Deciding the density type of a given regular language

Stavros Konstantinidis   Joshua Young

Department of Mathematics and Computing Science,
Saint Mary's University, Halifax, Nova Scotia, Canada

Prague Stringology Conference, 2013

Outline

1. Introduction/Previous Work
   - Density of a Language
   - Regular Language Given Via DFA

2. Regular language given via NFA
   - Regular language given via NFA
   - Direct Algorithm
   - Linear Time Algorithm
   - Implementation and Testing

Outline

Density of a Langauge

- The density of a language L is the function that returns, for every nonnegative integer n, the number of words in L of length n

Density of a Langauge

- The density of a language L is the function that returns, for every nonnegative integer n, the number of words in L of length n
- We say that a regular language L has *exponential density* if the density of L is not polynomially upper-bounded

Problem

- Given a regular language L, decide whether L is of exponential density.

Outline

1. Introduction/Previous Work
   - Density of a Language
   - Regular Language Given Via DFA
2. Regular language given via NFA
   - Regular language given via NFA
   - Direct Algorithm
   - Linear Time Algorithm
   - Implementation and Testing

Regular language given via DFA

- A regular language L has exponential density if and only if any trim deterministic automaton accepting L has a state that belongs to two different cycles.

A. Shur: Combinatorial complexity of rational languages. Discr. Anal. and Oper. Research,Ser. 1, 12 2005, pp. 78–99.

Regular language given via DFA

- A regular language L has exponential density if and only if any trim deterministic automaton accepting L has a state that belongs to two different cycles.
- This leads to a linear time algorithm for deciding whether a regular language is of exponential density when L is given via a *deterministic* finite automaton (DFA)

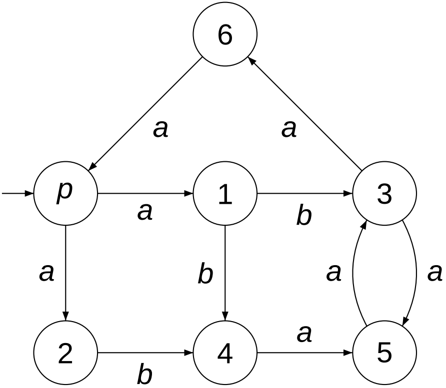
A. Shur: Combinatorial complexity of rational languages. Discr. Anal. and Oper. Research,Ser. 1, 12 2005, pp. 78–99.

Outline

Regular language given via NFA

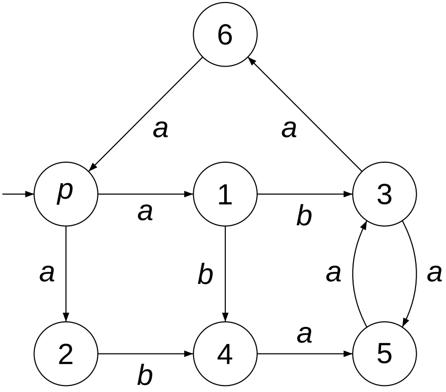
- A regular language L has exponential density if and only if any trim nondeterministic automaton accepting L has a strongly connected component containing two walks of the same length, starting at the same state, and whose labels are different.

Example

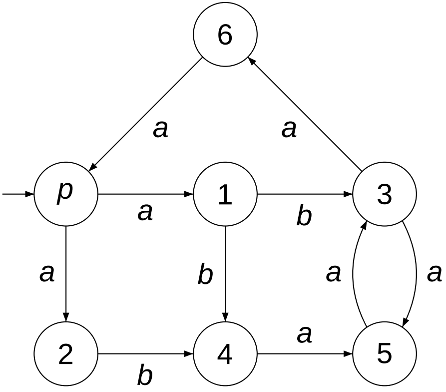


(p,a,1,b,3,a,6,a,p,a,2,b,4) and (p,a,2,b,4,a,5,a,3,a,5,a,3).

Proof

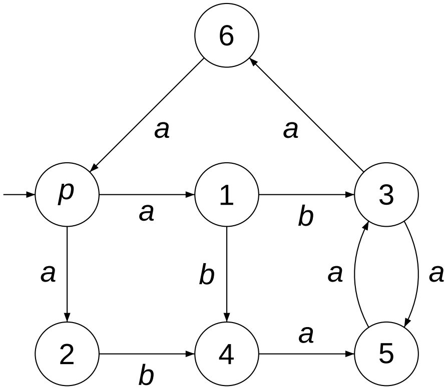


Proof



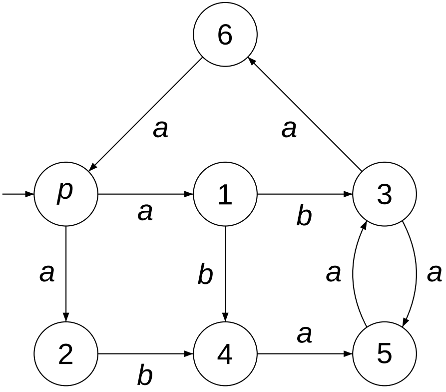
- p to q_1 and p to q_2, same length with *different* labels u_1, u_2

