $q_2 \leftarrow head[Q]$

        if (there exists $(q, w) \in q_2$ such that $q \in F_1$)

            then $F_2 \leftarrow F_2 \cup \{q_2\}$

                    $\phi_2(q_2) \leftarrow w$

        for each a such that $(q, w) \in q_2$ **and** $\delta_1(q, a)$ defined

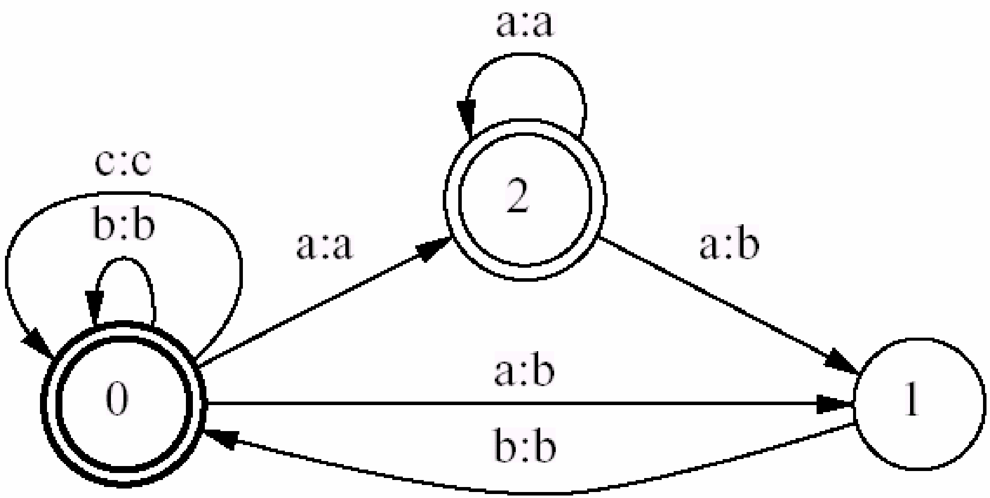
            do $\sigma_2(q_2, a) \leftarrow \bigwedge_{(q,a) \in J_1(a)} [w \bigwedge_{q' \in \delta_1(q,w)} \sigma_1(q, a, q')]$

                    $\delta_2(q_2, a) \leftarrow \bigcup_{(q,w,q') \in J_2(a)} \{(q', [\sigma_2(q_2, a)]^{-1} w \sigma_1(q, a, q'))\}$

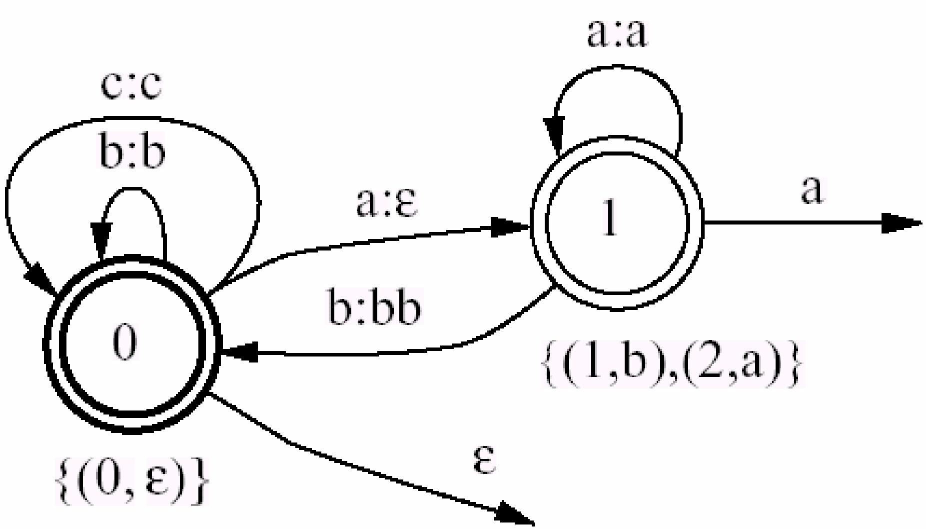
                    if ($\delta_2(q_2, a)$ is a new state)

                        then ENQUEUE$(Q, \delta_2(q_2, a))$

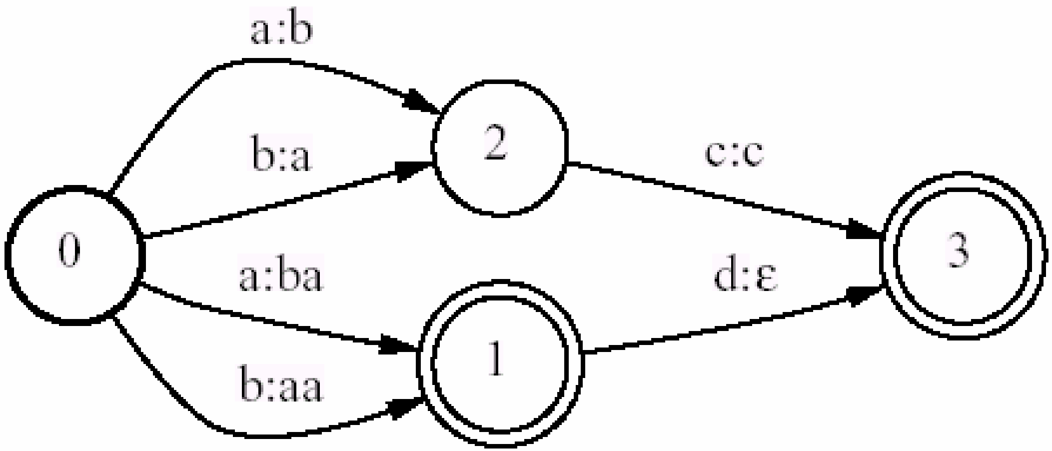
        DEQUEUE(Q)



