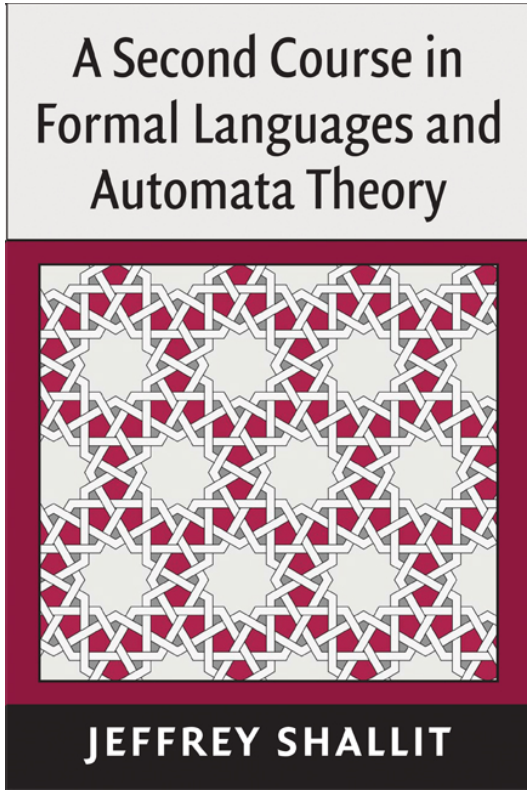
Enumeration of Automata, Languages, and Regular Expressions

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

Joint work with Hermann Gruber and Jonathan Lee.

An Advertisement



Published by Cambridge University Press! Order your copy today!

Some Hard Counting Problems

- How many different n-state automata are there?
- It seems like a well-specified question, but
  - deterministic or nondeterministic?
  - taking into account final states and initial states? or topology only?
  - up to isomorphism?
  - unary/binary/larger input alphabet?
  - with output on alphabet of size ≥ 3? or accept/reject only?
  - only initially-connected/strongly-connected automata?
  - minimal?
  - accepting distinct languages?
  - acyclic (generating finite languages)?

Topology vs. computer science

To illustrate this issue, consider unary automata (so that each edge is labeled by a single letter a).

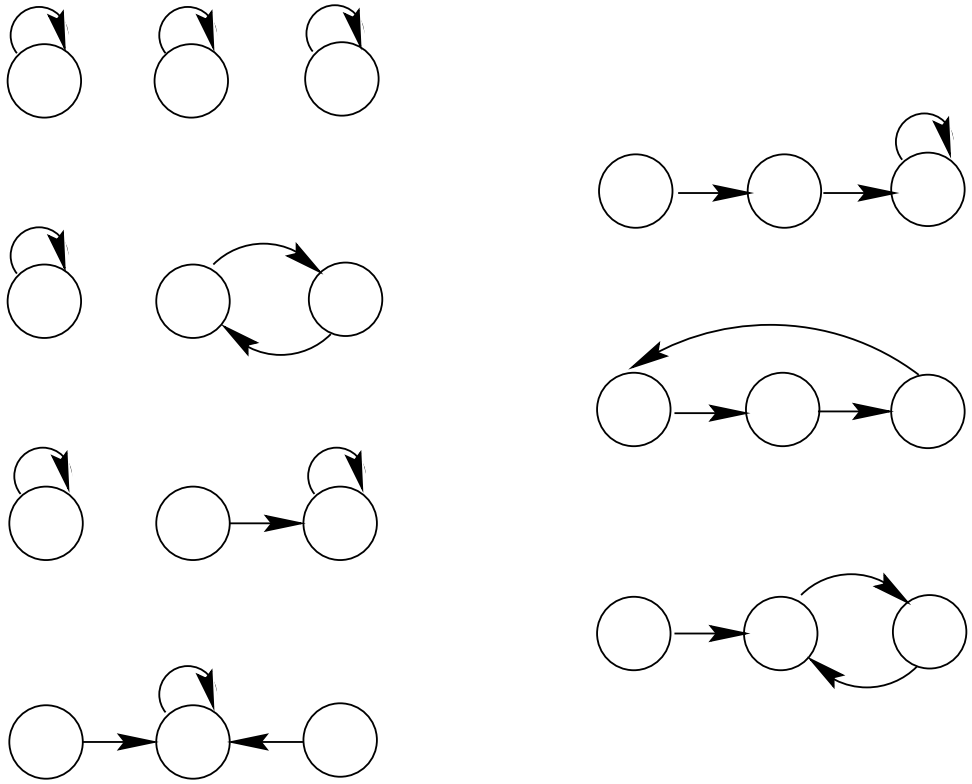
Disregarding initial and final states, the number of such automata on n labeled states is n^n, as each state can be sent to any of the n others.

