Regular Expressions Enumeration and State Complexity

Jeffrey Shallit
School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada
shallit@graceland.uwaterloo.ca
http://www.cs.uwaterloo.ca/~shallit

Two Problems

This talk has three sections:

1. Enumeration of regular expressions and the languages they represent (joint work with N. Rampersad and J. Lee)
2. State complexity of some restricted classes of regular expressions.
3. Open problems on regular expressions.

Regular Expressions

- By a regular expression, I mean a string over the alphabet

$$\Sigma \cup \{+, *, (,), \epsilon, \emptyset\}$$

  that represents a regular language

- Example: (0+10)*(1+ϵ) represents the language of all strings over $\{0, 1\}$ that do not contain two consecutive 1's

- Formal definition usually omitted, or given inaccurately
  - Is the empty expression valid?
  - Is () valid?
  - Is a** valid?

Regular Expressions

Definition from Martin [2003]:

"The set R of regular languages over Σ, and the corresponding expressions, are defined as follows:

1. $\emptyset$ is an element of R, and the corresponding regular expression is $\emptyset$.

2. $\{\Lambda\}$ is an element of R, and the corresponding regular expression is Λ.

3. For each $a \in \Sigma$, $\{a\}$ is an element of R and the corresponding regular expression is a.

4. If L_1 and L_2 are any elements of R, and r_1 and r_2 are the corresponding regular expressions,

(a) $L_1 \cup L_2$ is an element of R and the corresponding regular expression is $(r_1 + r_2)$;

(b) $L_1 L_2$ is an element of R, and the corresponding regular expression is $(r_1 r_2)$;

(c) L_1^* is an element of R, and the corresponding regular expression is (r_1^*). "

Length of Regular Expressions

In order to enumerate regular expressions, we need a definition of length.

- *Ordinary length*: total number of symbols, including parentheses, $\emptyset$, ϵ, etc., counted with multiplicity.
  - (0+10)*(1+ϵ) has ordinary length 12
- *Reverse polish length*: number of symbols in a reverse polish equivalent, including a symbol • for concatenation
- Equivalently, number of nodes in a syntax tree for the expression
  - (0+10)*(1+ϵ) in reverse polish would be 010•+*ϵ+•
  - This has reverse polish length 10
- *Alphabetic length*: number of symbols from Σ, not including ϵ, $\emptyset$, parens, operators
  - (0+10)*(1+ϵ) has alphabetic length 4

Length of Regular Expressions

Theorem.

Provided a regular expression does not have

- unnecessary parentheses;
- iterated $*$;
- ϵX or $X\epsilon$;
- $\epsilon + X$ or $X + \epsilon$ where $\epsilon \in L(X)$

the three measures

- ordinary length
- reverse polish length
- alphabetic length

are equivalent, up to a constant multiplicative factor.

Defining Regular Expressions with a Grammar

Example 1.

$$
\begin{aligned}
S &\rightarrow E_+ \mid E_\bullet \mid G \\
E_+ &\rightarrow E_+ + F \mid F + F \\
F &\rightarrow E_\bullet \mid G \\
E_\bullet &\rightarrow E_\bullet G \mid GG \\
G &\rightarrow E_* \mid C \mid P \\
C &\rightarrow \emptyset \mid \epsilon \mid a \quad (a \in \Sigma) \\
E_* &\rightarrow G * \\
P &\rightarrow (S)
\end{aligned}
$$

- gives a formal definition of regular expressions
- is unambiguous
- allows for enumeration

Enumeration Problems

- Counting distinct languages accepted by DFA's or NFA's with n states is hard!
- Some results are known
- Letting $G_1(n)$ denote the number of distinct languages accepted by NFA's with n states over a unary alphabet
- Best known bounds are

$$2^{n+(2.295-o(1))\sqrt{\frac{n}{\log n}}} < G_1(n) \leq \left(\frac{c_1 n}{\log n}\right)^n$$

- How about regular expressions and regular expression languages?