Proof



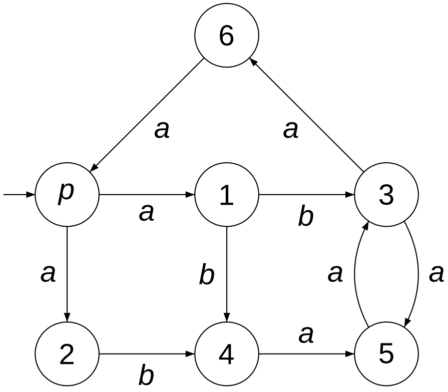
- p to q_1 and p to q_2, same length with *different* labels u_1, u_2
- q_1 to p and q_2 to p with labels v_1, v_2

Proof



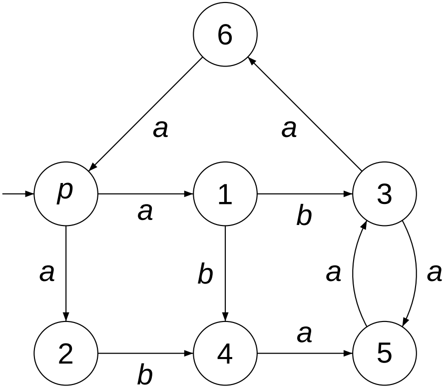
- p to q_1 and p to q_2, same length with *different* labels u_1, u_2
- q_1 to p and q_2 to p with labels v_1, v_2
- p to p with labels u_1v_1 and u_2v_2

Proof



- p to q_1 and p to q_2, same length with *different* labels u_1, u_2
- q_1 to p and q_2 to p with labels v_1, v_2
- p to p with labels u_1v_1 and u_2v_2
- p to p with labels $z_1 = u_1v_1u_2v_2$ and $z_2 = u_2v_2u_1v_1$, same length

Proof



- p to q_1 and p to q_2, same length with *different* labels u_1, u_2
- q_1 to p and q_2 to p with labels v_1, v_2
- p to p with labels u_1v_1 and u_2v_2
- p to p with labels $z_1 = u_1v_1u_2v_2$ and $z_2 = u_2v_2u_1v_1$, same length
- $C = \{z_1, z_2\}$, $xC^ny \subseteq L$.

Outline

Terminology

- *Product Construction*: for $G = (V, E)$ the graph G^2 has vertices all pairs in $V \times V$ and arcs all triples of the form $((p_1, p_2), (a_1, a_2), (q_1, q_2))$ such that (p_1, a_1, q_1) and (p_2, a_2, q_2) are arcs in E.

Terminology

- *Product Construction*: for $G = (V, E)$ the graph G^2 has vertices all pairs in $V \times V$ and arcs all triples of the form $((p_1, p_2), (a_1, a_2), (q_1, q_2))$ such that (p_1, a_1, q_1) and (p_2, a_2, q_2) are arcs in E.
- For any walk in G^2 there are two corresponding walks in G of the same length and, conversely, for any two walks in G of the same length there is a corresponding walk in G^2.

Algorithm

algorithm ExpDensityQT(p)

01. Make the NFA A trim
02. Compute the SCCs of A
03. FOUND = false
04. for each SCC G and while not FOUND
    05. Compute G^2
    06. Compute the set Q_1 of vertices (p_1, p_2) in G^2 such that there is an arc $((p_1, p_2), (a_1, a_2), (q_1, q_2))$ with $a_1 \neq a_2$
    07. Compute the set Q_2 of vertices in G^2 of the form (t, t)
    08. if (there is a walk from Q_2 to Q_1) then FOUND = true
09. if (FOUND) **return** TRUE, else **return** FALSE

Complexity

algorithm ExpDensityQT(p)
01. Make the NFA A trim (linear time)
02. Compute the SCCs of A (linear time)
03. FOUND = false
04. for each SCC G and while not FOUND ($O(n_1^2 + \cdots + n_k^2)$)
    05. Compute G^2 ($O(n_i^2)$)
    06. Compute the set Q_1 of vertices (p_1, p_2) in G^2 such that there is an arc $((p_1, p_2), (a_1, a_2), (q_1, q_2))$ with $a_1 \neq a_2$
    07. Compute the set Q_2 of vertices in G^2 of the form (t, t)
    08. if (there is a walk from Q_2 to Q_1) then FOUND = true
09. if (FOUND) **return** TRUE, else **return** FALSE

Outline

Terminology

- $\gcd(\mathcal{C})$ denotes the *greatest common divisor* of the lengths of all cycles in $\mathcal{C}$.