But many of these are isomorphic, in the sense that a permutation of state names preserves the structure.

These structures, up to isomorphism, are known as *functional digraphs*.

Counting functional digraphs

There are 7 functional digraphs on 3 vertices:



They were enumerated by Davies, Harary, and Read in independent papers from 1953–1961.

Topology vs. computer science

De Bruijn found the following exact formula (1972):

$$S_n = \sum_{n_1+2n_2+3n_3+\cdots=n} \left(\prod_{i\geq 1} \left(\sum_{j \mid i} j n_j \right)^{n_i} \frac{i^{-n_i}}{n_i!} \right)$$

Here the sum is over all partitions of n.
Livshits (1964) found this number is asymptotically $An^{-\frac{1}{2}}\tau^n$ for $A \doteq .45$, $\tau \doteq 2.94$.

Topology vs. computer science

However, considered as unary deterministic automata (with initial state and set of final states), most of these functional digraphs are uninteresting, since they are not initially connected.

There are exactly n initially-connected functional digraphs on n vertices.

Considered as automata, we want to enumerate the number of such n-state machines that are minimal.

Enumerating unary DFA's

Theorem (Nicaud)

An n-state unary DFA $M = (Q, \{a\}, \delta, q_0, F)$, where $Q = \{q_0, q_1, \ldots, q_{n-1}\}$, is minimal iff the following three conditions hold:

(a) *It is initially connected. Its transition diagram consists of a "loop" and a "tail".*

(b) *The loop is "minimal", that is, it cannot be replaced by an equivalent smaller loop.*

(c) *The last state of the loop and the last state of the tail are of opposite "finality", i.e., one is in F and the other isn't.*

Enumerating unary DFA's

The loop defined by states $q_j, \ldots, q_{n-1}$ is minimal if and only if the word $a_j a_{j+1} \cdots a_{n-1}$ defined by

$$a_i = \begin{cases} 1, & \text{if } q_i \in F; \\ 0, & \text{if } q_i \notin F; \end{cases}$$

is primitive.

(A nonempty word w is *primitive* if it cannot be written in the form $w = x^k$ for some word x and integer $k \geq 2$.)

Enumerating unary DFA's

Let $\psi_k(n)$ denote the number of primitive words of length n over a k-letter alphabet.

By Möbius inversion, we get

$$\psi_k(n) = \sum_{d \mid n} \mu(d) k^{n/d},$$

where μ is the Möbius function, defined as follows:

$$\mu(n) = \begin{cases} 0, & \text{if } n \text{ is divisible by a square} > 1; \\ (-1)^s, & \text{if } n = p_1 p_2 \cdots p_s, \text{ where the } p_i \text{ are distinct primes.} \end{cases}$$

Enumerating unary DFA's

Theorem (Domaratzki, Kisman, S)

If $f_1(n)$ is the number of minimal unary DFA's with n states, then

$$f_1(n) = \psi_2(n) + \sum_{1 \leq j \leq n-1} \psi_2(n-j)2^{j-1}.$$

Proof.

The result follows from Nicaud's theorem. The 2^{j-1} factor comes from the fact that there are j states in the tail and if $j \geq 1$, then the type of one of the states (final or non-final) is fixed by condition (c). □

Enumerating unary DFA's

If $g_1(n)$ is the number of distinct languages accepted by unary DFA's with n states, we get

$$g_1(n) = \sum_{1 \leq t \leq n} f_1(t).$$

Theorem

We have $g_1(n) = \sum_{1 \leq t \leq n} \psi_2(t) 2^{n-t}$.

Enumerating unary DFA's

Theorem (Domaratzki, Kisman, S)

We have

$$g_1(n) = 2^n(n - \alpha + O(n2^{-n/2})),$$

where

$$\alpha = \sum_{d \geq 2} \frac{\mu(d)}{1 - 2^{d-1}} \doteq 1.382714455402396285 47.$$

Corollary

We have

$$f_1(n) = 2^{n-1}(n + 1 - \alpha + O(n2^{-n/2})).$$

Enumerating unary DFA's

n	$f_1(n)$	$g_1(n)$
1	2	2
2	4	6
3	12	18
4	30	48
5	78	126
6	180	306
7	432	738
8	978	1716

Enumerating DFA's with larger alphabet size

We can use the previous result about unary automata to get a pretty good estimate on $f_k(n)$, the number of n-state minimal DFA's over a k-letter alphabet.

