Reducing acyclic cover transducers

J.-M. Champarnaud[1], F. Guingne[2] and J. Farré[2]

[1]Laboratoire LITIS, Université de Rouen, France

[2]Laboratoire I3S, Université de Nice - Sophia Antipolis & CNRS, France

Prague, July 16, 2007 / CIAA 2007

Outline

1. An overview
   - State of the art
   - New algorithms
2. Algorithms
   - Algorithm 1
   - Algorithm 2

Introduction

- Cover automata
  - C. Câmpeanu, N. Sântean, and S. Yu, *Minimal Cover-automata for Finite Languages*, Theoret. Comput. Sci. **267** (2001), 3–16.
  - H. Körner, *A Time and Space Efficient Algorithm for Minimizing Cover Automata for Finite Languages*, Int. J. of Foundations of Comput. Sci. **14** (2003), 1071–1086.
- Extension to cover transducers
  - J.-M. Champarnaud, F. Guingne and G. Hansel, *Cover Transducers for Functions with Finite Domain*, Intern. J. of Foundations of Comp. Sc., 16–5(2005), 851–865.
- Motivation
  - compact representation of dictionaries
  - reducing lexicons in NLP

Preliminaries

- subsequential transducer and subsequential function
- longest common prefix and prefix transducer
- right function, k-function and prefix k-function

Subsequential function and subsequential transducer

- A *subsequential transducer* is a tuple $\mathcal{S} = (\Sigma, \Omega, Q, q\_, F, \mathtt{i}, \mathtt{t}, \cdot, \star)$ where:
  - Σ (resp. Ω) is the *input* (resp. *output*) *alphabet*,
  - Q is the finite set of *states*;
  - $q\_ \in Q$ is the *initial state*,
  - $\mathtt{i} \in \Omega^*$ is the *initialization value*
  - $\mathtt{t} : Q \to \Omega^*$ is the *termination function*,
  - $F = \mathrm{dom}(\mathtt{t})$ is the set of *final states*,
  - $\cdot$ is the *transition function*: $Q \times \Sigma \to q \cdot a \in Q$,
  - $\star$ is the *output function*: $Q \times \Sigma \to q \star a \in \Omega^*$.
- A *subsequential function* $S : \Sigma^* \to \Omega^*$ is realized by a subsequential transducer $\mathcal{S}$

$$S(x) = \mathtt{i}(q\_ \star x)\mathtt{t}(q\_ \cdot x)$$

Example of subsequential transducer

dom(α)	α
a	*abba*
ab	*abbaba*
ba	*babba*
aaa	*abbababba*
abb	*abbababa*
bab	*babbaba*
bba	*bbabba*

Tab. 1. The function α ...

Example of subsequential transducer

$\mathrm{dom}(\alpha)$	α
a	*abba*
ab	*abbaba*
ba	*babba*
aaa	*abbababba*
abb	*abbababa*
bab	*babbaba*
bba	*bbabba*

Tab. 1. The function α ...

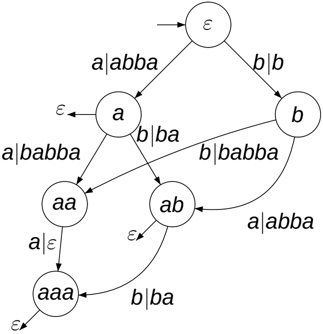
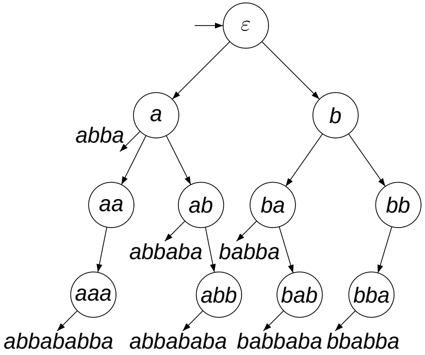
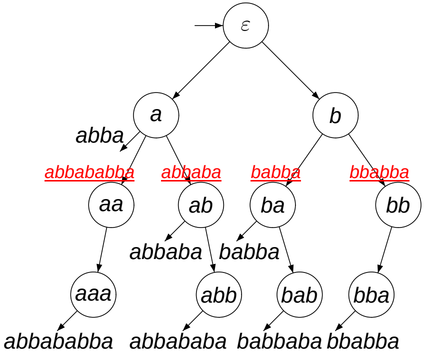


Figure: ... and a subsequential transducer for α.