Terminology

- $\gcd(\mathcal{C})$ denotes the *greatest common divisor* of the lengths of all cycles in $\mathcal{C}$.
- A state q in $\mathcal{C}$ *occurs at level* i, for some $i \in \mathbb{N}_0$ (when starting at state p), if there is a walk of length i from p to q.

Terminology

- $\gcd(\mathcal{C})$ denotes the *greatest common divisor* of the lengths of all cycles in $\mathcal{C}$.
- A state q in $\mathcal{C}$ *occurs at level* i, for some $i \in \mathbb{N}_0$ (when starting at state p), if there is a walk of length i from p to q.
- For each $i \in \mathbb{N}$, $A_p(i)$ denotes the *set of symbols at level* i, that is, all symbols σ such that there is a transition (q, σ, r) in $\mathcal{C}$ and state q occurs at level $i - 1$.

Terminology

- $\gcd(\mathcal{C})$ denotes the *greatest common divisor* of the lengths of all cycles in $\mathcal{C}$.
- A state q in $\mathcal{C}$ *occurs at level* i, for some $i \in \mathbb{N}_0$ (when starting at state p), if there is a walk of length i from p to q.
- For each $i \in \mathbb{N}$, $A_p(i)$ denotes the *set of symbols at level* i, that is, all symbols σ such that there is a transition (q, σ, r) in $\mathcal{C}$ and state q occurs at level $i - 1$.
- $\mathcal{C}$ has exponential density if and only if there is a level i such that $A_p(i)$ contains more than one symbol.

Algorithm

algorithm BFS(p)
01. for each state q, set $\mathrm{LEV}_p(q) = ?$
02. for each $i \in \{1, \ldots, N\}$, set $\mathbf{b}_p(i) = ?$
03. Initialize a queue Q to consist of p
04. set $\mathrm{LEV}_p(p) = 0$
05. while (Q is not empty)
  06. remove q, the first state in Q
  07. for each transition (q, σ, r)
    08. set $j = \mathrm{LEV}_p(q)$
    09. if $\mathbf{b}_p(j+1) \neq ?$ and $\mathbf{b}_p(j+1) \neq \sigma$, **return** λ
    10. set $\mathbf{b}_p(j+1) = \sigma$
    11. if $(\mathrm{LEV}_p(r) = ?)$
      set $\mathrm{LEV}_p(r) = j + 1$
      append r to Q
12. Let k be the last index such that $\mathbf{b}_p(k) \neq ?$
13. **return** the word $\mathbf{b}_p(1) \cdots \mathbf{b}_p(k)$

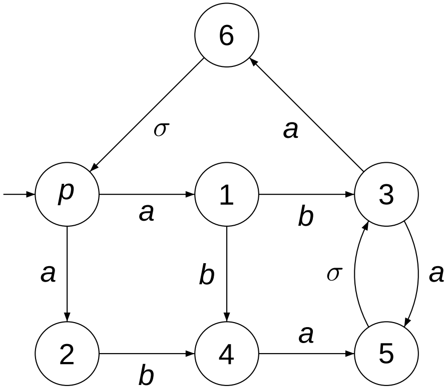
Algorithm

- There is a level i_0 such that $A_p(i_0)$ contains more than one symbol, if and only if, either that level is found by BFS(p), or the word $\mathbf{b}_p$ is *not* periodic with a period of length gcd($\mathcal{C}$).

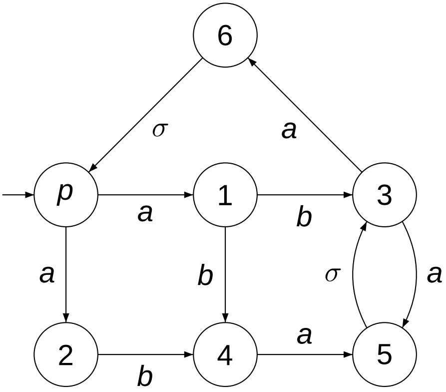
Algorithm

algorithm ExpDensity(p)
1. Let $\mathbf{b}_p$ = BFS(p)
2. if ($\mathbf{b}_p = \lambda$) **return** TRUE
3. Let g = the gcd of the cycles in the SCC
4. Let $v = \mathbf{b}_p(1) \cdots \mathbf{b}_p(g)$
5. if ($\mathbf{b}_p \notin \mathrm{Prefix}(v^*)$) **return** TRUE
6. else **return** FALSE