To do so, let the restriction of such a DFA be minimal over a chosen letter, and choose the transition function on the remaining letters to be any of the $n^{(k-1)n}$ possible functions.

We get

Theorem (Domaratzki, Kisman, S)

We have $f_k(n) \geq (k - o(1))n2^{n-1}n^{(k-1)n}$.

Enumeration: what kind of answer do you want?

Possible answers:

- An exact enumeration: e.g., $f_1(8) = 978$ (there are exactly 978 distinct minimal DFAs with 8 states over a unary alphabet)
- A formula giving the exact number for all n, e.g., $f_1(n) = \psi_2(n) + \sum_{1 \leq j \leq n-1} \psi_2(n-j)2^{j-1}$
- An asymptotic expression for the number of automata, e.g., $f_1(n) = 2^{n-1}(n + 1 - \alpha + O(n2^{-n/2}))$.

Not all exact enumerations are created equal

Two formulas for the number $C_k(n)$ of n-state initially-connected DFA's (without considering final states) over an alphabet of size k:

$$C_k(n) = n^{kn} - \sum_{1 \le j < n} \binom{n-1}{k-1} C_k(j) n^{k(n-j)}.$$

$$C_k(n) = \sum_{1 \le b_1 \le k} \sum_{1 \le b_2 \le 2k - b_1} \cdots \sum_{1 \le b_{n-1} \le k(n-1) - (b_1 + \cdots + b_{n-2})} \prod_{1 \le j \le n} j^{b_j - 1}.$$

Why do we care?

- The joy of counting
  - e.g., Gauss: "Very often, I passed an idle quarter of an hour to count through another chiliad here and there")
- Test conjectures by exhaustive search
  - e.g., state complexity
- Good enumeration can suggest methods for random generation
  - Can also test conjectures by random generation (e.g., what is the distribution of state complexities under the transformation $L \to L^R$?)
- Counting arguments give lower bounds

Counting arguments give lower bounds on complexity

- Consider finding regular expressions for finite languages: more specifically, for a set S of n binary words of length n.
- Clearly we can find a regular expression of length $O(n^2)$ for any such set S.
- But are there such sets where $\Omega(n^2)$ regular expression size is necessary?
- A counting argument solves this.
- There are at most c^r distinct regular expressions of length $< r$.
- But there are $\binom{2^n}{n}$ different subsets of size n of $\{0,1\}^n$.
- Now $\binom{2^n}{n} \sim \frac{2^{n^2}}{n!}$.
- So if $\frac{2^{n^2}}{n!} > c^r$, then there is some subset requiring a length-r regular expression.
- Taking logs, we get that some subset requires a regular expression of length $\Omega(n^2)$.

Lower bound for nondeterministic automaticity

- nondeterministic automaticity of a language L: function $N_L : n \longrightarrow$ smallest number of states in any NFA M such that $L(M) \cap \Sigma^{\leq n} = L \cap \Sigma^{\leq n}$
- Want lower bound for "almost all" unary languages (all but a set of measure 0)
- Our model is that every string $\epsilon, 0, 0^2, \ldots$ is either in L or not in L with probability $1/2$.