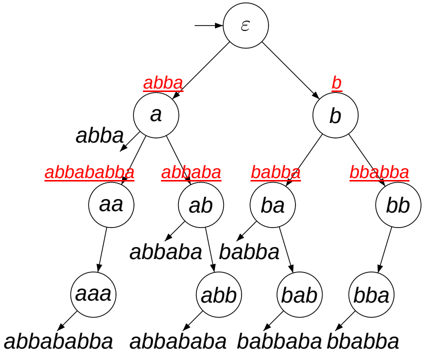
Longest common prefix and prefix transducer



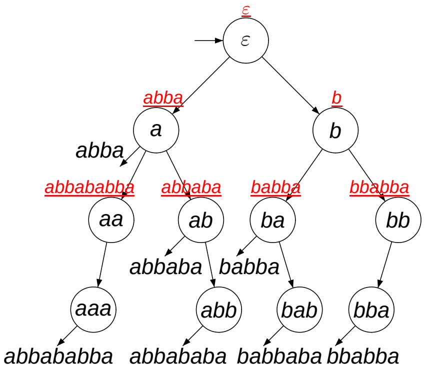
Longest common prefix and prefix transducer



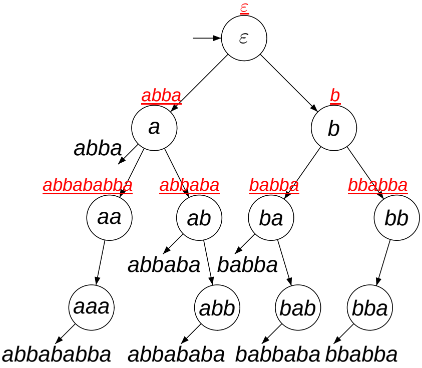
Longest common prefix and prefix transducer



Longest common prefix and prefix transducer



Longest common prefix and prefix transducer



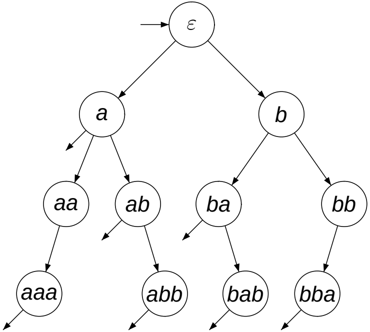
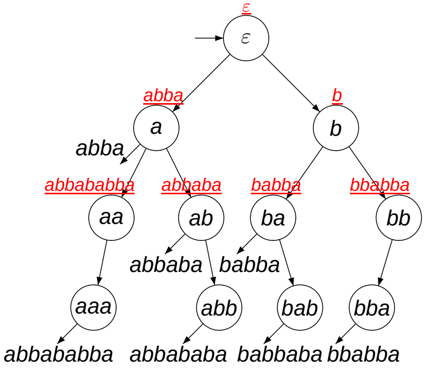


Figure: From the prefix-tree transducer $\mathcal{S}_\alpha$ ...

Longest common prefix and prefix transducer



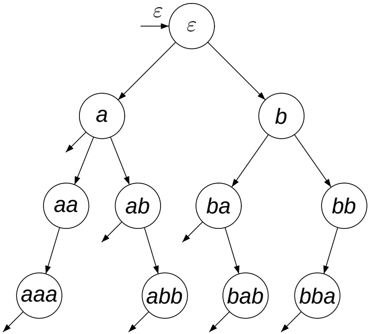


Figure: From the prefix-tree transducer $\mathcal{S}_\alpha$ ...

Longest common prefix and prefix transducer

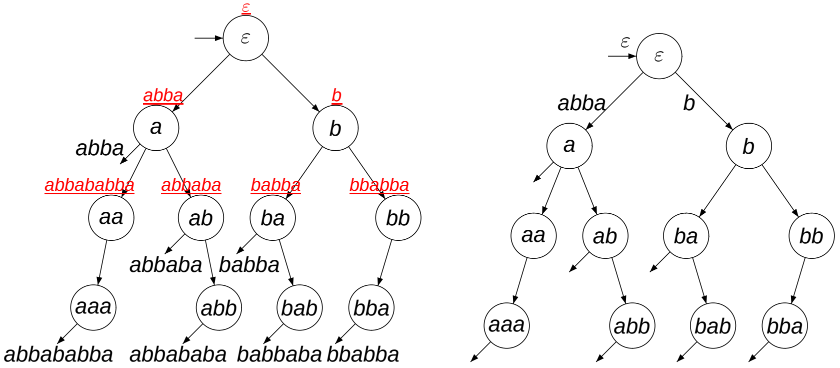
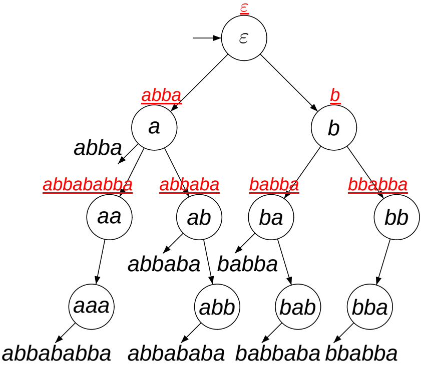


Figure: From the prefix-tree transducer $\mathcal{S}_\alpha$ ...