Example

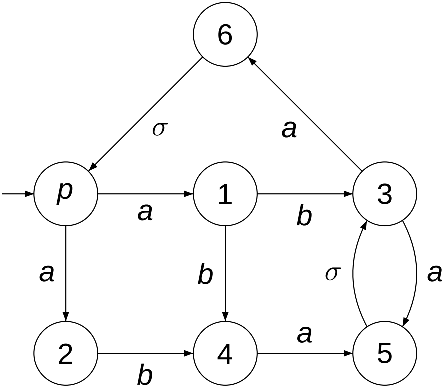


Example



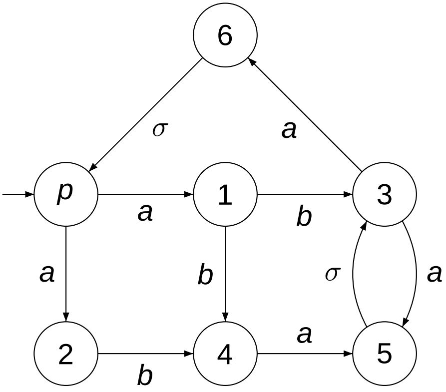
- $\gcd(\mathcal{C}) = 2$.

Example



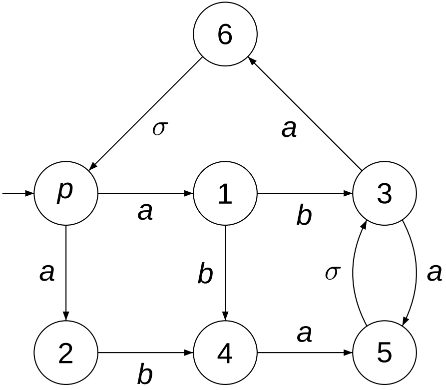
- $\gcd(\mathcal{C}) = 2$.
- $A_p(4) = \sigma$.

Example



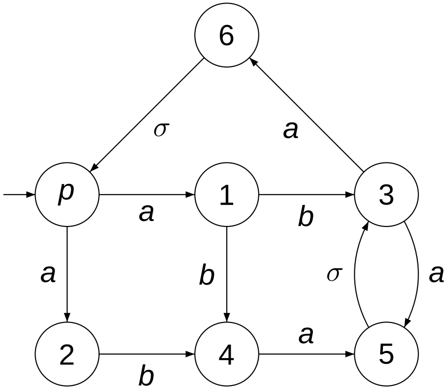
- $\gcd(\mathcal{C}) = 2$.
- $A_p(4) = \sigma$.
- $\mathbf{b}_p = aba\sigma$,

Example



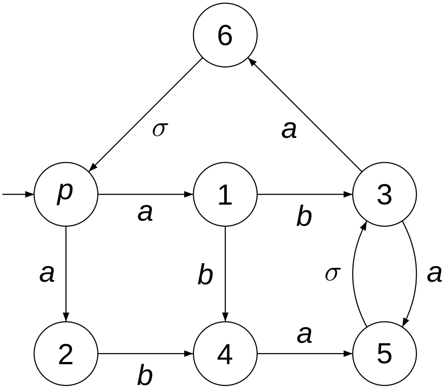
- $\gcd(\mathcal{C}) = 2$.
- $A_p(4) = \sigma$.
- $\mathbf{b}_p = aba\sigma$,
- $\mathbf{b}_p(1)\mathbf{b}_p(2) = ab$

Example



- $\gcd(\mathcal{C}) = 2$.
- $A_p(4) = \sigma$.
- $\mathbf{b}_p = aba\sigma$,
- $\mathbf{b}_p(1)\mathbf{b}_p(2) = ab$
- if $\sigma = a$ then $abaa \notin \mathrm{Prefix}((ab)^*)$.

Example



- $\gcd(\mathcal{C}) = 2$.
- $A_p(4) = \sigma$.
- $\mathbf{b}_p = aba\sigma$,
- $\mathbf{b}_p(1)\mathbf{b}_p(2) = ab$
- if $\sigma = a$ then $abaa \notin \mathrm{Prefix}((ab)^*)$.
- if $\sigma = b$ then $abab \in \mathrm{Prefix}((ab)^*)$.