Lower bound for nondeterminstic automaticity

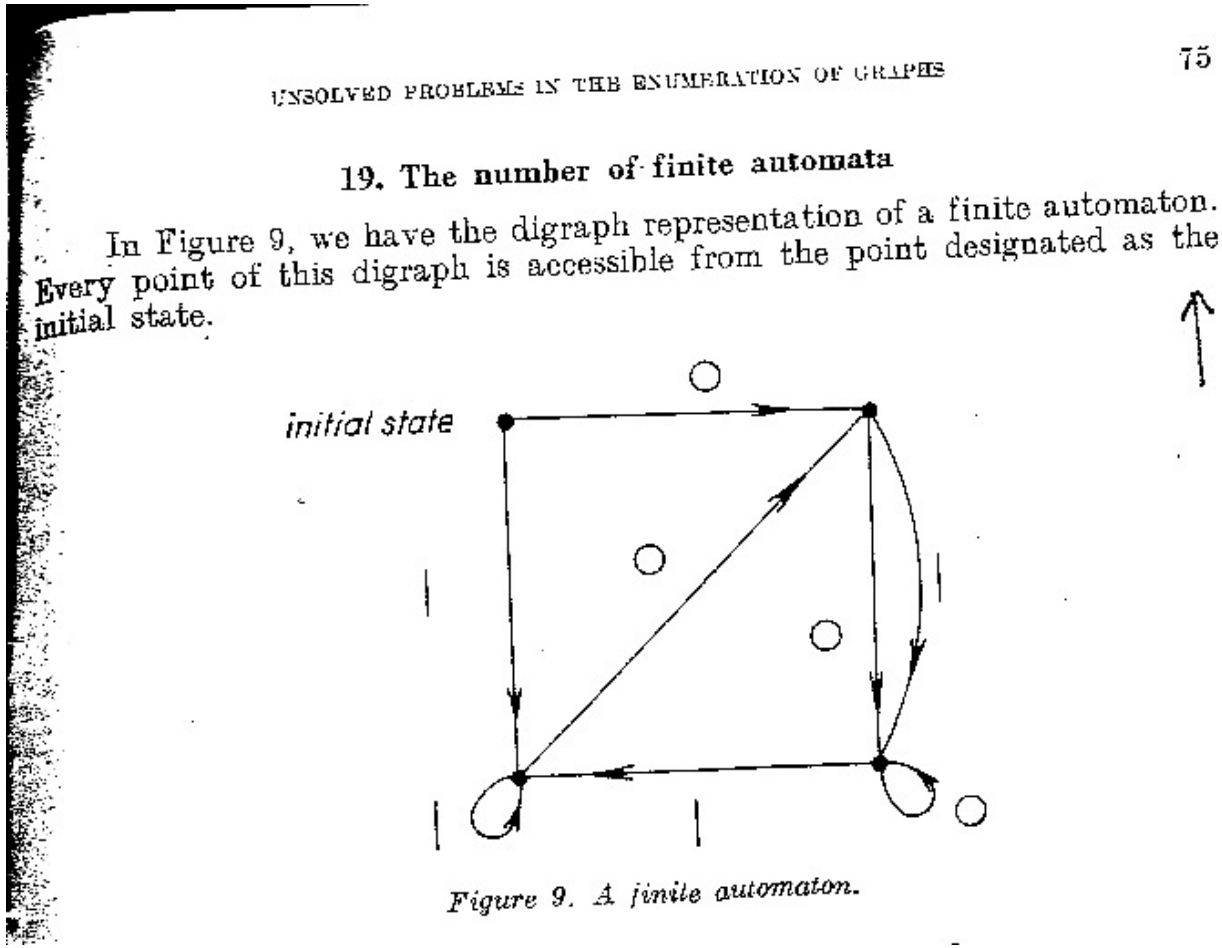
- The number of languages that are subsets of $\{0\}^{\leq n}$ is 2^{n+1}
- the number of languages that are subsets of $\{0\}^{\leq n}$ that are specified by NFA's with $\leq q$ states is $\leq G_1(q)$, the number of distinct unary NFA languages using at most q states.
- We have the upper bound $G_1(q) = O(q/\log q)^q$ (Pomerance, Robson, S)
- the probability that a randomly chosen language over $\{0\}^{\leq n}$ has nondeterministic automaticity $N_L(n) \leq cn/\log n$ is at most $G_1(cn/\log n)/2^{n+1} \leq 2^{-c'n}$.
- Since $\sum_{n\geq 0} 2^{-c'n}$ converges, by the Borel-Cantelli lemma, with probability 1 we have $N_L(n) > cn/\log n$ for all sufficiently large n.

Other lower bounds

Gramlich & Schnitger used bounds on the number of languages accepted by NFA's with n states to prove inapproximability results on finding minimal NFA's for a given DFA.

History

Enumeration of DFA's was stated as problem 19 in Harary's 1960 paper, Unsolved problems in the enumeration of graphs.



UNSOLVED PROBLEMS IN THE ENUMERATION OF GRAPHS 75

19. The number of finite automata

In Figure 9, we have the digraph representation of a finite automaton. Every point of this digraph is accessible from the point designated as the initial state.

Figure 9. A finite automaton.

History

Harrison (1965) solved the enumeration problem for arbitrary input and output alphabet size (but no initial state or set of final states), in the Mealy model (outputs associated with transitions).

Harary and Palmer (1967) obtained similar results.

---

35 #6492 94.40 (05.00)
Harary, Frank; Palmer, Ed
Enumeration of finite automata.
Information and Control **10** 1967 499--508.