Longest common prefix and prefix transducer



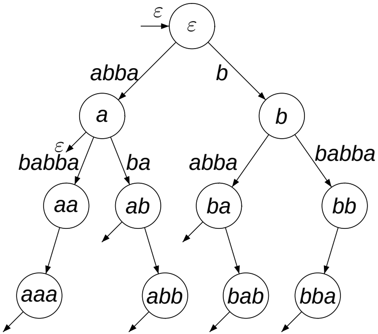
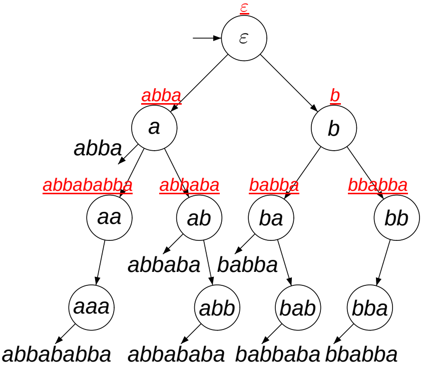


Figure: From the prefix-tree transducer $\mathcal{S}_\alpha$ ...

Longest common prefix and prefix transducer



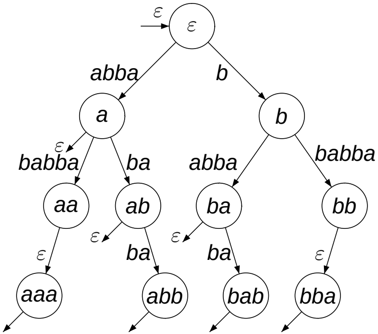


Figure: From the prefix-tree transducer $\mathcal{S}_\alpha$ ...

Longest common prefix and prefix transducer

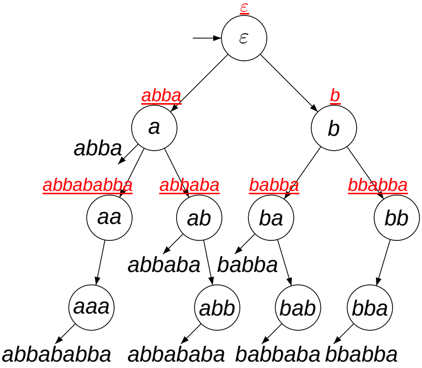


Figure: From the prefix-tree transducer $\mathcal{S}_\alpha$ ...

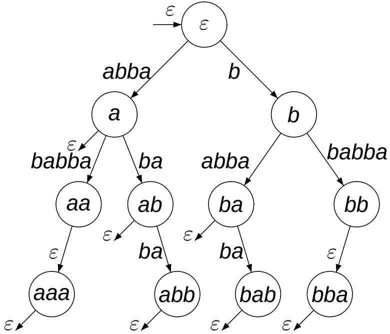


Figure: ... to the prefix transducer $\mathcal{P}_\alpha$.

Right function, k-function and prefix k-function

Right function

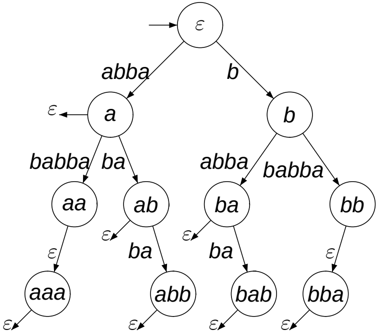


Figure: Prefix-tree transducer.

Right function, k-function and prefix k-function

Right function

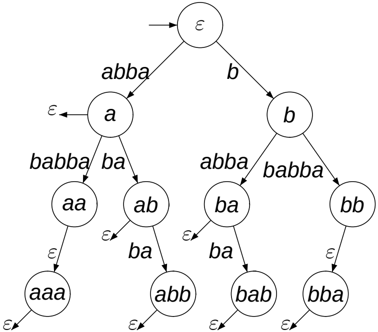


Figure: Prefix-tree transducer.

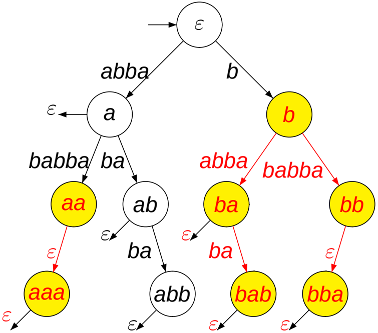


Figure: Right-functions of states *aa* and *b*.