Enumeration with Context-Free Grammars

Theorem. (Chomsky-Schützenberger)
Let $L \subseteq \Sigma^*$ be a context-free language defined by an unambiguous grammar G. Then the associated formal power series

$$L(x) := \sum_{n \geq 0} |L \cap \Sigma^n| x^n$$

is algebraic over $\mathbb{Q}(x)$. Furthermore, the equation satisfied by $L(x)$ can be deduced by solving the system of equations given by the *commutative interpretation* of G.

The *commutative interpretation* of G is the system of equations obtained from G by

- replacing $\rightarrow$ by $=$
- replacing $\mid$ by $+$
- replacing terminals by x

Enumeration with Context-Free Grammars

Example. The commutative interpretation of

$$\begin{aligned}
S &\to E_+ \mid E_\bullet \mid G \\
E_+ &\to E_+ + F \mid F + F \\
F &\to E_\bullet \mid G \\
E_\bullet &\to E_\bullet G \mid GG \\
G &\to E_* \mid C \mid P \\
C &\to \emptyset \mid \epsilon \mid a \quad (a \in \Sigma) \\
E_* &\to G * \\
P &\to (S)
\end{aligned}$$

is

$$\begin{aligned}
S &= E_+ + E_\bullet + G \\
E_+ &= E_+ F x + F^2 x \\
F &= E_\bullet + G \\
E_\bullet &= E_\bullet G + G^2 \\
G &= E_* + C + P \\
C &= (k+2)x \quad (k = |\Sigma|) \\
E_* &= Gx \\
P &= Sx^2
\end{aligned}$$

Enumeration with Context-Free Grammars

We can now solve this system for the variable S. One way to do so is to use Gröbner bases. We eliminate the variables using a lexicographic order, with S last in the ordering.

For the grammar of all regular expressions over an alphabet of size k, this is

$$(x^2+x^3)S^2+((k+3)x^2+(k+3)x-1)S+(k+2)x = 0.$$

We can expand this as a power series to get a generating function enumerating the regular expressions of length n. For example, for $k = 2$, we have

$$S = 4x+20x^2+120x^3+716x^4+4356x^5+26880x^6+\cdots.$$

The 20 regular expressions of length 2 are

- yz
- $y*$

where $y, z \in \{\epsilon, \emptyset, \mathtt{a}, \mathtt{b}\}$.

Asymptotics with Context-Free Grammars

We can obtain the asymptotics for the coefficients as follows:

- Compute the discriminant. (This is a generalization of the expression $b^2 - 4ac$ for the polynomial $ax^2 + bx + c$.)
- Determine the smallest positive real root r of this polynomial.
- The coefficient r_n of x^n of the generating function is $O(r^{-n})$.
- More detailed analysis of the coefficients is sometimes possible using Darboux's method.

Asymptotics with Context-Free Grammars

In our example, for $k = 2$, the equation

$$(x^3 + x^2)S^2 + (5x^2 + 5x - 1)S + 4x$$

has discriminant

$$9x^4 + 34x^3 + 15x^2 - 10x + 1$$

with real roots

$$\begin{aligned} -3.127689322538 \\ -1.036751658241 \\ 0.137551275774 \\ 0.249111927228 \end{aligned}$$

We have $1/.137551275774 = 7.2700161767$, so $r_n = O(7.28^n)$.

Doing This With Maple

```
with(Groebner);
# loads the Groebner basis package

gp := [S-(EP+ED+G), EP-(EP*F*x+F*F*x), ...
# define the system of equations

tt := gbasis(gp,plex(EP,F,ED,G,C,ES,P,S));
# find a Groebner basis, using pure
# lexicographic order, eliminate S last

uu := collect(tt[1], S);
# collect the coefficients of the
# first element of tt

gg := discrim(uu, S);
# compute the discriminant

fsolve(gg=0,x);
# find real roots of the discriminant
```

