Succinctness of the Complement and Intersection of Regular Expressions

Wouter Gelade and Frank Neven

Hasselt University and transnational University of Limburg

February 21, 2008
Introduction

Regular Languages and Expressions

- One of the most fundamental concepts of (theoretical) computer science.
- Applications: Pattern matching, XML, ....
Introduction

Regular Languages and Expressions
- One of the most fundamental concepts of (theoretical) computer science.
- Applications: Pattern matching, XML, ....

Succinctness
- Regular expressions to automata.
- Operations on automata (state complexity).
Introduction

Regular Languages and Expressions

- One of the most fundamental concepts of (theoretical) computer science.
- Applications: Pattern matching, XML, ... 

Succinctness

- Regular expressions to automata.
- Operations on automata (state complexity).
- Automata to regular expressions?
- Operations on regular expressions???
Main Questions

Questions

Given regular expressions $r, r_1, \ldots, r_n$ over an alphabet $\Sigma$, what is the complexity of constructing a regular expression defining

- $\Sigma^* \setminus L(r)$, i.e., the complement of $r$; or,
Questions

Given regular expressions $r, r_1, \ldots, r_n$ over an alphabet $\Sigma$, what is the complexity of constructing a regular expression defining

- $\Sigma^* \setminus L(r)$, i.e., the complement of $r$; or,
- $L(r_1) \cap \cdots \cap L(r_n)$, i.e., the intersection of $r_1$ to $r_n$. 
Questions

Given regular expressions \( r, r_1, \ldots, r_n \) over an alphabet \( \Sigma \), what is the complexity of constructing a regular expression defining

- \( \Sigma^* \setminus L(r) \), i.e., the complement of \( r \); or,
- \( L(r_1) \cap \cdots \cap L(r_n) \), i.e., the intersection of \( r_1 \) to \( r_n \).

Answer

It’s double exponential!
Outline

1. Complement

2. Intersection

3. Restricted Classes of Regular Expressions
Proposition

Given a regular expression $r$, a regular expression $s$ defining $\Sigma^* \setminus L(r)$ can be constructed in time double exponential in the size of $r$. 
Given a regular expression $r$, a regular expression $s$ defining $\Sigma^* \setminus L(r)$ can be constructed in time double exponential in the size of $r$.

**Algorithm**

Given a regular expression $r$:

- Construct an NFA $A$ with $L(A) = L(r)$. (polynomial)
- Construct a DFA $B$ with $L(B) = \Sigma^* \setminus L(A)$. (exponential)
- Construct a RE $s$ with $L(s) = L(B) = \Sigma^* \setminus L(r)$. (exponential)
For every $n \in \mathbb{N}$, there is a regular expression $r_n$ of size $O(n)$ such that any regular expression $r$ defining $\Sigma^* \setminus L(r_n)$ is of size at least $2^{2^n}$. 
A Language by Ehrenfeucht and Zeiger

**Definition**

For every $n \in \mathbb{N}$, let $Z_n$ be defined by the complete DFA on $n$ states with

- only initial and final states; and
- a different label on every edge. ($\Sigma_n = \{a_{i,j} | 0 \leq i, j < n\}$)

**Example: $Z_3$**
Definition
For every $n \in \mathbb{N}$, let $Z_n$ be defined by the complete DFA on $n$ states with
- only initial and final states; and
- a different label on every edge. ($\Sigma_n = \{a_{i,j} \mid 0 \leq i, j < n\}$)

Example: $a_{1,0}a_{0,2}a_{2,2} \in Z_3$
Theorem [Ehrenfeucht and Zeiger 1976]

- Any regular expression defining $Z_n$ must be of size at least $2^{n-1}$.
- There is a DFA of size $\mathcal{O}(n^2)$ accepting $Z_n$. 

Complement

Lower Bound: Proof Sketch
Theorem [Ehrenfeucht and Zeiger 1976]

- Any regular expression defining $Z_n$ must be of size at least $2^{n-1}$.
- There is a DFA of size $\mathcal{O}(n^2)$ accepting $Z_n$.

Corollary

In the translation from DFAs to regular expressions, an exponential blow-up can not be avoided.
Theorem [Ehrenfeucht and Zeiger 1976]
- Any regular expression defining $Z_n$ must be of size at least $2^{n-1}$.
- There is a DFA of size $O(n^2)$ accepting $Z_n$.