Right function, k-function and prefix k-function

k-function and prefix k-function

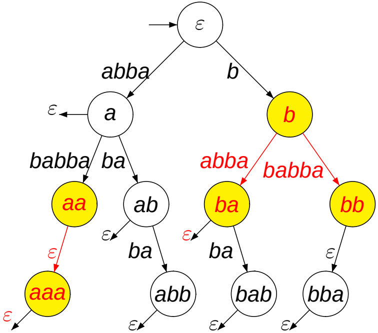


Figure: 1-functions of states *aa* and *b*.

Right function, k-function and prefix k-function

k-function and prefix k-function

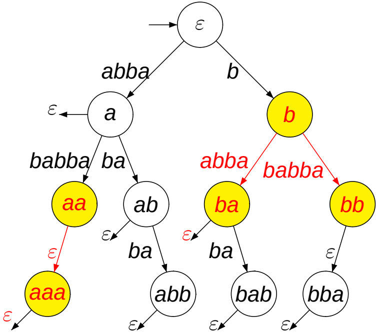


Figure: 1-functions of states *aa* and *b*.

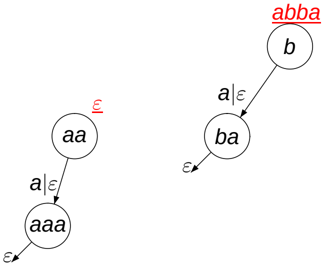


Figure: Prefix 1-functions of states *aa* and *b*.

Overview

- state of the art
  - prefix-tree transducer
  - minimal transducer
  - approximate reduction w.r.t. k-functions
- new algorithms
  - exact reduction w.r.t. k-functions
  - a further reduction via prefix k-functions

Prefix-tree transducer

dom(α)	α
a	*abba*
ab	*abbaba*
ba	*babba*
aaa	*abbababba*
abb	*abbababa*
bab	*babbaba*
bba	*bbabba*

Tab. 1. The function α ...

Prefix-tree transducer

dom(α)	α
a	*abba*
ab	*abbaba*
ba	*babba*
aaa	*abbababba*
abb	*abbababa*
bab	*babbaba*
bba	*bbabba*

Tab. 1. The function α . . .

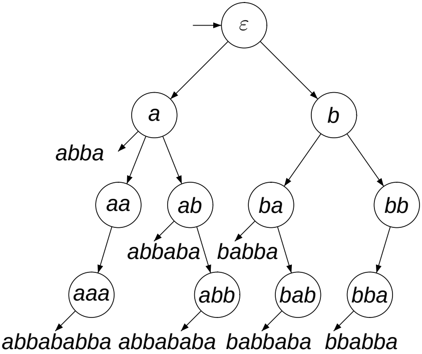


Figure: . . . and its prefix-tree transducer S_α.

Minimal transducer

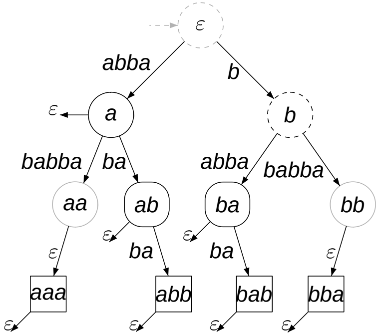


Figure: Partitioning $\mathcal{P}_\alpha$ according to right functions.

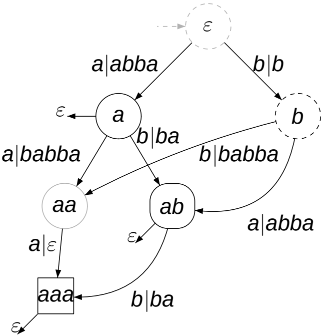


Figure: Minimal transducer $\mathcal{M}_\alpha$.

Approximate reduction w.r.t. k-functions

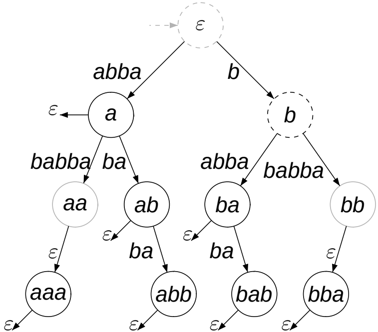


Figure: Approximate k-function partitioning of $\mathcal{P}_\alpha$.

Approximate reduction w.r.t. k-functions

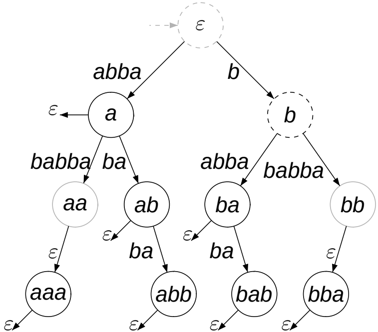


Figure: Approximate k-function partitioning of $\mathcal{P}_\alpha$.