Enumeration of Regular Expressions

Example 2: counting regular expressions without redundant parentheses

$$
\begin{aligned}
S &\to E_+ \mid E_\bullet \mid E_* \mid C \\
E_+ &\to E_+ + F \mid F + F \\
F &\to E_\bullet \mid E_* \mid C \\
E_\bullet &\to E_\bullet G \mid GG \\
G &\to (E_+) \mid E_* \mid C \\
C &\to \emptyset \mid \epsilon \mid a \quad (a \in \Sigma) \\
E_* &\to (E_+)* \mid (E_\bullet)* \mid E_** \mid C*
\end{aligned}
$$

When $k = 2$, this gives

$$(x^9 + x^6)S^4 + (2x^6 + 5x^5 + x^3)S^3 +$$
$$(5x^4+5x^3+5x^2-x)S^2+(8x^2+5x-1)S+4x = 0$$

The discriminant is a polynomial of degree 30.
The smallest positive root is $.146378001$.
The asymptotic growth rate is $O(6.832^n)$.

Enumeration of Regular Expressions

Example 3: counting regular expressions without redundant parentheses, without nested *

$$
\begin{aligned}
S &\to E_+ \mid E_\bullet \mid E_* \mid C \\
E_+ &\to E_+ + F \mid F + F \\
F &\to E_\bullet \mid E_* \mid C \\
E_\bullet &\to E_\bullet G \mid GG \\
G &\to (E_+) \mid E_* \mid C \\
C &\to \emptyset \mid \epsilon \mid a \quad (a \in \Sigma) \\
E_* &\to (E_+)* \mid (E_\bullet)* \mid C*
\end{aligned}
$$

The equation for $k = 2$ is

$$
\begin{aligned}
&(x^9 + x^7 + x^6)S^4 + (6x^6 + 5x^5 + x^4 + x^3)S^3 + \\
&(4x^5+9x^4+9x^3+4x^2-x)S^2+(8x^3+12x^2+4x-1)S \\
&+4x^2 + 4x = 0.
\end{aligned}
$$

The discriminant is a polynomial of degree 33.
The smallest positive root is $.148870451896$.
This gives an asymptotic growth rate of $O(6.71725^n)$.

Enumeration of Regular Expression Languages

Example 4: counting regular expressions

- without redundant parens
- without nested *
- where $\emptyset$ only appears by itself
- no ϵX or $X\epsilon$ or ϵ^*
- no $\epsilon + X^*$
- if ϵ appears in sum, it is first term

$$
\begin{aligned}
S &\to E_+ \mid E_\bullet \mid E_* \mid \emptyset \mid \epsilon \mid a \quad (a \in \Sigma) \\
E_\bullet &\to E_\bullet R \mid RR \\
R &\to (E_+) \mid E_* \mid a \quad (a \in \Sigma) \\
E_* &\to (E_+)^* \mid (E_\bullet)^* \mid a^* \quad (a \in \Sigma) \\
E_+ &\to \epsilon + X \mid Y + Z \\
X &\to T \mid T + X \\
T &\to E_\bullet \mid a \quad (a \in \Sigma) \\
Y &\to Z \mid Y + Z \\
Z &\to E_\bullet \mid E_* \mid a \quad (a \in \Sigma)
\end{aligned}
$$

The equation and discriminant are too big to give here!

The smallest positive root is $.225437150475$, so the growth rate is $O(4.436^n)$.

For unary regular expressions, we get a bound of $O(3.134^n)$.

These have recently been improved to $O(4.29^n)$ and $O(3.01^n)$ by Jon Lee.

Enumeration of Regular Expression Languages

Now we move to lower bounds.

Evidently we can obtain k^n different languages by taking as the regular expression the k^n strings of length n, over an alphabet of size k.

To improve this bound, we can create unambiguous grammars generating many different regular expressions, each generating a different language.

We start with the grammar

$$\begin{aligned} S &\rightarrow Y \mid Y(\epsilon + S) \\ Y &\rightarrow aY \mid a \qquad (a \in \Sigma) \end{aligned}$$

The idea in this grammar is to generate expressions using Horner's rule.

The corresponding commutative interpretation

$$\begin{aligned} S &= Y + YSx^4 \\ Y &= kxY + kx \end{aligned}$$

has solution

$$S = -\frac{kx}{kx^5 + kx - 1}.$$

For $k = 2$ this gives an asymptotic growth rate of $\Omega(2.102^n)$ and for $k = 1$ we get $\Omega(1.324^n)$.