Except for a few minor variants, the results of this paper may be found in an article by the reviewer [Canad. J. Math. 17 (1965), 100--113; MR **30** #1010].

Reviewed by *M. A. Harrison*

Cited in: 48 #8139 47 #46

History

Korshunov (1967) obtained asymptotic estimates in the Mealy model.

Liskovets (1969) determined the number of initially-connected automata in the Mealy and Moore models. This was later done by Robinson (1985), independently.

Korshunov (1966–1978) enumerated many different flavors of automata, giving both exact expressions and asymptotic results, but usually without a distinguished initial state.

History

More recently, Bassino and Nicaud found that the number of non-isomorphic initially-connected DFA's with n states over a k-letter alphabet is $\Theta(n2^n \left\{ {kn \atop n} \right\})$, where $\left\{ {kn \atop n} \right\}$ is a Stirling number of the second kind.

Domaratzki (2004) enumerated the number of automata accepting finite languages.

Enumerating regular expressions

- The number of regular expressions of length n is clearly bounded above and below by an expression of the form c^n.
- So our goal is to try to establish as tight bounds as possible on c.
- Not considered until recently

Enumerating regular expressions

- No agreement on what is a valid regular expression
  - e.g., is the empty expression valid?
  - is () valid?
  - is a** valid?
- Many definitions of regular expressions in the literature are wrong or not rigorous
- the most common problem is lack of rigorous definition when parentheses are needed

Some incorrect definitions

Hopcroft and Ullman's definition doesn't permit ab as a valid regular expression, although they do say "In writing regular expressions we can omit many parentheses..."

One textbook: "If r and s are regular expressions over Σ denoting languages L_1 and L_2, respectively, then rs ... [is a] .. regular expression, where rs denotes the language $L_1 L_2$..."

Take $r = 1 + 2$, $s = 3$. Then $rs = 1 + 23 \neq (1 + 2)3$.

How can we define regular expressions more rigorously?

An unambiguous grammar for valid regular expressions

$$
\begin{array}{rrr}
S \to & E_+ \mid E_\bullet \mid G & \text{all regular expressions} \\
E_+ \to & E_+ + F \mid F + F & \text{unparenthesized, last op } + \\
F \to & E_\bullet \mid G & \\
E_\bullet \to & E_\bullet\, G \mid GG & \text{unparenthesized, last op } \bullet \\
G \to & E_* \mid C \mid P & \\
C \to & \emptyset \mid \epsilon \mid a \quad (a \in \Sigma) & \text{unparenthesized, no last op} \\
E_* \to & G* & \text{last op } * \\
P \to & (S) & \text{parenthesized exps}
\end{array}
$$

"parenthesized" – there is at least one pair of enclosing parentheses.

This grammar allows $a**$ but disallows $()$.

Enumerating context-free languages with Chomsky-Schützenberger

Once we agree on what constitutes a regular expression, and we find an unambiguous grammar generating them, we can enumerate them using the Chomsky-Schützenberger theorem:

Theorem (Chomsky-Schützenberger)

If L is a context-free language having an unambiguous grammar and $a_n = |L \cap \Sigma^n|$, then $A(x) := \sum_{n \geq 0} a_n x^n$ is a formal power series in $\mathbb{Z}[[x]]$ that is algebraic over $\mathbb{Q}(x)$.

(For proof, see the book of Kuich and Salomaa, or paper of Panholzer.)

Enumerating regular expressions with Chomsky-Schützenberger

Furthermore, $A(x)$ is a zero of the system of equations given by the commutative image of the grammar, where

- Every terminal is replaced by a variable x
- Every occurrence of ϵ is replaced by the integer 1
- Every occurrence of $\longrightarrow$ is replaced by $=$
- Every occurrence of $|$ is replaced by $+$

Enumerating regular expressions with Chomsky-Schützenberger

Let's look at a simple example.

Consider the unambiguous grammar

$$\begin{aligned} S &\to M \mid U \\ M &\to 0M1M \mid \epsilon \\ U &\to 0S \mid 0M1U \end{aligned}$$

which represents strings of "if-then-else" clauses.