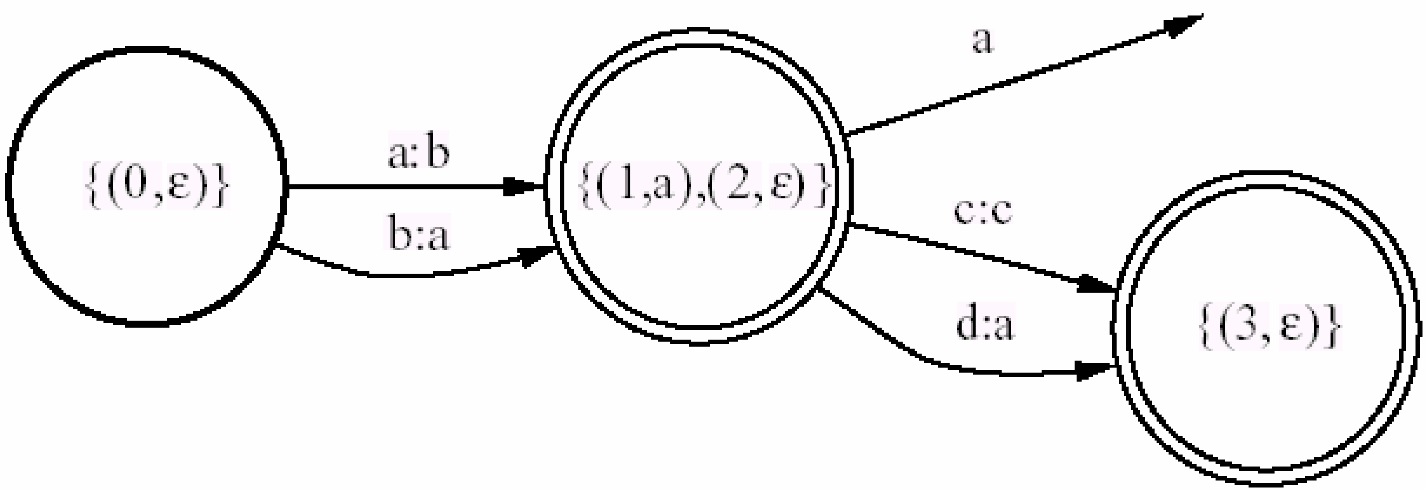
Transducer T_1



Subsequential transducer T_2 obtained from T_1 by determinization.