Enumeration of Regular Expression Languages

My student Jonathan Lee has improved this grammar to generate all finite languages, factoring out common prefixes. Here is his grammar for $\Sigma = \{0, 1\}$:

$$
\begin{aligned}
S &\rightarrow \emptyset \mid T \mid \epsilon \mid \epsilon + T \\
T &\rightarrow Y \mid Z \\
Y &\rightarrow 0 \mid 1 \mid 0E \mid 1E \\
E &\rightarrow Y \mid (Z) \mid (\epsilon + T) \\
Z &\rightarrow 0 + 1 \mid 0E + 1 \mid 0 + 1E \mid 0E + 1E
\end{aligned}
$$

The commutative interpretation is

$$
\begin{aligned}
S &= 2x + (1 + x^2)T \\
T &= Y + Z \\
Y &= 2x(1 + E) \\
E &= Y + Zx^2 + Tx^4 \\
Z &= x^3(E + 1)^2
\end{aligned}
$$

This gives a growth rate of $\Omega(2.567^n)$ in the binary case.

In the unary case we get $\Omega(1.380^n)$.

State Complexity

- The *state complexity* of a regular language L is defined to be the number of states in the minimal DFA accepting L.
- We write it as $\mathrm{sc}(L)$.
- Many researchers on this topic, including Meyer & Fischer, Yu, K. Salomaa, etc.

State Complexity of RE to DFA Conversion

- Suppose we measure the length of a regular expression by its alphabetic length. What is the worst-case state complexity in going from a RE to a DFA?
- Worst case known is

$$(0^*(01^*)^{r-1}0)^*,$$

  which has alphabetic length $2r$ and state complexity $\frac{5}{4}2^r$
- Thus a blow-up of $\frac{5}{4}2^{n/2}$.
- Best upper bound known is $2^n + 1$.

Restricted Regular Expressions

We can look at certain restricted classes of regular expressions. For example,

- expressions without $+$ (union)
  - expressions of the form $w_1^* w_2^* \cdots w_k^*$
  - expressions of the form $w_0 x_1^* w_1 x_2^* \cdots x_k^* w_k$
- expressions without $*$ (Kleene closure)
  - expressions for uniform languages (every string has the same length)
- expressions without $\cdot$ (concatenation), allowing concatenation between elements of Σ only
  - expressions of the form $(w_1 + w_2 + \cdots + w_k)^*$
- unambiguous expressions
- expressions that generate $\Sigma^* - \{w\}$

Of these, $(w_1 + w_2 + \cdots + w_k)^*$ is one of the most interesting.

The Frobenius Problem

- The Frobenius problem asks: given a set of positive integers

  $$\{a_1, a_2, \ldots, a_k\}$$

  with

  $$a_1 < a_2 < \cdots < a_k$$

  and

  $$\gcd(a_1, a_2, \ldots, a_k) = 1,$$

  find the largest integer $G(a_1, a_2, \ldots, a_k)$ which is *not* a non-negative integer linear combination of the a_i.
- Example: the "chicken nuggets problem": $G(6, 9, 20) = 43$.
- Simple closed form known for $k = 2$
- Fast, simple algorithm for $k = 3$
- An impractical polynomial-time algorithm for fixed k
- NP-hard in the general case

The Frobenius Problem and State Complexity

- The relationship between the Frobenius number and state complexity is as follows: if

$$\gcd(a_1, a_2, \ldots, a_k) = 1,$$

then

$$\mathrm{sc}((0^{a_1}+0^{a_2}+\cdots+0^{a_k})^*) = G(a_1, a_2, \ldots, a_k)+2.$$

- Many bounds for G known; one of the simplest is

$$G(a_1, \ldots, a_k) < a_k^2.$$

- This bound implies that the state complexity of $(0^{a_1}, 0^{a_2}, \ldots, 0^{a_k})^*$ is $O(a_k^2)$.
- Furthermore, this bound is tight, since $G(n-1, n) = n^2 - 3n + 1$.
- What about state complexity for larger alphabets?
- This can be considered a version of the "non-commutative Frobenius problem".