Corollary
Any regular expression defining $Z_{2^n}$ must be of size at least $2^{2n-1}$. 
# Lower Bound: Proof Sketch

## Theorem [Ehrenfeucht and Zeiger 1976]
- Any regular expression defining $Z_n$ must be of size at least $2^{n-1}$.
- There is a DFA of size $O(n^2)$ accepting $Z_n$.

## Corollary
Any regular expression defining $Z_{2^n}$ must be of size at least $2^{2^n-1}$.

## End of Proof
Construct regular expression of size $O(n)$ defining $\Sigma^* \setminus Z_{2^n}$.
Complement

Lower Bound: Proof Sketch

Theorem [Ehrenfeucht and Zeiger 1976]
- Any regular expression defining $Z_n$ must be of size at least $2^{n-1}$.
- There is a DFA of size $O(n^2)$ accepting $Z_n$.

Corollary
Any regular expression defining $Z_{2^n}$ must be of size at least $2^{2n-1}$.

End of Proof
Construct regular expression of size $O(n)$ defining $\Sigma^* \setminus Z_{2^n}$.

Problem
The alphabet of $Z_{2^n}$ is of size $(2^n)^2$. 
Complement

Lower Bound: Proof Sketch

Binary Encoding of $\mathbb{Z}_n$

For every $a_{i,j} \in \Sigma_n$ define

$$\rho_n(a_{i,j}) = \text{enc}(j)\text{enc}(i)\#,$$

where $\text{enc}(i)$ and $\text{enc}(j)$ are the $\lceil \log(n) \rceil$-bit encodings of $i$ and $j$.

Extend $\rho_n$ to strings as $\rho_n(a_{i_0,i_1} \cdots a_{i_{k-1},i_k}) = \rho_n(a_{i_0,i_1}) \cdots \rho_n(a_{i_{k-1},i_k})$. 

Example: $w = a_{0,2}a_{2,1}a_{1,3} \in \mathbb{Z}_4$ and thus, $\rho_n(w) = 10\$00\#01\$10\#11\$01\# \in K_4$. 

W. Gelade (Hasselt University)

Succinctness of Regular Expressions

February 21, 2008 11 / 27
Complement

Lower Bound: Proof Sketch

Binary Encoding of $\mathbb{Z}_n$
For every $a_{i,j} \in \Sigma_n$ define

$$\rho_n(a_{i,j}) = \text{enc}(j)\text{enc}(i)\#,$$

where $\text{enc}(i)$ and $\text{enc}(j)$ are the $\lceil \log(n) \rceil$-bit encodings of $i$ and $j$.

Extend $\rho_n$ to strings as

$$\rho_n(a_{i_0,i_1} \cdots a_{i_{k-1},i_k}) = \rho_n(a_{i_0,i_1}) \cdots \rho_n(a_{i_{k-1},i_k}).$$

The Language $K_n$: Definition

$$K_n = \{\rho_n(w) \mid w \in \mathbb{Z}_n\}$$ (over the alphabet $\Sigma = \{0, 1, $, $\#\}$).
Complement

Lower Bound: Proof Sketch

Binary Encoding of $\mathbb{Z}_n$
For every $a_{i,j} \in \Sigma_n$ define

$$\rho_n(a_{i,j}) = \text{enc}(j)\text{enc}(i)\#,$$

where $\text{enc}(i)$ and $\text{enc}(j)$ are the $\lceil \log(n) \rceil$-bit encodings of $i$ and $j$. Extend $\rho_n$ to strings as $\rho_n(a_{i_0,i_1} \cdots a_{i_{k-1},i_k}) = \rho_n(a_{i_0,i_1}) \cdots \rho_n(a_{i_{k-1},i_k})$.

The Language $K_n$: Definition

$K_n = \{ \rho_n(w) \mid w \in \mathbb{Z}_n \}$ (over the alphabet $\Sigma = \{0, 1, $, $\#\}$).