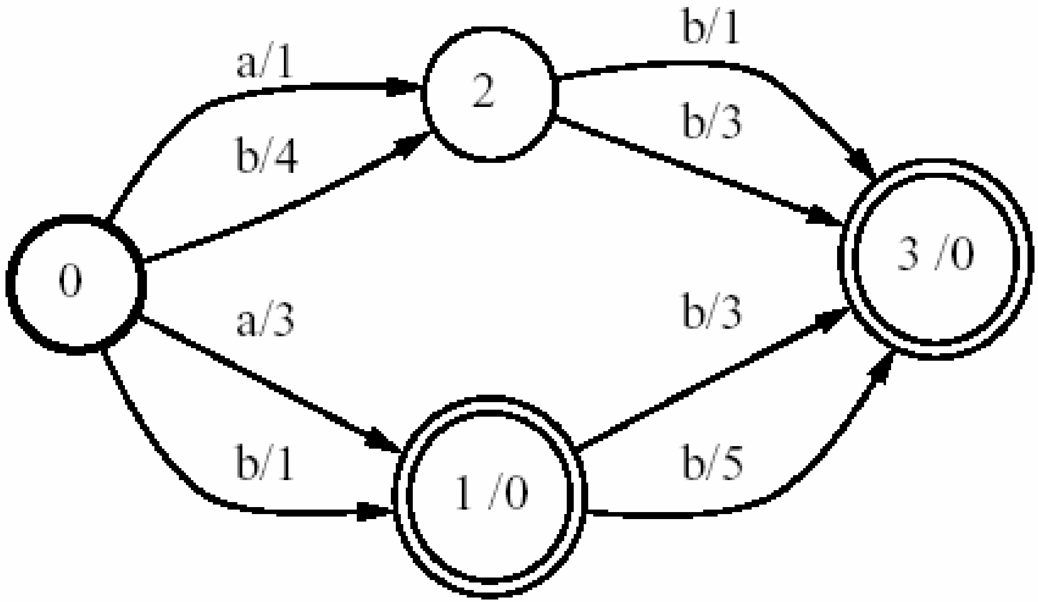
Transducer T_1



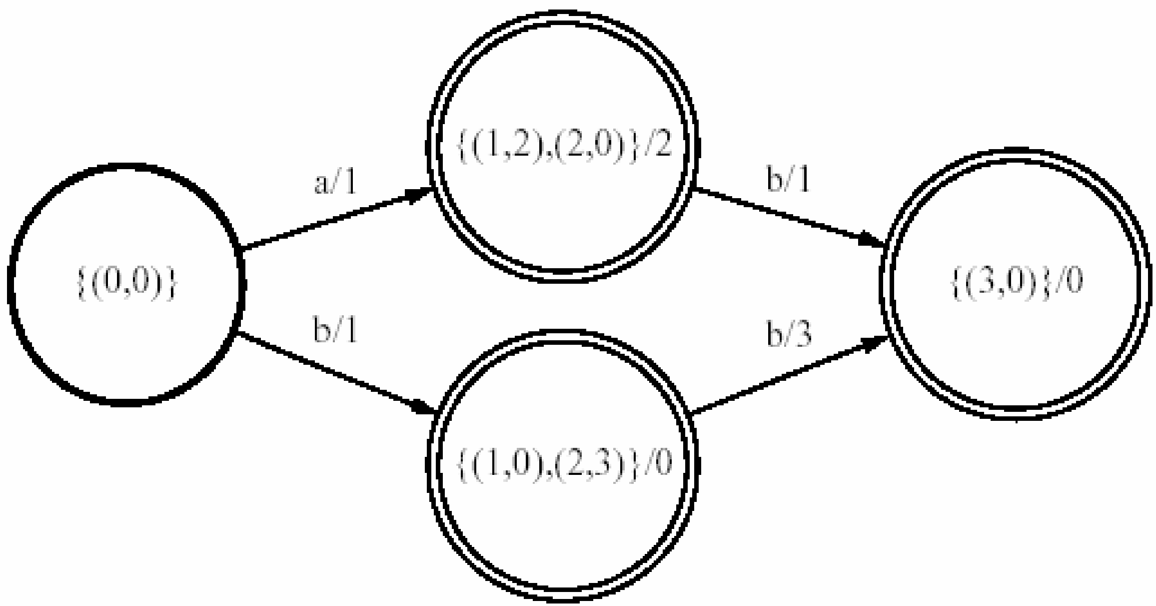
Subsequential transducer T_2 obtained from T_1 by determinization.

PowerSeriesDeterminization(τ_1, τ_2)

$F_2 \leftarrow \emptyset$
$\lambda_2 \leftarrow \bigoplus_{i \in I_1} \lambda_1(i)$
$i_2 \leftarrow \bigcup_{i \in I_1} \{(i, \lambda_2^{-1} \odot \lambda_1(i))\}$
$Q \leftarrow \{i_2\}$
while $Q \neq \emptyset$
 do $q_2 \leftarrow head[Q]$
    if (there exists $(q, x) \in q_2$ such that $q \in F_1$)
     then $F_2 \leftarrow F_2 \cup \{q_2\}$
         $\rho_2(q_2) \leftarrow \bigoplus_{q \in F_1, (q, x) \in q_2} x \odot \rho_1(q)$
    for each a such that $\Gamma(q_2, a) \neq \emptyset$
     do $\sigma_2(q_2, a) \leftarrow \bigoplus_{(q, x) \in \Gamma(q_2, a)} [x \odot \bigoplus_{t = (q, a, \sigma_1(t), n_1(t)) \in E_1} \sigma_1(t)]$
         $\delta_2(q_2, a) \leftarrow \bigcup_{q' \in \nu(q_2, a)} \{(q', \bigoplus_{(q, x, t) \in \gamma(q_2, a), n_1(t) = q'} [\sigma_2(q_2, a)]^{-1} \odot x \odot \sigma_1(t)\}$
         if ($\delta_2(q_2, a)$ is a new state)
          then ENQUEUE($Q, \delta_2(q_2, a)$)
    DEQUEUE(Q)



a/1
b/4
a/3
b/1
b/1
b/3
b/3
b/5
0
2
3 /0
1 /0



{(0,0)}
a/1
b/1
{(1,2),(2,0)}/2
{(1,0),(2,3)}/0
b/1
b/3
{(3,0)}/0

Definition of a sequential string-to-string transducer

- More formally, a sequential string-to-string transducer T is a 7-tuple ($Q, I, F, \Sigma, \Delta, \delta, \sigma$).
  - Q is the set of states,
  - $i \in Q$ is the initial state,
  - $F \subseteq Q$, the set of final states,
  - Σ and Δ, finite sets corresponding respectively to the input and output alphabets of the transducer,
  - Δ , the state transition function which maps $Q \times \Sigma$ *to* Q ,
  - σ, the output function which maps $Q \times \Sigma$ to Δ^* .

Subsequential and *p* -Subsequential transducers

- *p* :at most *p* final output strings at each final state.
- *p* -subsequential transducers seem to be sufficient for describing linguistic ambiguities.