The Non-Commutative Frobenius Problem

- Find a tight upper bound for

$$\mathrm{SC}((w_1 + w_2 + \cdots + w_k)^*)$$

  in terms of the alphabetic length

$$n = |w_1| + \cdots + |w_k|.$$

- An upper bound is

$$\mathrm{SC}((w_1 + w_2 + \cdots + w_k)^*) \leq 2^{n-k+1}.$$

- For a long time it was thought that $O(n^2)$ was the "true" bound.
- However, recently I found a class of examples that achieves a state complexity of $2^{\Omega(\sqrt{n})}$.

The Non-Commutative Frobenius Problem

Let t be an integer ≥ 2, and define strings as follows:

$$y := 01^{t-1}0$$
$$x_i := 1^{t-i-1}01^{i+1}, \quad 0 \leq i \leq t-2 \ .$$

Let $S_t := \{0, x_0, x_1, \ldots, x_{t-2}, y\}$.

Thus, for example,

$$S_6 := \{0, 1111101, 1111011, 1110111,$$
$$1101111, 1011111, 0111110\}.$$

Theorem. S_t^* has state complexity $3t2^{t-2} + 2^{t-1}$.

The proof of this theorem is rather complicated, so we just sketch a proof of the following slightly weaker result:

Theorem. $\mathrm{sc}(S_t^*) \geq 2^{t-2}$.

The Non-Commutative Frobenius Problem

Proof. First, we create an NFA M_t with $3t-1$ states that accepts S_t^*. This NFA has states

$$Q = \{p_0, p_1, \ldots, p_t, q_1, q_2, \ldots, q_{t-1}, r_1, r_2, \ldots, r_{t-1}\}$$

with only one final state $F = \{p_0\}$.

For example, here is the NFA M_6.

