Regular Expressions: New Results and Open Problems

Keith Ellul, Jeffrey Shallit, and Ming-wei Wang
School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada
shallit@graceland.uwaterloo.ca
http://www.math.uwaterloo.ca/~shallit

Regular Languages

Regular languages can be represented in many ways.

- as the language accepted by
  - a deterministic finite automaton (DFA);
  - a nondeterministic finite automaton (NFA); or
  - a nondeterministic finite automaton with ϵ-transitions (NFA-ϵ);
- as the language specified by
  - a regular expression (RE), allowing the operations of union ($+$), concatenation (typically represented implicitly by juxtaposition), and Kleene closure ($*$)
  - a generalized regular expression (GRE), allowing the additional operations of intersection ($\cap$) and complement ($\neg$)

Regular Expressions

- Introduced by Kleene [1956]
- Often used as specification in software (e.g., `grep`, `lex`)
- Many problems about their descriptive capabilities remain open

Size of Regular Expressions

No agreement in the literature about the proper measure for the size of a regular expression.

- One obvious measure is the *ordinary length*: the total number of symbols, including parentheses
  - The regular expression $(0+10)^*(1+\epsilon)$ has ordinary length 12.
  - Note there is no explicit symbol for concatenation in such a scheme.
- Another measure is based on the length of the expression converted to (parenthesis-free) reverse polish form, using an explicit operator $\bullet$ for concatenation.
  - Thus the previous regular expression would be written $010\bullet+*1\epsilon+\bullet$ and has *reverse polish length* 10.

Size of Regular Expressions

- The most useful measure in practice seems to be the total number of *alphabetic symbols*, counted with multiplicity.
  - By an alphabetic symbol we mean an element of Σ, ignoring all operations, parentheses, and the special symbols ϵ and $\emptyset$.
  - Under this measure, the expression

    $$(0 + 10)^*(1 + \epsilon)$$

    has length 4.

The three measures

- ordinary length
- reverse polish length
- number of alphabetic symbols

are all essentially identical, up to a constant multiplicative factor.

Size of Regular Expressions

We say "essentially identical" because one can always artificially inflate the ordinary length of a regular expression by adding arbitrarily many multiplicative factors of ϵ, additive factors of $\emptyset$, etc.

For example,

$$\epsilon\epsilon\epsilon\cdots\epsilon$$

has ordinary length n, but 0 alphabetic symbols.

In order to avoid such trivialities, we define what it means for a regular expression to be collapsible, as follows:

Collapsible Regular Expressions

Definition.

Let E be a regular expression over the alphabet Σ, and let $L(E)$ be the language specified by E. We say E is *collapsible* if any of the following conditions hold:

1. E contains the symbol $\emptyset$ and $|E| > 1$;
2. E contains a subexpression of the form FG or GF where $L(F) = \{\epsilon\}$;
3. E contains a subexpression of the form $F+G$ or $G+F$ where $L(F) = \{\epsilon\}$ and $\epsilon \in L(G)$.

Otherwise, if none of the conditions hold, E is said to be *uncollapsible*.

Irreducible Regular Expressions

Definition.

If E is an uncollapsible regular expression such that

1. E contains no superfluous parentheses; and
2. E contains no subexpression of the form F^{**}.

then we say E is *irreducible*.

Note that a minimal regular expression for E is uncollapsible and irreducible, but the converse does not necessarily hold.

Length Measures for Regular Expressions

Theorem. Let E be a regular expression over Σ. Let

- $|E|$ denote its ordinary length;
- $|\mathrm{rpn}(E)|$ denote its reverse polish length; and let
- $|\mathrm{alph}(E)|$ denote the number of alphabetic symbols contained in E.

Then we have

(a) $|\mathrm{alph}(E)| \leq |E|$;

(b) If E is irreducible and $|\mathrm{alph}(E)| \geq 1$, then $|E| \leq 11|\mathrm{alph}(E)| - 4$;

(c) $|\mathrm{rpn}(E)| \leq 2|E| - 1$;

(d) $|E| \leq 2|\mathrm{rpn}(E)| - 1$;

(e) $|\mathrm{alph}(E)| \leq \frac{1}{2}(|\mathrm{rpn}(E)| + 1)$;

(f) If E is irreducible and $|\mathrm{alph}(E)| \geq 1$, then $|\mathrm{rpn}(E)| \leq 7|\mathrm{alph}(E)| - 2$.