Subsequential and p -Subsequential transducers (cont.)

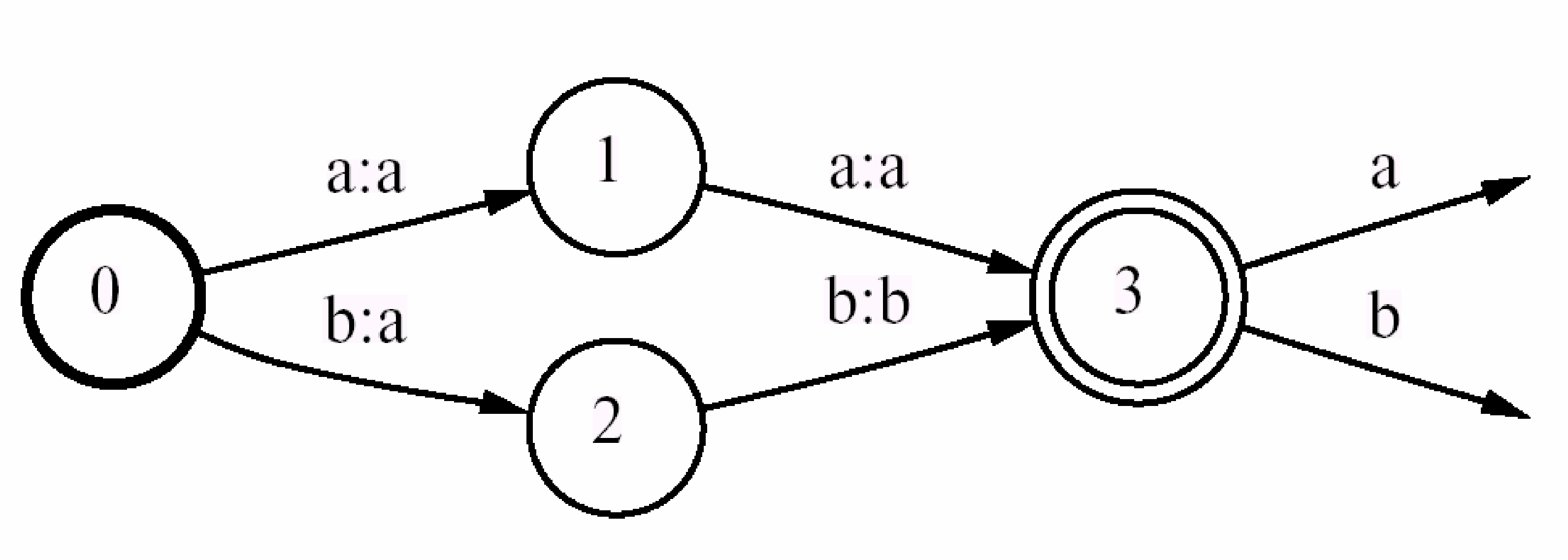


Figure 2
Example of a 2-subsequential transducer t_1

EX.input string w = aa gives two distinct outputs aaa and aab .

Composition

- If t_1 is a transducer from *input1* to *output1* and t_2 is a transducer from *input2* to *output2*,then $t_1 \circ t_2$ maps from *input1* to *output2.*
- *making the intersection of the outputs of* t_1 *with the inputs of* t_2*.*

Theorem 1

- Let $f : \Sigma^* \to \Delta^*$ be a sequential (resp. p -subsequential) and $g : \Delta^* \to \Omega^*$ be a sequential (resp. q -subsequential) function, then $g \circ f$ is sequential (resp. pq -subsequential).

Proof

- f: $\tau_1 = (Q_1, i_1, F_1, \Sigma, \Delta, \delta_1, \sigma_1, \rho_1)$ a *p* –subsequential transducer
- g: $\tau_2 = (Q_2, i_2, F_2, \Delta, \Omega, \delta_2, \sigma_2, \rho_2)$ a q –subsequential transducer
- ρ_1 and ρ_2 denote the final output functions of τ_1 and τ_2 which map respectively F_1 to $(\Delta^*)^p$ and F_2 to $(\Omega^*)^q$.
- $\rho_1(r)$ represents for instance the set of final output strings at a final state *r* .
- Define the pq -subsequential transducer

$\tau = (Q, i, F, \Sigma, \Omega, \delta, \sigma, \rho)$ by $Q = Q_1 \times Q_2$, $i = (i_1, i_2)$,

$F = \{(q_1, q_2) \in Q : q_1 \in F_1, \delta_2(q_2, \rho_1(q_1)) \cap F_2 \neq \emptyset\}$

Proof(cont.)