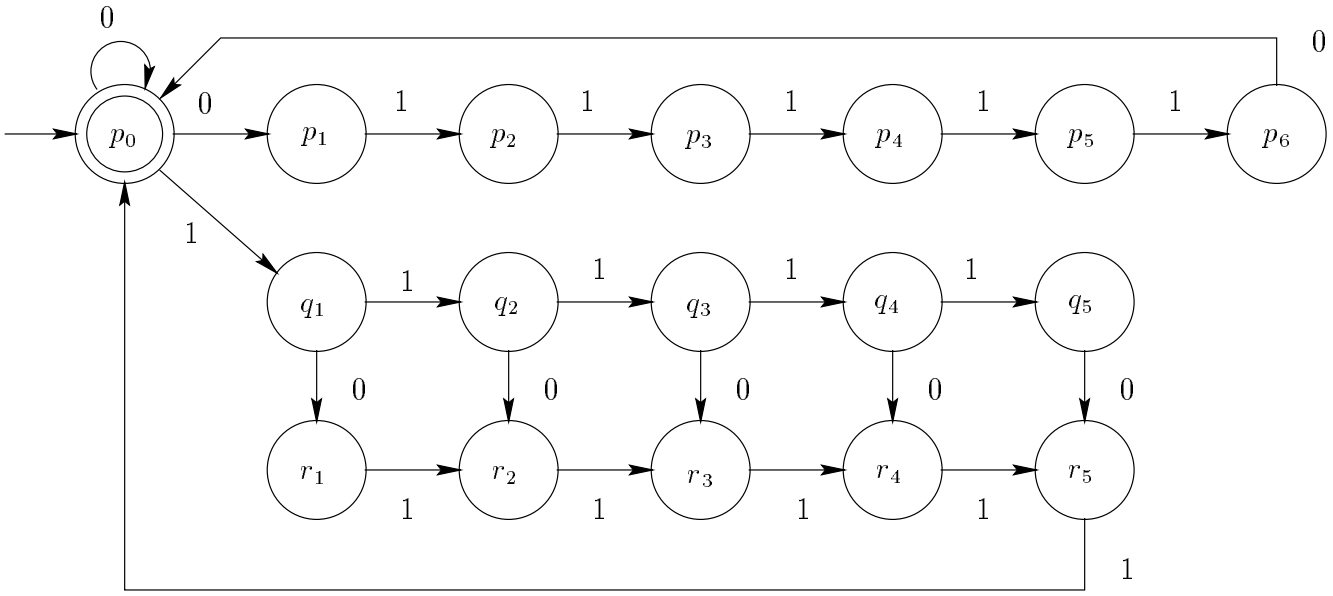

Figure 1: The NFA M_6

The Non-Commutative Frobenius Problem

Let T be any subset of $\{r_1, r_2, \ldots, r_{t-2}\}$, and write $T = \{r_{i_1}, r_{i_2}, \ldots, r_{i_j}\}$ for j indices

$$1 \leq i_1 < i_2 < \cdots < i_j \leq t-2.$$

We claim that the 2^{t-2} strings

$$y \; x_{t-2}y \; x_{t-3}x_{t-2}y \; x_{t-4}x_{t-3}x_{t-2}y$$

$$\cdots \; x_1x_2\cdots x_{t-2}y \; x_{i_1}x_{i_2}\cdots x_{i_j}y$$

where

$$1 \leq i_1 < i_2 < \cdots < i_j \leq t-2$$

are pairwise inequivalent under the Myhill-Nerode equivalence relation.

The Non-Commutative Frobenius Problem

To show this, we first argue that any subset of states of the form $T' := \{p_0, r_{t-1}\} \cup T$, where T is as in the previous paragraph, is reachable from p_0. I claim that the following path reaches T':

$$\{p_0\} \xrightarrow{y} \{p_0, r_{t-1}\} \xrightarrow{x_{t-2}y} \{p_0, r_{t-2}, r_{t-1}\}$$
$$\xrightarrow{x_{t-3}x_{t-2}y} \{p_0, r_{t-3}, r_{t-2}, r_{t-1}\}$$
$$\xrightarrow{x_{t-4}x_{t-3}x_{t-2}y} \cdots$$
$$\xrightarrow{x_1x_2\cdots x_{t-2}y} \{p_0, r_1, r_2, \ldots, r_{t-1}\}$$
$$\xrightarrow{x_{i_1}x_{i_2}\cdots x_{i_j}y} \{p_0, r_{i_1}, r_{i_2}, \ldots, r_{i_j}, r_{t-1}\}.$$