Example

1. $w = a_{0,2}a_{2,1}a_{1,3} \in \mathbb{Z}_4$ and thus,
   $$\rho_n(w) = 10$00\#01$10\#11$01\# \in K_4.$$

W. Gelade (Hasselt University) Succinctness of Regular Expressions February 21, 2008 11 / 27
Lower Bound: Proof Sketch

**Theorem**

1. Any regular expression defining $K_n$ is of size at least $2^n$.
2. There is a DFA $A_n$ of size $\mathcal{O}(n^2 \log n)$ defining $K_n$. 

**Corollary**

In the translation from DFAs to regular expressions, an exponential blow-up cannot be avoided, even when the alphabet is fixed.
Lower Bound: Proof Sketch

**Theorem**

1. Any regular expression defining $K_n$ is of size at least $2^n$.
2. There is a DFA $A_n$ of size $O(n^2 \log n)$ defining $K_n$.

**Corollary**

In the translation from DFAs to regular expressions, an exponential blow-up can not be avoided, even when the alphabet is fixed.
Corollary

For $n \in \mathbb{N}$, any regular expression defining $K_{2^n}$ must be of size at least $2^{2n}$. 
Corollary

For $n \in \mathbb{N}$, any regular expression defining $K_{2^n}$ must be of size at least $2^{2^n}$.

Defining the Complement of $K_{2^n}$

- Expression is disjunction of expressions capturing all mistakes in a string. For instance:
Corollary

For $n \in \mathbb{N}$, any regular expression defining $K_{2n}$ must be of size at least $2^{2n}$.

Defining the Complement of $K_{2n}$

- Expression is disjunction of expressions capturing all mistakes in a string. For instance:

- String does not end with #: $\Sigma^*(0 + 1 + \#)$.
Corollary

For $n \in \mathbb{N}$, any regular expression defining $K_{2^n}$ must be of size at least $2^{2^n}$.

Defining the Complement of $K_{2^n}$

- Expression is disjunction of expressions capturing all mistakes in a string. For instance:
- String does not end with '#': $\Sigma^*(0 + 1 + \$)$.
- String has two corresponding bits which are not equal ($10\$00\#01\$10\#11\$00\#)$:

Corollary

For $n \in \mathbb{N}$, any regular expression defining $K_{2n}$ must be of size at least $2^{2n}$.

Defining the Complement of $K_{2n}$

- Expression is disjunction of expressions capturing all mistakes in a string. For instance:
  - String does not end with $\#$: $\Sigma^*(0 + 1 + \$$).
  - String has two corresponding bits which are not equal
    $(10$00$\#01$10$\#11$00$\#)$:
      $((0 + 1)^* + \Sigma^* \#(0 + 1)^*) 1\Sigma^{3n+2} 0\Sigma^* + \ldots$
    - $\ldots$
Outline

1. Complement
2. Intersection
3. Restricted Classes of Regular Expressions
Proposition

Let \( r_1, \ldots, r_n \) be regular expressions. A regular expression \( r \) defining \( \bigcap_{i \leq n} L(r_i) \) can be constructed in time double exponential in the size of \( r_1 \) to \( r_n \).
Intersection

An arbitrary number of expressions: Upper bound

**Proposition**

Let \( r_1, \ldots, r_n \) be regular expressions. A regular expression \( r \) defining \( \bigcap_{i \leq n} L(r_i) \) can be constructed in time **double exponential** in the size of \( r_1 \) to \( r_n \).

**Theorem**

Let \( n \in \mathbb{N} \). There exist expressions \( r_1, \ldots, r_m \), each of size \( O(n) \), such that any regular expression defining \( \bigcap_{i \leq m} L(r_i) \) is of size at least \( 2^{2^n} \).
An arbitrary number of expressions: Upper bound

**Proposition**

Let $r_1, \ldots, r_n$ be regular expressions. A regular expression $r$ defining $\bigcap_{i \leq n} L(r_i)$ can be constructed in time double exponential in the size of $r_1$ to $r_n$.

**Theorem**

Let $n \in \mathbb{N}$. There exist expressions $r_1, \ldots, r_m$, each of size $\mathcal{O}(n)$, such that any regular expression defining $\bigcap_{i \leq m} L(r_i)$ is of size at least $2^{2^n}$.