transition and output functions

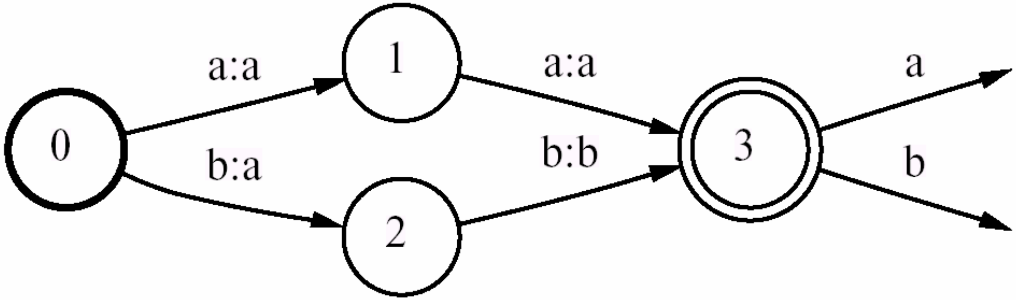
$$\forall a \in \Sigma, \ \forall (q_1, q_2) \in Q$$

$$\begin{aligned} \delta((q_1, q_2), a) &= (\delta_1(q_1, a), \delta_2(q_2, \sigma_1(q_1, a))) \\ \sigma((q_1, q_2), a) &= \sigma_2(q_2, \sigma_1(q_1, a)) \end{aligned}$$

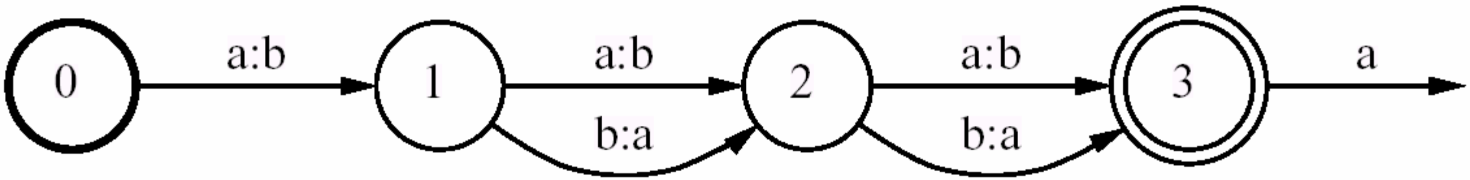
final output function

$$\forall (q_1, q_2) \in F,$$

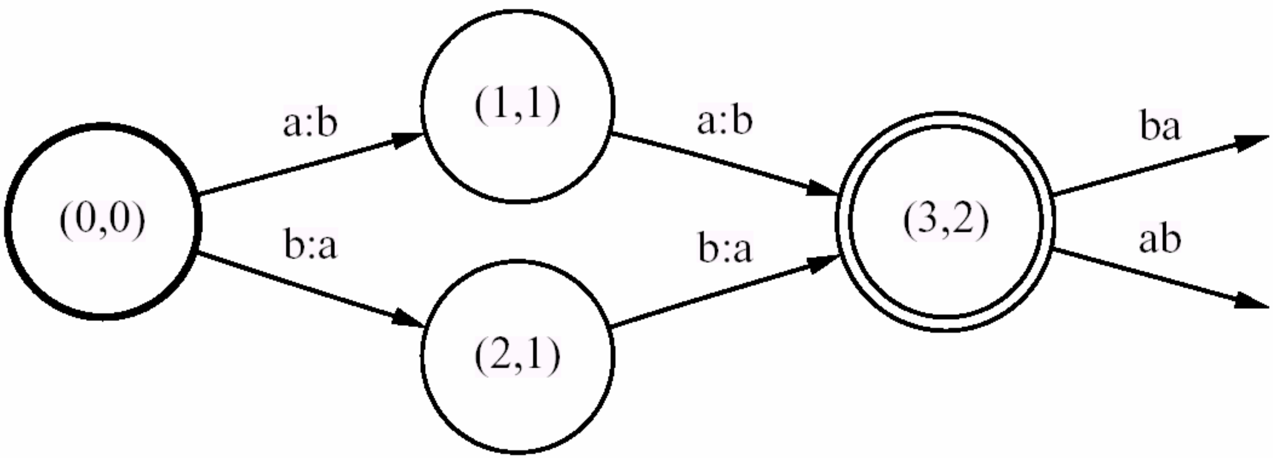
$$\rho((q_1, q_2)) = \sigma_2(q_2, \rho_1(q_1))\rho_2(\delta(q_2, \rho_1(q_1)))$$



Example of a 2-subsequential transducer τ_1.