The Non-Commutative Frobenius Problem

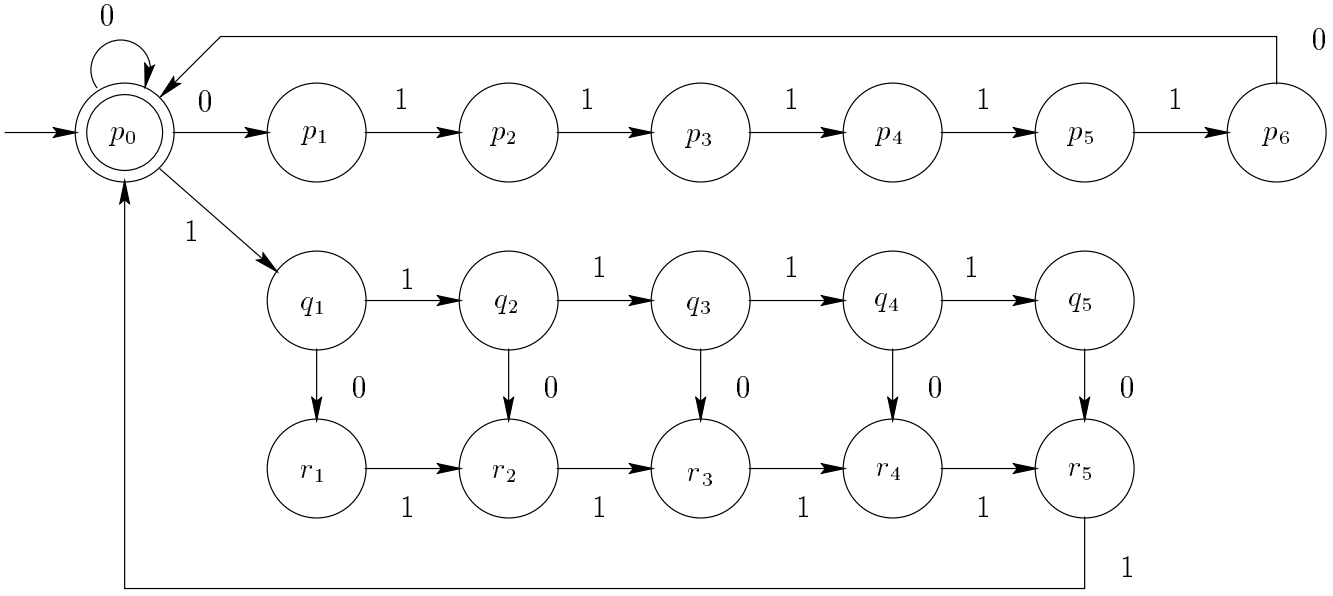

Figure 2: The NFA M_6

Finally, we argue that each of these subsets of states is inequivalent. This is because given two distinct such subsets, say T' and T'', there must be an r_i, $1 \leq i \leq t-2$, that is contained in one (say T') but not the other. Then reading the string 1^{t-i} takes T' to p_0, but not T''.

Since the alphabetic length of the strings in S_t is $n = t^2+t+1$, it now follows that this example has state complexity $2^{\Omega(\sqrt{n})}$.

Open Problems on Regular Expressions

1. Prove an exponential lower bound for DFA $\rightarrow$ RE conversion.

- Ehrenfeucht & Zeiger (1976) did this for unbounded alphabet size
- Waizenegger (2000) proved a lower bound of $2^{\sqrt[3]{n}}$ for alphabet size 4
- Unfortunately, there are few good lower bound techniques for regular expression size.

Relation to Boolean Formula Size

Take a regular expression for a uniform language over $\{0, 1\}$.

- Replace each 1 in position i by x_i and each 0 by $\overline{x_i}$.
- Replace each $\cdot$ by $\wedge$ and each $+$ by $\vee$
- The resulting boolean expression is satisfiable by $(a_1, a_2, \ldots, a_n)$ iff $a_1 a_2 \cdots a_n \in L$.
- By a result of Khrapchenko on boolean formula size, this shows any regular expression for $\{x \in \{0,1\}^n \ : \ |x|_1 \text{ is even}\}$ has $\Omega(n^2)$ size.

Open Problems on Regular Expressions

2. Determine the worst-case blow-up in going from a regular expression for L to a regular expression for $\overline{L}$.

- Best upper bound known is c^{2^n}
- Worst example known is

$$E_n = (0+1)^*0(0+1)^{n-1}0(0+1)^*$$

of length $2n+4$, where the best expression for $\overline{E_n}$ has length $\geq 2^n$.

Open Problems on Regular Expressions

3. Determine worst-case blow-up in going from regular expressions for L_1, L_2 to regular expression for $L_1 \cap L_2$.

- Best upper bound known is c^{mn}
- No example known does worse than mn except
  $L_1 = (0+1)^n$ of length $2n$
  $L_2 = (0+10^*1)^*$ of length 4.

Open Problems on Regular Expressions

4. What are the optimal regular expressions for the binomial languages

$$B(n,k) = \{x \in \{0,1\}^n \ : \ |x|_1 = k\}$$

It is known that length $O(n(\log n)^k)$ suffices.

For Further Reading

1. M. Domaratzki, D. Kisman, and J. Shallit, On the number of distinct languages accepted by finite automata with n states, *J. Automata, Languages, and Combinatorics* **7** (2002), 469–486.

2. K. Ellul, B. Krawetz, J. Shallit, and M.-w. Wang, Regular expressions: new results and open problems, *J. Automata, Languages, and Combinatorics*, to appear.