Then this grammar has the following commutative image:

$$\begin{aligned} S &= M + U \\ M &= x^2M^2 + 1 \\ U &= Sx + x^2MU \end{aligned}$$

From the system to an equation for S

By the Chomsky-Schützenberger theorem, each variable satisfies an algebraic equation over $\mathbb{Q}(x)$.

$$\begin{aligned} S &= M + U && (1)\\ M &= x^2M^2 + 1 && (2)\\ U &= Sx + x^2MU && (3) \end{aligned}$$

We can solve the system above to find the equation for S, as follows:

First, we solve (3) to get $U = \frac{Sx}{1-x^2M}$, and substitute back in (1) to get $S = M + \frac{Sx}{1-x^2M}$.

From the system to an equation for S

Multiplying $S = M + \frac{Sx}{1-x^2M}$ by $1 - x^2M$ gives
$S - x^2MS = M - x^2M^2 + Sx$, which, by (2), is equivalent to
$S - x^2MS = 1 + Sx$.

Solving for S, we get $S = \frac{1}{1-x^2M-x}$.

Now

$$(1-x^2M-x)^2 = x^2(1-M+x^2M^2)-x(2x-1)-(2x-1)(1-x^2M-x),$$

so we get $S^{-2} = -x(2x-1) - (2x-1)S^{-1}$ and hence

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

This is an equation for S.

From the equation to the Taylor series

Now that we have the equation for S

$$x(2x-1)S^2 + (2x-1)S + 1 = 0,$$

we can solve for S to obtain

$$S = \frac{1 - 2x + \sqrt{1-4x^2}}{2x(2x-1)}.$$

This gives us the Taylor series expansion

$$S = 1 + x + 2x^2 + 3x^3 + 6x^4 + 10x^5 + 20x^6 + \cdots .$$

From the equation to a closed form and asymptotics

We can now get the closed form

$$S = \sum_{n \geq 0} a_n x^n$$

where

$$a_n = \binom{n}{\lfloor \frac{n}{2} \rfloor}.$$

From this we get the asymptotic expression

$$a_n = \sqrt{\frac{2}{\pi}} n^{-\frac{1}{2}} 2^n.$$

Finding the equation, in general

We can use the theory of *Gröbner bases* to solve the system of algebraic equations obtained from the grammar.

This theory guarantees that a defining equation for S can be found, but the solution may take exponential time.

Singularity analysis

Once we have found the equation $f(S) = 0$, we can get the asymptotics of the n'th term of its power series expansion using *singularity analysis*.

Roughly speaking, this amounts to finding the smallest real root r of the discriminant of f.

Then $a_n = O((1/r + \epsilon)^n)$ for any $\epsilon > 0$, and in some cases, we can get a more detailed understanding of the asymptotics.

Using Gröbner bases

All this can be automated, e.g., with Maple:

```
eqs := [ -S + M + U, -M + x^2*M^2 + 1, -U + S*x + x^2*M*U ];
```

$$eqs := [-S + M + U, -M + x^2 M^2 + 1, -U + Sx + x^2 MU] .$$

Maple provides an elimination ordering called `lexdeg` to compute a reduced Gröbner basis using this ordering:

```
Groebner[Basis](eqs, lexdeg([M, U], [S]));
```

$$[1 + (-1 + 2x) S + \left(-x + 2x^2\right) S^2, 1 + (-1 + x) S + Ux,$$

$$-1 + (1 - 2x) S + Mx] .$$

Using Gröbner bases

The algebraic equation satisfied by S is the first polynomial in this set:

```
algeq := %[1];
```

$$algeq := 1 + (-1 + 2\,x)\,S + \left(-x + 2\,x^2\right)\,S^2\,.$$

Using Gröbner bases

To compute the Laurent series zeros of S using this polynomial, we solve for S and expand the solutions as Laurent series in the indeterminate x:

```
map(series, [solve(algeq, S)], x);
```

$$[(-x^{-1} - 1 - x - 2\,x^2 - 3\,x^3 - 6\,x^4 - 10\,x^5 + O\left(x^6\right)),$$
$$(1 + x + 2\,x^2 + 3\,x^3 + 6\,x^4 + 10\,x^5 + O\left(x^6\right))]\,.$$

Our desired power series solution is the second entry in the above returned list.

Singularity analysis using Maple

```
> algeq;
```

$$1 + (-1 + 2x)\,S + \left(-x + 2x^2\right) S^2 .$$

We compute the discriminant D:

```
> d := discrim(algeq,S);
```

$$d := -(2x + 1)(-1 + 2x) .$$

The real roots of D are given by:

```
> realroots := [fsolve(%)];
```

$$realroots := [-0.5000000000, 0.5000000000] .$$

Finally, an upper bound is given by taking the inverse of the smallest positive real root:

```
> 1/min(op(select(type, realroots, positive)));
```

$$2.000000000 .$$

Length of a regular expression

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

Relationship between different ways to measure length

These measures are essentially identical, up to a constant multiplicative factor.