Irreducible Regular Expressions

The key idea is a lemma due to Ilie and Yu [2002].

Lemma. Let E be an uncollapsible regular expression over Σ containing at least one alphabetic symbol. Then the number of occurrences of ϵ in E is $\leq$ the number of alphabetic symbols in E. If equality occurs, then $\epsilon \in L(E)$.

Proof. By induction on the height of the expression tree induced by E.

Lower Bounds on Regular Expression Size

Very few techniques are known for bounding the size of a regular expression for a given language.

One technique uses the following easy observations:

Proposition. Let L be a nonempty regular language.

(a) If the length of the shortest string in L is n, then $|\mathrm{alph}(E)| \geq n$ for any regular expression E with $L(E) = L$.

(b) If further L is finite, and the length of the longest string in L is N, then $|\mathrm{alph}(E)| \geq n$ for any regular expression E with $L(E) = L$.

Lower Bounds on Regular Expression Size

Another technique uses a theorem of Leiss, which states that given an RE E with $|\mathrm{alph}(E)| = n$, there exists an NFA M with at most $n+1$ states such that $L(M) = L(E)$.

Hence a lower bound on the number of states in an NFA accepting L implies a similar lower bound on the length of an equivalent regular expression.

To obtain lower bounds on the number of states in an NFA we may use the following result of Birget

Theorem. Let L be a regular language. If there exist t pairs of strings $\{(x_1, y_1), \ldots, (x_t, y_t)\}$ such that

(i) $x_i y_i \in L$ for $1 \leq i \leq t$;

(ii) $x_i y_j \notin L$ or $x_j y_i \notin L$ for all i, j with $1 \leq i < j \leq t$;

then any NFA accepting L must have at least t states.

Lower Bounds on Regular Expression Size

There are also ad hoc methods. For example, by considering the nodes in the expression tree for the corresponding regular expression we can prove

Theorem.

Any regular expression for the set of permutations over $\{1, 2, \ldots, n\}$ is of length at least c^n, where $c \doteq 1.89$.

Finally, there is the method of Ehrenfeucht & Zeiger which, while powerful, seems restricted in application to finite automata over large alphabets in which each transition is labeled with a unique symbol.

Some Examples

Sometimes short regular expressions can be found through an analogue of Horner's rule for evaluating polynomials.

For example, consider the language

$$R_n := \{0, 0^2, 0^3, \ldots, 0^n\}.$$

We can specify this language with a regular expression of the form

$$0 + 00 + 000 + \cdots + 0^n;$$

such a regular expression is of length $\Theta(n^2)$.

However, the regular expression

$$0 + 0(0 + 0(0 + 0(\cdots)))$$

is of length $\Theta(n)$.

Less trivially, consider

$$S_n := \{0^i 1^i \ : \ 0 \leq i \leq n\}$$

and

$$T_n := \{0^i 1^j \ : \ 0 \leq i \leq j \leq n\}.$$

It is not hard to see that both S_n and T_n have regular expressions of size $\Theta(n)$.

By the technique of the longest string, all these bounds are optimal, up to a constant factor.

Conversion Problems

Converting from a regular expression to a DFA or NFA has been well-studied.

Theorem. (Leiss, Glushkov) Let E be a regular expression with $|\mathrm{alph}(E)| = n$. Then there exists

- an NFA accepting $L(E)$ with $\leq n + 1$ states; and
- a DFA accepting E with $\leq 2^n + 1$ states.

The bound for RE $\rightarrow$ NFA conversion is tight (consider the case of a language of a single word of length n).

RE to DFA conversion

Open Problem: What is the worst case in RE to DFA conversion for alphabet size ≥ 2?

The regular expression $E_r := (0 + (01^*)^{r-1}0)^*$ has $2r$ alphabetic symbols and Leung proved that the minimal DFA for $L(E_r)$ has 2^r states. This implies a worst case lower bound of $2^{n/2}$ for RE to DFA conversion.