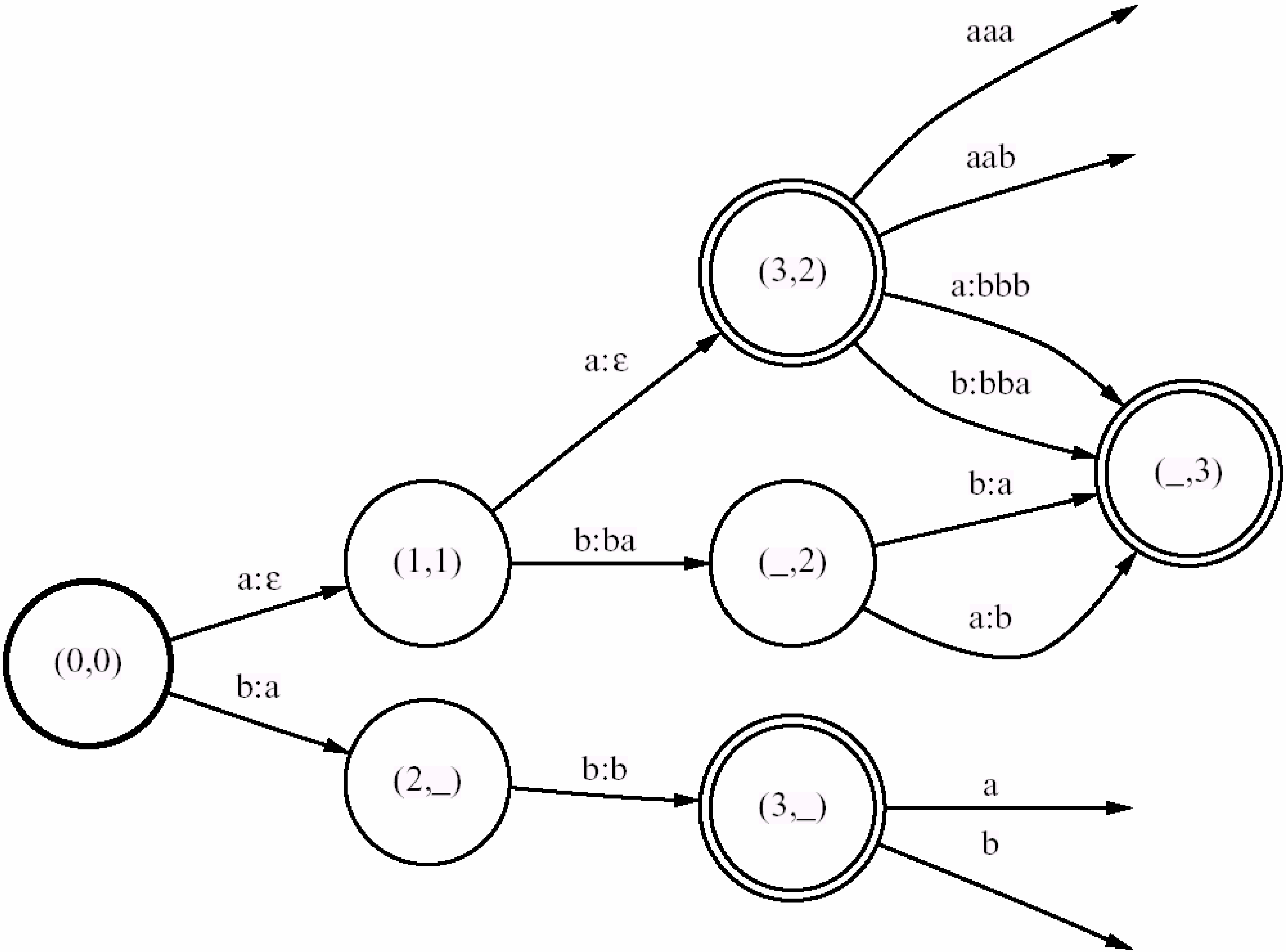
Example of a subsequential transducer τ_2.



2-Subsequential transducer τ_3, obtained by composition of τ_1 and τ_2.

Theorem 2

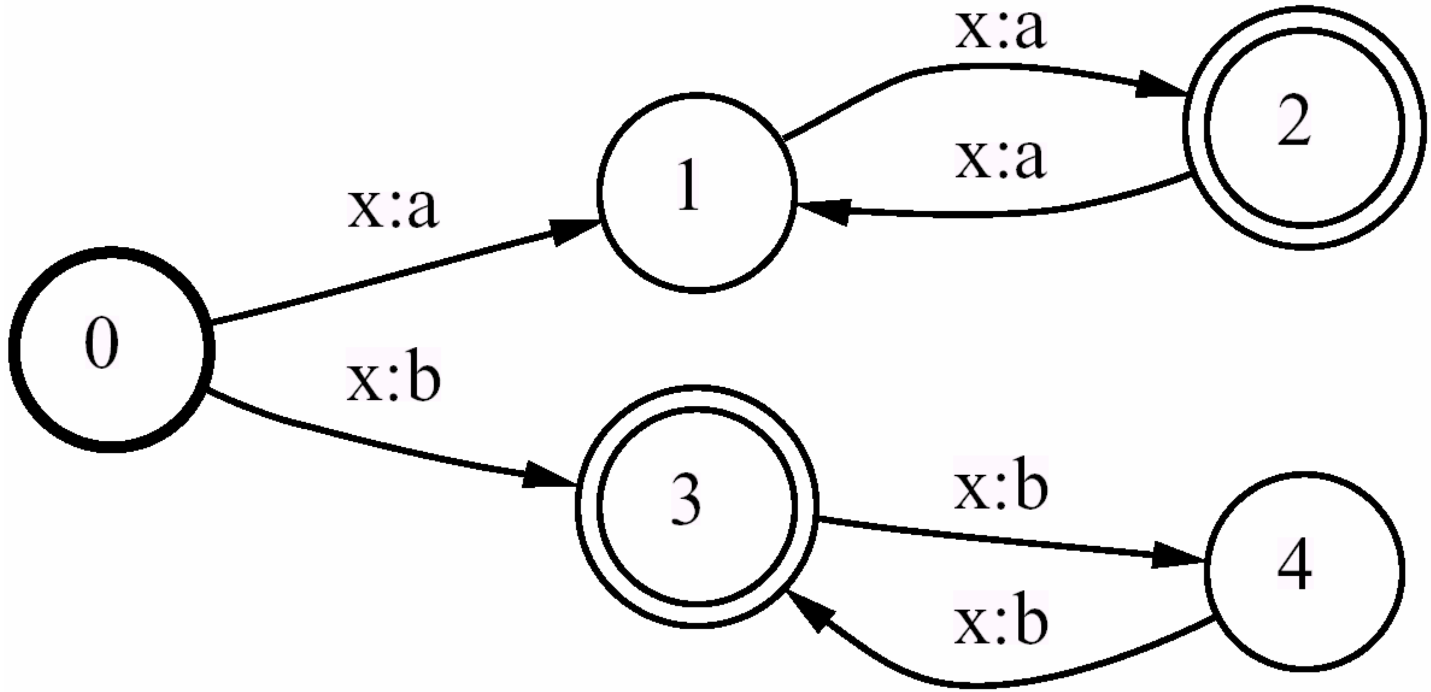
- Let $f : \Sigma^* \rightarrow \Delta^*$ be a sequential (resp. *p* -subsequential) and $g : \Sigma^* \rightarrow \Delta^*$ be a sequential (resp. q -subsequential) function, then *g* + *f* is 2-subsequential (resp. *(p* + *q*)-subsequential).