Complexity

algorithm ExpDensity(p)
1. Let $\mathbf{b}_p$ = BFS(p) (linear time)
2. if ($\mathbf{b}_p = \lambda$) **return** TRUE
3. Let g = the gcd of the cycles in the SCC (linear time)
4. Let $v = \mathbf{b}_p(1) \cdots \mathbf{b}_p(g)$
5. if ($\mathbf{b}_p \notin \mathrm{Prefix}(v^*)$) **return** TRUE (linear time)
6. else **return** FALSE

Outline

Implementation

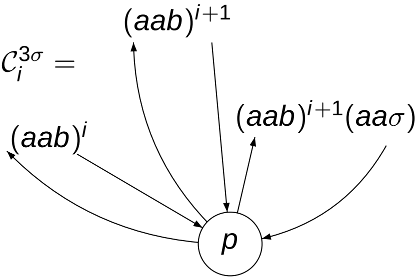
- Both the quadratic and linear time algorithms implemented using the FAdo library for automata (http://fado.dcc.fc.up.pt/)

Implementation

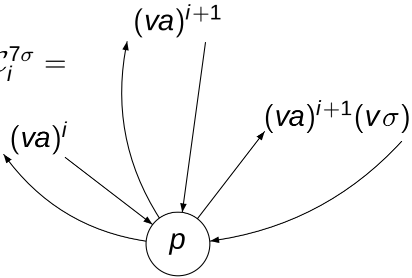
- Both the quadratic and linear time algorithms implemented using the FAdo library for automata (http://fado.dcc.fc.up.pt/)
- BFS algorithm is adjusted to compute $\gcd(\mathcal{C})$ in addition to the word $\mathbf{b}_p(1)\cdots\mathbf{b}_p(k)$.

Test Cases

$\mathcal{C}_i^{3\sigma} =$

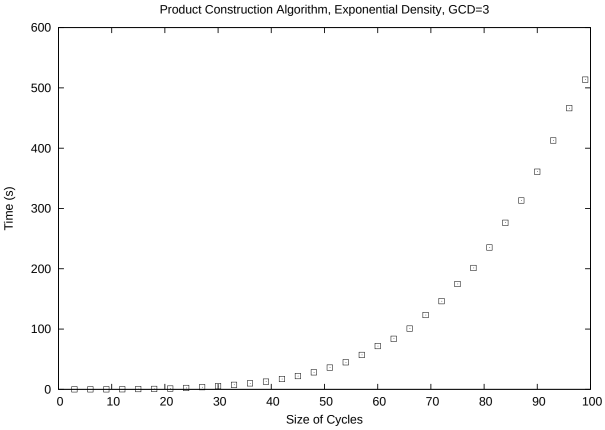


$\mathcal{C}_i^{7\sigma} =$

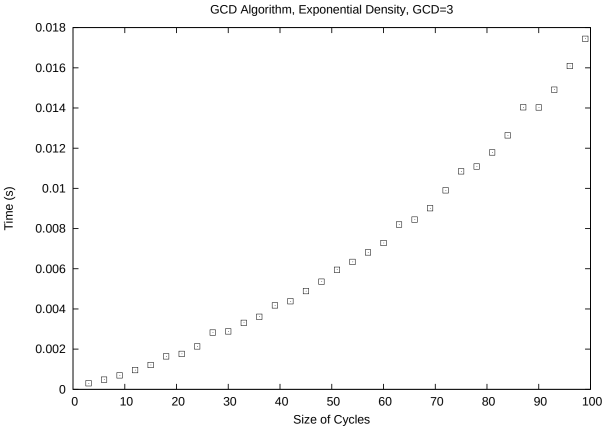


$$v = aabbab$$

Results


Product Construction Algorithm, Exponential Density, GCD=3
Time (s)
Size of Cycles
0
100
200
300
400
500
600
10
20
30
40
50
60
70
80
90
100

Results


GCD Algorithm, Exponential Density, GCD=3
Time (s)
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
10
20
30
40
50
60
70
80
90
100
Size of Cycles

Summary

- A regular language L has exponential density if and only if any trim nondeterministic automaton accepting L has a strongly connected component containing two walks of the same length, starting at the same state, and whose labels are different.

Summary

- A regular language L has exponential density if and only if any trim nondeterministic automaton accepting L has a strongly connected component containing two walks of the same length, starting at the same state, and whose labels are different.
- Direct quadratic time algorithm for deciding the density type of a Regular language given via NFA

Summary

- A regular language L has exponential density if and only if any trim nondeterministic automaton accepting L has a strongly connected component containing two walks of the same length, starting at the same state, and whose labels are different.
- Direct quadratic time algorithm for deciding the density type of a Regular language given via NFA
- Fast linear time algorithm for deciding the density type of a Regular language given via NFA