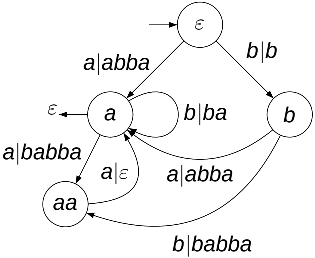


Figure: Reduced cover transducer $\mathcal{R}^1_\alpha$.

Exact reduction w.r.t. k-functions (1)

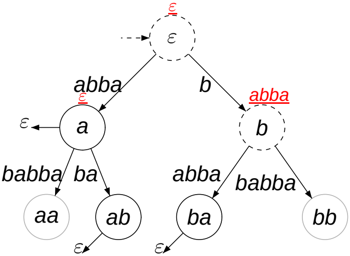


Figure: Construction of the prefix 2-function for state ϵ.

Exact reduction w.r.t. k-functions (1)

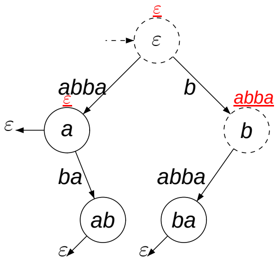


Figure: Construction of the prefix 2-function for state ϵ.

Exact reduction w.r.t. k-functions (1)

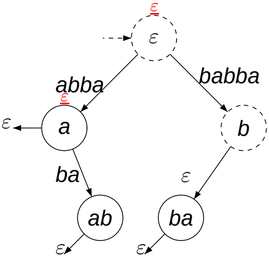


Figure: Prefix 2-function for state ϵ.

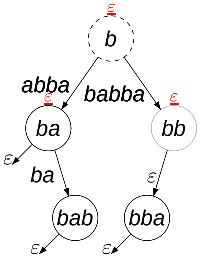


Figure: Prefix 2-function for state b.

Exact reduction w.r.t. k-functions (2)

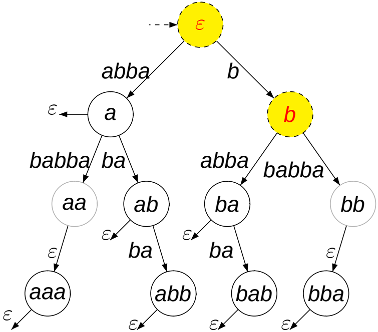


Figure: Exact k-function partitioning of $\mathcal{P}_\alpha$ via prefix k-functions.

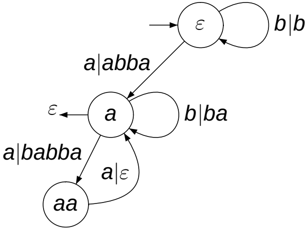


Figure: Reduced cover transducer $\mathcal{R}_\alpha^2$.

A further reduction via prefix k-functions (1)

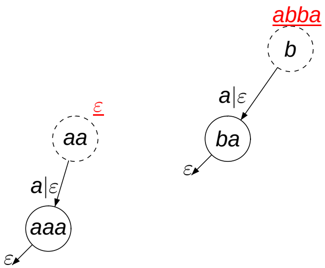


Figure: Prefix 1-functions of states b and aa.

A further reduction via prefix k-functions (1)

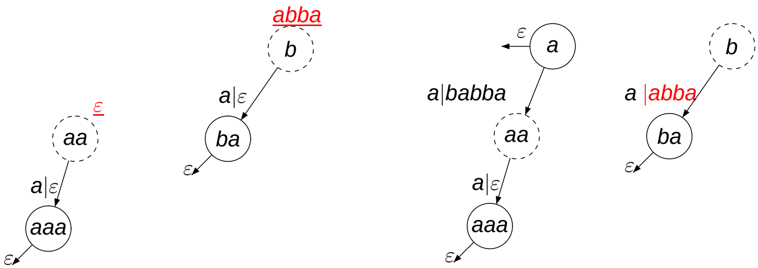


Figure: Prefix 1-functions of states b and aa.

A further reduction via prefix k-functions (1)

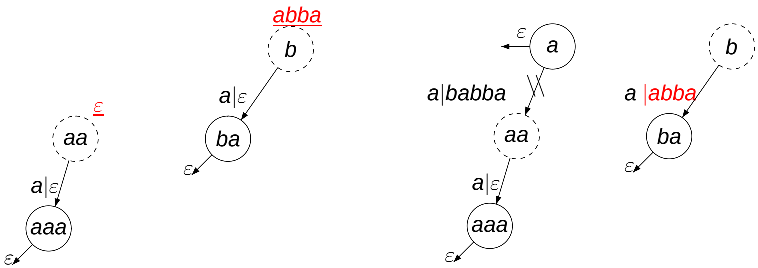


Figure: Prefix 1-functions of states b and aa.