(0,0)
a:ε
b:a
(1,1)
(2,_)
a:ε
b:ba
b:b
(3,2)
(_,2)
(3,_)
aaa
aab
a:bbb
b:bba
b:a
a:b
(_,3)
a
b

UNION_p_SUBSEQUENTIAL_TRANSDUCER(T, T_1, T_2)

```
F ← ∅
i ← {(i_1, ε), (i_2, ε)}
Q ← {i}
while Q ≠ ∅
   do  q ← head[Q]   ▷ one can write: q = {(q_1, w_1), (q_2, w_2)}
       if (q_1 ∈ F_1 or q_2 ∈ F_2)
              then  F ← F ∪ {q}
                    for each output φ_ij(q_i) (i ∈ {1, 2}, j ≤ p)
                      do    ADD_OUTPUT(φ, q, w_i φ_ij(q_i))
       for each a such that δ_1(q_1, a) defined or δ_2(q_2, a) defined
              do    if (δ_1(q_1, a) undefined)
                      then  σ(q, a) ← w_2 σ_2(q_2, a)
                            δ(q, a) ← {(UNDEFINED, ε), (δ_2(q_2, a), ε)}
                    else if (δ_2(q_2, a) undefined)
                      then  σ(q, a) ← w_1 σ_1(q_1, a)
                            δ(q, a) ← {(δ_1(q_1, a), ε), (UNDEFINED, ε)}
                      else  σ(q, a) ← w_1 σ_1(q_1, a) ∧ w_2 σ_2(q_2, a)
                            δ(q, a) ← {(δ_1(q_1, a), [σ(q, a)]^{-1} w_1 σ_1(q_1, a)),
                                       (δ_2(q_2, a), [σ(q, a)]^{-1} w_2 σ_2(q_2, a))}
                    if (δ(q, a) is a new state)
                      then  ENQUEUE(Q, δ(q, a))
       DEQUEUE(Q)
```



$$\forall w \in \{x\}^+$$

$$\begin{aligned} f(w) &= a^{|w|} \quad if\ |w|\ is\ even, \\ &= b^{|w|} \quad otherwise \end{aligned}$$

We denote by $|w|$ the length of a string w.

Theorem 3

- Let f be a rational function mapping Σ^* to Δ^*

  f is sequential iff there exists a positive integer *K* such that:

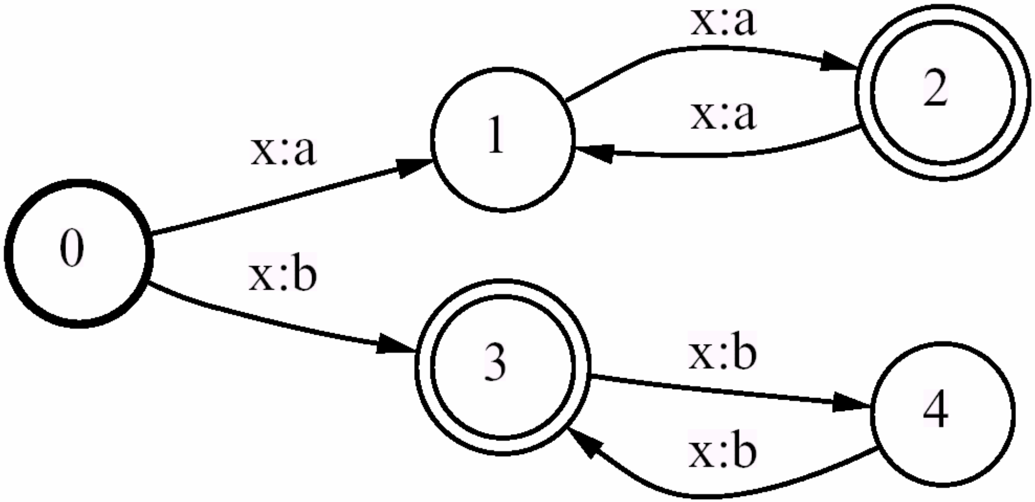
$$\forall u \in \Sigma^*, \forall a \in \Sigma,$$