Theorem (Ellul, Krawetz, S, Wang; Ilie, Yu)

If an expression

- *contains no unnecessary occurrences of* $\emptyset$
- *contains no subexpression of the form* FG *or* GF *where* $L(F) = \epsilon$
- *contains no subexpression of the form* $F + G$ *or* $G + F$ *where* $L(F) = \epsilon$ *and* $L(G)$ *contains* ϵ
- *contains no superfluous parentheses*
- *contains no subexpression of the form* F^{**}

then the length of an expression in each measure is bounded by a constant times the length in every other measure.

Enumeration of regular expressions

We can take our unambiguous grammar for regular expressions

$$
\begin{array}{rrl}
S \to & E_+ \mid E_\bullet \mid G & \text{all regular expressions} \\
E_+ \to & E_+ + F \mid F + F & \text{unparenthesized, last op } + \\
F \to & E_\bullet \mid G & \\
E_\bullet \to & E_\bullet\, G \mid GG & \text{unparenthesized, last op } \bullet \\
G \to & E_* \mid C \mid P & \\
C \to & \emptyset \mid \epsilon \mid a \quad (a \in \Sigma) & \text{unparenthesized, no last op} \\
E_* \to & G* & \text{last op } * \\
P \to & (S) & \text{parenthesized exps}
\end{array}
$$

and compute its commutative image...

Commutative image of our grammar

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

Now we can use Gröbner bases to find the algebraic equation satisfied by S.

Finding the equation for S

It is

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

Solving for S, we get

$$S = \frac{-(k+3)x^2 - (k+3)x + 1 - \sqrt{D}}{2(x^2 + x^3)}.$$

where the discriminant

$$D = (k+1)^2x^4 + 2(k^2+4k+5)x^3 + (k+1)(k+3)x^2 - 2(k+3)x + 1.$$

Enumeration of regular expressions

We can expand S as a power series to get a generating function enumerating the regular expressions of length n.

For example, for $k = 2$, we have

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

Asymptotic analysis

The asymptotic growth rate of the coefficients of the generating function for S depends on the reciprocal of the smallest zero of the discriminant D.

For S, this smallest zero can be represented by the asymptotic series

$$\frac{1}{k} - \frac{6}{k^2} + \frac{42}{k^3} - \frac{1319}{4k^4} + \frac{22463}{8k^5} - \cdots,$$

and its reciprocal is

$$k + 6 - \frac{6}{k} - \frac{167}{4k^2} - \frac{2615}{8k^3} - \cdots.$$

For $k = 1$ the smallest zero is about .16246250262 and for $k = 2$ it is about .13755127577.

Asymptotic analysis

We can prove

Theorem

We have $S_k(n) \sim c_k \alpha_k^n n^{-3/2}$ *for some constant* c_k*, where* $\alpha_1 \doteq 6.1552665$ *and* $\alpha_2 \doteq 7.2700161767$*.*

At least asymptotically, this solves the problem of enumerating regular expressions over an alphabet of size k.

Counting languages represented by regular expressions

Next we consider a harder problem: counting the distinct languages represented by regular expressions of length n.

We want both lower bounds and upper bounds.

The idea in both cases is the same: find an appropriate unambiguous grammar; use the Chomsky-Schützenberger theorem; find an equation for the generating series; and then do asymptotic analysis.

For the lower bound, we try to generate as many distinct languages as we can without duplication.

For the upper bound, we create a normal form that specifies languages concisely.

Lower bound example

Consider regular expressions of the form

$$w_1(\epsilon + w_2(\epsilon + w_3(\epsilon + \cdots)))$$

where the w_i denote nonempty words.

Each distinct choice of the w_i specifies a distinct language.

These expressions can be generated by the grammar

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

which has the commutative image

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

Lower bound example

The solution to this system is

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

The asymptotic behavior of the coefficients of the power series for S depend on the zeros of $1 - kx - kx^5$.