A further reduction via prefix k-functions (1)

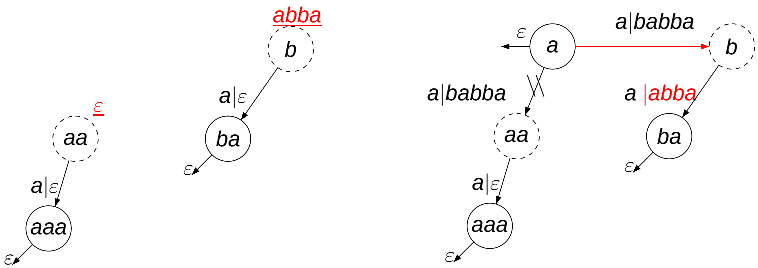


Figure: Prefix 1-functions of states b and aa.

A further reduction via prefix k-functions (1)

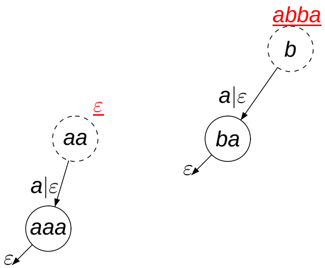


Figure: Prefix 1-functions of states b and aa.

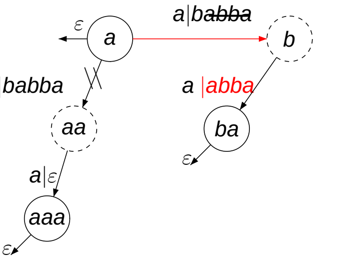


Figure: Merging b and aa in $\mathcal{P}_\alpha$ w.r.t. prefix 1-functions.

A further reduction via prefix k-functions (2)

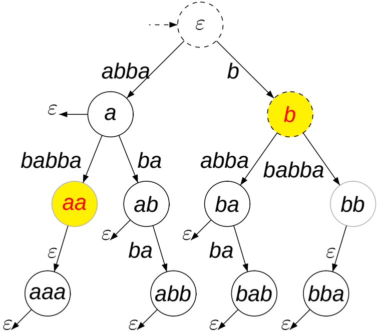


Figure: Exact k-function partitioning of $\mathcal{P}_\alpha$ via prefix k-functions.

A further reduction via prefix *k*-functions (2)

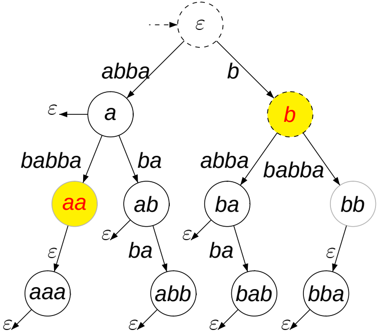


Figure: Exact *k*-function partitioning of $\mathcal{P}_\alpha$ via prefix *k*-functions.

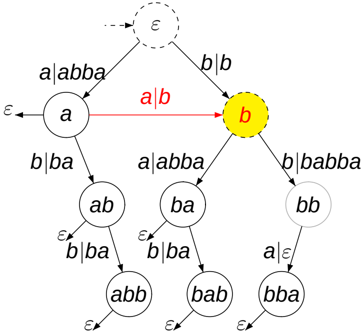


Figure: A further reduction: merging of states *aa* and *b*.

A further reduction via prefix k-functions (3)

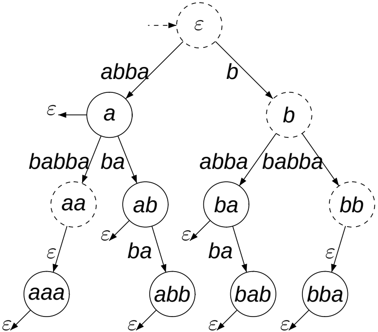


Figure: Partitioning $\mathcal{P}_\alpha$ according to prefix k-functions.

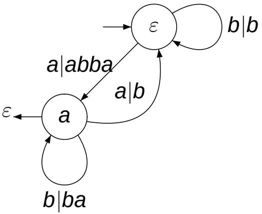


Figure: Reduced cover transducer $\mathcal{R}^3_\alpha$.