$$\exists w \in \Delta^*, \ |w| \leq K :$$

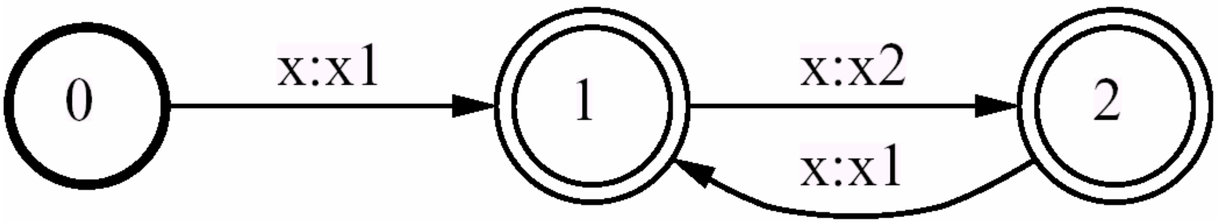
$$f(ua) = f(u)w$$

Theorem 4

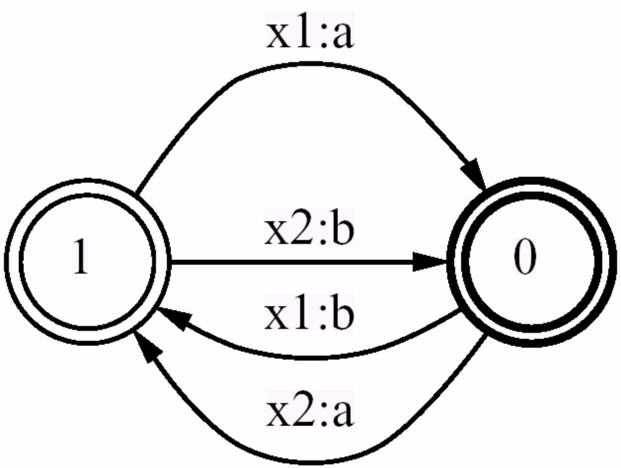
- Let f be a partial function mapping Σ^* to Δ^* . f is rational iff there exist a left sequential function $l : \Sigma^* \to \Omega^*$ and a right sequential function $r : \Omega^* \to \Delta^*$ such that $f = r \circ l$.



Transducer *T* with no equivalent sequential representation.



Left to right sequential transducer *L* .



Right to left sequential transducer R

Theorem 5

- Let T be a transducer mapping Σ^* to Δ^*. It is decidable whether T is sequential.
- Based on the definition of a metric on Σ^* Denote by $u \wedge v$ the longest common prefix of two strings u and v in Σ^*. It is easy to verify that the following defines a metric on Σ^*:

$$d(u, v) = |u| + |v| - 2|u \wedge v|$$

Theorem 6

- Let *f* be a partial function mapping Σ^* to Δ^*. *f* is subsequential iff:
  - 1. *f* has bounded variation (according to the metric defined above).
  - 2. for any rational subset Y of Δ^*, $f^{-1}(Y)$ is rational.

Theorem 7

- Let $f = (f_1, \ldots, f_p)$ be a partial function mapping. $Dom(f) \subseteq \Sigma^*$ to $(\Delta^*)^p$ f is *p* –subsequential iff:
  - 1. f has bounded variation (using the metric d on Σ^* and d_∞ on $(\Delta^*)^p$).
  - 2. for all *i* (1<= *i*<= *p*) and any rational subset Y of Δ^*, $f_i^{-1}(Y)$ is rational.

Theorem 8

- Let *f* be a rational function mapping Σ^* to $(\Delta^*)^p$. *f* is *p* -subsequential iff it has bounded variation (using the semi-metric d'_p on $(\Delta^*)^p$).

Application to language processing

- The composition, union,and equivalence algorithms for subsequential transducers are also very useful in many applications.

Representation of very large dictionaries.

- The corresponding representation offers very fast look-up since then the recognition does not depend on the size of the dictionary but only on that of the input string considered.
- As an example, a French morphological dictionary of about 21.2 Mb can be compiled into a p -subsequential transducer of size 1.3 Mb, in a few minutes (Mohri, 1996b).

Compilation of morphological and phonological rules

- Similarly, context-depen-dent phonological and morphological rules can be represented by finite-state transducers (Kaplan and Kay, 1994).
- This increases considerably the time efficiency of the transducer. It can be further minimized to reduce its size.

Syntax

- Finite-state machines are also currently used to represent local syntactic constraints (Silberztein, 1993; Roche, 1993; Karlsson et al., 1995; Mohri, 1994d).
- Linguists can conveniently introduce local grammar transducers that can be used to disambiguate sentences.