The smallest (indeed, the only) real root is, asymptotically as $k \to \infty$, given by

$$\sum_{i \geq 0} \frac{(-1)^i \binom{5i}{i}}{4i+1} k^{-(4i+1)} = \frac{1}{k} - \frac{1}{k^5} + \frac{5}{k^9} - \frac{35}{k^{13}} + \cdots.$$

Lower bound example

The reciprocal of this series is

$$\sum_{i\geq 0} \frac{4\binom{5i+5}{i+1}}{5(5i+4)} k^{1-4i} = k + \frac{1}{k^3} - \frac{4}{k^7} + \frac{26}{k^{11}} - \frac{204}{k^{15}} + \frac{1771}{k^{19}} - \cdots .$$

For $k = 1$ the only real root of $1 - kx - kx^5$ is approximately .754877666 and for $k = 2$ it is about .4756527435. Thus we have

Theorem

$R_1(n) = \Omega(1.3247^n)$ and $R_2(n) = \Omega(2.102374^n)$.

Improving the lower bound

Outline:

First, we develop a trie representation for finite languages.

Next, we incorporate Kleene * selectively to take into account some infinite languages

With this we can improve our lower bounds to, for example, $R_2 = \Omega(2.7799^n)$.

Improving the upper bound

Idea: find "normal form" for regular expressions that does not increase length.

$S \to$	$Q \mid A \mid T \mid C \mid K$
$Q \to$	$A + \epsilon \mid T + \epsilon \mid C + \epsilon$
$A \to$	$T + A_T \mid C + A_C \mid K + A_K$
$A_T \to$	$T \mid T + A_T \mid A_C$
$A_C \to$	$C \mid C + A_C \mid A_K$
$A_K \to$	$K \mid K + A_K$
$T \to$	$a_1 \mid a_2 \mid \cdots \mid a_k$
$C \to$	$C_0\, C_0 \mid C_0\, C$
$C_0 \to$	$(Q) \mid (A) \mid T \mid K$
$K \to$	$(A)* \mid T* \mid (C)*$

With this, we can get upper bounds of the form $R_1 = O(2.5946^n)$ and $R_2 = O(4.2877^n)$.

Summary of results: languages generated by regular expressions

n	Lower bound	Upper bound
1	$\Omega(1.3247^n)$	$O(2.5946^n)$
2	$\Omega(2.7799^n)$	$O(4.2877^n)$
3	$\Omega(3.9582^n)$	$O(5.4659^n)$
4	$\Omega(5.0629^n)$	$O(6.5918^n)$
5	$\Omega(6.1319^n)$	$O(7.6870^n)$

Enumerating context-free languages

Theorem (Domaratzki, Okhotin, S)

Let $f(n)$ denote the number of distinct CFL's of size n with respect to a given, computable, well-behaved descriptional complexity measure. Then the function $f(n)$ is uncomputable.

Idea: use $f(n)$ to determine equivalence classes of grammars by testing on words of increasing length. This allows us to determine if $L(G) = L(G')$, an unsolvable problem.

Enumerating context-free languages

Let $\lambda_k(n)$ denote the number of distinct CFL's generated by CFG's with n symbols (total of lengths of productions) over a k-letter alphabet.

Theorem (Domaratzki, Okhotin, S)

We have $\lg(\lambda_k(n)) \leq (n-1)\lg(n/2+k+1)$ *and*

$$\lg(\lambda_k(n)) \geq \frac{k-1}{4k+2} n \lg n + O(n \lg k).$$

Open Problems

1. Improve the known upper and lower bounds on $G_1(n)$, the number of distinct languages accepted by unary n-state NFA's. The best upper bound known is $(cn/\log n)^n$ and the best lower bound known is $2^{n+c'\sqrt{\frac{n}{\log n}}}$.
2. Improve the upper and lower bounds on $R_k(n)$, the number of distinct languages specified by regular expressions of length n

Open Problems

3. Prove something about the distribution of the sizes of minimal DFA's for L^R, where L is accepted by an n-state DFA
   - implications for average-case of Brzozowski's minimization algorithm
4. Prove something about the distribution of the sizes of the minimal DFA's for L, where L is accepted by an n-state NFA
5. Prove something about the distribution of the sizes of the DFA's generated by the subset construction on n-state NFA's

For Further Reading

- M. Domaratzki, D. Kisman, and J. Shallit, On the number of distinct languages accepted by finite automata with n states, *J. Autom. Lang. Combin.* **7** (2002), 469–486.
- M. Domaratzki, Enumeration of formal languages, *Bull. EATCS* No. 89 (June 2006), 113–133.
- J. Lee and J. Shallit, Enumerating regular expressions and their languages, Proc. CIAA 2005, pp. 2–22.