Several relations on states

Definitions

Let p, q be two states of Q, and $h = min(height(p), height(q))$

- $p \cong_k q$, $0 \leq k \leq h \iff p$ and q have identical prefix k-functions
- $p \sim_\varepsilon q \iff p$ and q have identical h-functions $\iff$ p and q have identical prefix h-functions and the associated lcps are identical
- $p \sim q \iff p$ and q have identical prefix h-functions and the associated lcp of p is a suffix of the weight of any incoming transition of state q

relations on states

On our Example

	ε	b	aa	bb	a	ab	ba	aaa	abb	bab	bba
$\cong_0$	0	0	0	0	1	1	1	1	1	1	1
$\nu_P(p,0)$	0	0	0	0	ε	ε	ε	ε	ε	ε	ε
$\sim_\varepsilon$	0	0	0	0	1	1	1	1	1	1	1
$\sim$	0	0	0	0	1	1	1	1	1	1	1
$\cong_1$	0	0	0	0	1	1	1				
$\nu_P(p,1)$	$abba$	$abba$	ε	ε	ε	ε	ε				
$\sim_\varepsilon$	2	2	0	0	1	1	1	1	1	1	1
$\sim$	0	0	0	0	1	1	1	1	1	1	1
$\cong_2$	0	0			1						
$\nu_P(p,2)$	ε	ε			ε						
$\sim_\varepsilon$	2	2	0	0	1	1	1	1	1	1	1
$\sim$	0	0	0	0	1	1	1	1	1	1	1

Algorithm 1

1: Input: $\mathcal{P} = (\Sigma, \Omega, Q, q\_, F, \mathtt{i}, \mathtt{t}, \cdot, \star)$, the prefix of the prefix-tree transducer of α.
2: Output: the partition C_k of Q_k w.r.t. $\cong_k$, $\forall 0 \leq k < l$.
3: Comments: $N[p, k]$ is the rank of the class of p in C_k $(0 \leq N[p, k] \leq |C_k| - 1)$.
4: Initializations: $\nu_{\mathcal{P}}(p, 0) = \mathtt{t}[p]$, $\forall p \in Q$; $C_0 = \{Q \setminus F, F\}$.

Algorithm 1

Input: $\mathcal{P} = (\Sigma, \Omega, Q, q\_, F, \mathtt{i}, \mathtt{t}, \cdot, \star)$, the prefix of the prefix-tree transducer of α.
Output: the partition C_k of Q_k w.r.t. $\cong_k$, $\forall 0 \leq k < l$.
Comments: $N[p, k]$ is the rank of the class of p in C_k $(0 \leq N[p, k] \leq |C_k| - 1)$.
Initializations: $\nu_{\mathcal{P}}(p, 0) = \mathtt{t}[p]$, $\forall p \in Q$; $C_0 = \{Q \setminus F, F\}$.
for all $k \in 1 \,..\, l - 1$ **do**
  Computation of the relation $\cong_k$
  for all $C \in C_{k-1}$ **do**
    $C^+ = \{p \in C \mid \mathrm{height}(p) \geq k\}$
    for all $p \in C^+$ **do**
      $\nu_{\mathcal{P}}(p, k) = \bigwedge_{a \in \Sigma}(\mathtt{t[p]}, (\mathtt{p} \star \mathtt{a})\nu_{\mathcal{P}}(\mathtt{p} \cdot \mathtt{a}, \mathtt{k} - 1))$
      Computation of the key keya of p
      IF $\nu_{\mathcal{P}}(p, k) = 0$ THEN $A[p] = 0$; $\forall a \in \Sigma$, $B[p, a] = 0$
      ELSE
      $A[p] = \nu_{\mathcal{P}}(p, k)^{-1}\mathtt{t[p]}$
      for all $a \in \Sigma$ **do**
        $B[p, a] = \nu_{\mathcal{P}}(p, k)^{-1}(p \star a)$
      end for
      FI
      $\mathrm{keya}[p] = ((N[p \cdot a, k - 1])_{a \in \Sigma}, A[p], (B[p, a])_{a \in \Sigma})$
    end for
    $\widehat{C^+} = \mathrm{Partition}(C^+, \mathrm{keya})$
    Insert the blocks of $\widehat{C^+}$ into the set C_k.
  end for
end for

Algorithm 2

1: Initializations: $\nu_{\mathcal{P}}(p, 0) = \mathtt{t}[\mathrm{p}]$, $\forall p \in Q$.
2: Initializations: $D_0 = \{Q \setminus F, F_1, F_2\}$, with $F_1 = \{p \in F \mid \nu_{\mathcal{P}}(p, 0) = \varepsilon\}$ and $F_2 = F \setminus F_1$.
3: **for all** $k \in 1 \,..\, l - 1$ **do**
4:     Computation of the partition D_k

Algorithm 2

Initializations: $\nu_{\mathcal{P}}(p, 0) = \mathtt{t}[\mathtt{p}]$, $\forall p \in Q$.
Initializations: $D_0 = \{Q \setminus F, F_1, F_2\}$, with $F_1 = \{p \in F \mid \nu_{\mathcal{P}}(p, 0) = \varepsilon\}$ and $F_2 = F \setminus F_1$.
for all $k \in 1 \mathinner{..} l - 1$ **do**
    Computation of the partition D_k
    for all $D \in D_{k-1}$ **do**
        IF $D^+ = \emptyset$ THEN Insert D into the partition D_k
        ELSE
        for all $p \in D^+$ **do**
            Computation of a partition of D^+ w.r.t. $\cong_k$