**Proof Idea**

- Construct expressions describing properties any string in (a variant of) $K_{2^n}$ must have.
- Variant of $K_{2^n}$ is defined by intersection of expressions.
Intersection

A fixed number of expressions

Upper bound
For any fixed $k \in \mathbb{N}$, let $r_1, \ldots, r_k$ be regular expressions. A regular expression defining $\bigcap_{i \leq k} L(r_i)$ can be constructed in time exponential in the sizes of $r_1$ to $r_k$.

Lower Bound: Theorem
For every $n \in \mathbb{N}$, there are regular expressions $r_n$ and $s_n$ of size $O(n^2)$ such that any regular expression defining $L(r_n) \cap L(s_n)$ is of size at least $2^{n-1}$.
Outline

1. Complement

2. Intersection

3. Restricted Classes of Regular Expressions
Deterministic (or one-unambiguous) Regular Expressions

**Definition**

A regular expression $r$ is **deterministic** if

- when matching any string from left to right against $r$, we can deterministically match every symbol against a position in $r$, without looking ahead in the string; or,
Deterministic (or one-unambiguous) Regular Expressions

Definition

A regular expression $r$ is deterministic if

- when matching any string from left to right against $r$, we can deterministically match every symbol against a position in $r$, without looking ahead in the string; or,

- the Glushkov construction translates it into a DFA [Bruggeman-Klein and Wood, 1998].
Deterministic (or one-unambiguous) Regular Expressions

**Definition**
A regular expression $r$ is deterministic if
- when matching any string from left to right against $r$, we can deterministically match every symbol against a position in $r$, without looking ahead in the string; or,
- the Glushkov construction translates it into a DFA [Bruggeman-Klein and Wood, 1998].

**Example**
- $(a + b)^* a$ is not deterministic (Counterexample $baa$).
- $(b^* a)(b^* a)^*$ is deterministic
Deterministic (or one-unambiguous) Regular Expressions

XML

Regular expressions in XML schema languages DTD and XML Schema are required to be deterministic.

Properties [Bruggemann-Klein and Wood, 1998]

- Not every regular language is definable by a deterministic regular expression.
- Not closed under complement or intersection.
Complement

Given a deterministic regular expression $r$, a regular expression $s$ defining its complement can be constructed in polynomial time.
Operations On Deterministic Regular Expressions

Complement

Given a deterministic regular expression \( r \), a regular expression \( s \) defining its complement can be constructed in polynomial time.

Proof idea

- Translation to DFA is polynomial \( \Rightarrow \) naive algorithm is exponential.
- Do not use translation to automata.
- Immediately construct regular expression defining complement of Glushkov automaton.
Intersection

Constructing a regular expression for deterministic expressions is as hard as for normal regular expressions:

- double-exponential for an arbitrary number of expressions.
- exponential for a fixed number of expressions.
Single Occurrence Regular Expressions (SOREs)

Definition

A **SORE** is a regular expressions in which every alphabet symbol occurs at most once.

Example

- 
  \((a + b)^* c\) is a SORE.
- 
  \((a + b)^* a\) is not a SORE.
**Definition**

A **SORE** is a regular expressions in which every alphabet symbol occurs at most once.

**Example**

- \((a + b)^* c\) is a SORE.
- \((a + b)^* a\) is not a SORE.

**Properties**

- Every SORE is a deterministic regular expression.
- SOREs are the regular expressions used in practical XML schemas [Bex et al. 2004].
Intersection of SOREs

Proposition

Let \( r_1, \ldots, r_n \) be SOREs. A regular expression \( r \) defining \( \bigcap_{i \leq n} L(r_i) \) can be constructed in time \textit{exponential} in the size of \( r_1 \) to \( r_n \).
Proposition
Let \( r_1, \ldots, r_n \) be SOREs. A regular expression \( r \) defining \( \bigcap_{i \leq n} L(r_i) \) can be constructed in time exponential in the size of \( r_1 \) to \( r_n \).

Theorem
For every \( n \in \mathbb{N} \), there exist SOREs \( r_n, s_n \) of size \( O(n^2) \) such that any regular expression defining \( L(r_n) \cap L(s_n) \) is of size at least \( 2^{n-1} \).
Conclusion

- Taking complement or intersection of expressions is hard.
- Intersection remains hard, even for very simple subclasses.
- Does there exist a useful fragment of the regular expressions for which taking intersection also becomes PTIME?