Algorithm 2

Initializations: $\nu_{\mathcal{P}}(p, 0) = \mathtt{t[p]}, \forall p \in Q$.
Initializations: $D_0 = \{Q \setminus F, F_1, F_2\}$, with $F_1 = \{p \in F \mid \nu_{\mathcal{P}}(p, 0) = \varepsilon\}$ and $F_2 = F \setminus F_1$.
for all $k \in 1 \,..\, l - 1$ **do**
    Computation of the partition D_k
    for all $D \in D_{k-1}$ **do**
        IF $D^+ = \emptyset$ THEN Insert D into the partition D_k
        ELSE
        for all $p \in D^+$ **do**
            Computation of a partition of D^+ w.r.t. $\cong_k$
            $\nu_{\mathcal{P}}(p, k) = \bigwedge_{a \in \Sigma}(\mathtt{t[p]}, (\mathtt{p} \star \mathtt{a})\nu_{\mathcal{P}}(\mathtt{p} \cdot \mathtt{a}, \mathtt{k} - 1))$
            $\mathrm{keyb}[p] = \mathrm{Pref}(\nu_{\mathcal{P}}(p, k))$
            Compute $\mathrm{keya}[p]$ according to Lines 11–19 of the Algorithm 1
        end for
        $\widehat{D^+} = \mathrm{Partition}(D^+, \mathrm{keya})$
        $\mathit{linked} = \mathrm{false}$
        for all $C \in \widehat{D^+}$ **do**
            Computation of a partition of C w.r.t. $\sim_\varepsilon$

Algorithm 2

Initializations: $\nu_{\mathcal{P}}(p,0) = \mathtt{t[p]}, \forall p \in Q$.
Initializations: $D_0 = \{Q \setminus F, F_1, F_2\}$, with $F_1 = \{p \in F \mid \nu_{\mathcal{P}}(p,0) = \varepsilon\}$ and $F_2 = F \setminus F_1$.
for all $k \in 1 \,..\, l-1$ **do**
  <u>Computation of the partition D_k</u>
  for all $D \in D_{k-1}$ **do**
    IF $D^+ = \emptyset$ THEN Insert D into the partition D_k
    ELSE
    for all $p \in D^+$ **do**
      <u>Computation of a partition of D^+ w.r.t. $\cong_k$</u>
      $\nu_{\mathcal{P}}(p,k) = \bigwedge_{a \in \Sigma}(\mathtt{t[p]}, (\mathtt{p} \star \mathtt{a})\nu_{\mathcal{P}}(\mathtt{p} \cdot \mathtt{a}, \mathtt{k}-1))$
      $\mathrm{keyb}[p] = \mathrm{Pref}(\nu_{\mathcal{P}}(p,k))$
      Compute $\mathrm{keya}[p]$ according to Lines 11–19 of the Algorithm 1
    end for
    $\widehat{D^+} = \mathrm{Partition}(D^+, \mathrm{keya})$
    linked $= \mathrm{false}$
    for all $C \in \widehat{D^+}$ **do**
      <u>Computation of a partition of C w.r.t. $\sim_\varepsilon$</u>
      IF $\exists p \in C \mid \mathrm{height}(p) = k$
      THEN $\widehat{C} = \mathrm{Partition}(C, \mathrm{keyb})$
          $\widehat{C} = \{E_1, E_2\}$, with $E_2 = \{p \mid \nu_{\mathcal{P}}(p,k) \succ \varepsilon\}$
          IF $\neg$*linked* THEN $E_1 = E_1 \cup D^-$; *linked* $= \mathrm{true}$ FI
          Insert E_1 into the partition D_k
      ELSE IF $\neg$*linked* THEN $C = C \cup D^-$; *linked* $= \mathrm{true}$ FI
          Insert C into the partition D_k
      FI; **end for**; FI; **end for**; **end for**

Complexity

– $s = sizeof(int)$,
– $t_{max} = \max_{p \in Q}\{\mathrm{length}(\mathtt{t}[p])\}$
– $k_{max} = (|\Sigma| + 1)t_{max} + |\Sigma| + s$

– l is the order of the subsequential function,
– n is the number of states of the transducer,

The Algorithm 2 computes a minimal partition of Q w.r.t. the relation $\sim_\varepsilon$ in $O(k_{max}nl)$ time.

Conclusion

- improvement of a previous algorithm for reducing cover transducers
- further work:
  - develop experimental study
  - find heuristics to compute a partionning as small as possible w.r.t the relation $\sim$
  - extend this study to possibly cyclic cover transducers