Unary RE to DFA conversion

In the unary case, we have, with

$$g(n) := \max_{\sum n_i \leq n} \operatorname{lcm}(n_1, n_2, \ldots)$$

the following

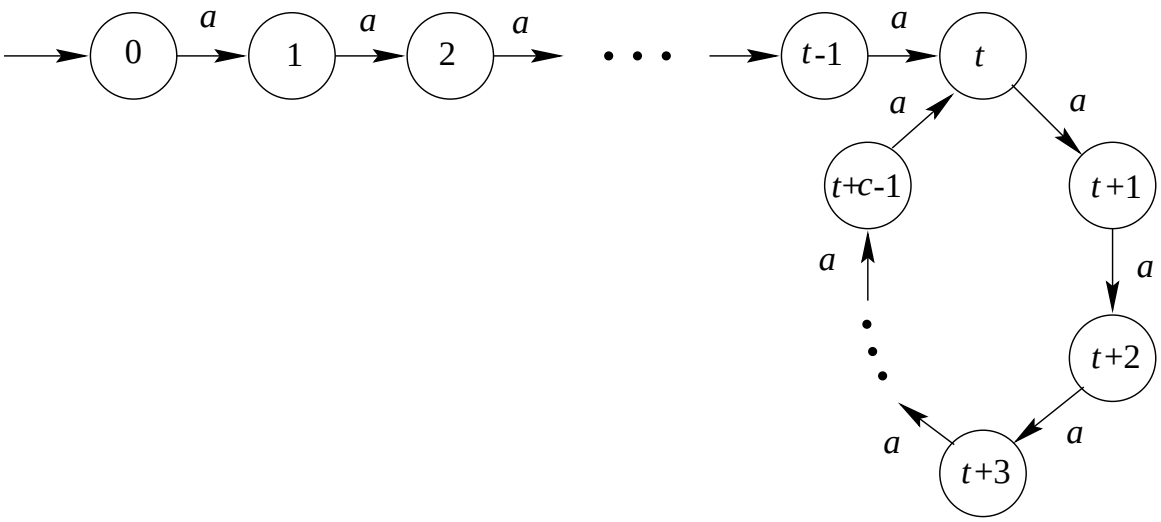
Theorem. (Mandl & ESW) Let E be a regular expression with $|\mathrm{alph}(E)| = n$. Then there exists a DFA accepting E with $\leq g(n)+(n-1)^2+2$ states. Furthermore for each n there exists a regular expression E with $|\mathrm{alph}(E)| = n$ such that the minimum equivalent DFA has $g(n)$ states.

DFA,NFA $\rightarrow$ RE conversion

We now turn to the converse problem, which has not received as much attention: converting from a DFA or NFA to an RE. For unary languages, we have the following easy result:

Theorem. If L is a unary regular language, accepted by a DFA M with n states, then it is specified by an RE of size $O(n)$. Furthermore, there exist infinitely many regular languages for which the smallest corresponding regular expression is of size $\Omega(n)$.

Proof. We may assume without loss of generality that the transition diagram of M is connected. This transition diagram has a "tail" of $t \geq 0$ states and a "cycle" of $c \geq 1$ states, with $n = t + c$.



Transition diagram of unary DFA
(accepting states not identified)

It follows that there exist sets $A \subseteq \{\epsilon, a, \ldots, a^{t-1}\}$ and $B \subseteq \{\epsilon, a, \ldots, a^{c-1}\}$ such that

$$L(M) = A + Ba^t(a^c)^* \qquad (1)$$

Now, using Horner's rule we can generate A with a regular expression of length $O(t+1)$, and B with a regular expression of length $O(c)$.

Unary DFA $\rightarrow$ RE conversion

For the lower bound, consider the regular language $\{a^{n-1}\}$. This is accepted by a DFA with $n+1$ states, and specified by a regular expression of length $n-1$. No smaller regular expression will suffice.

The case of unary NFA $\rightarrow$ RE conversion will be discussed by Andrew Martinez.

DFA,NFA $\rightarrow$ RE conversion

For languages over arbitrary alphabets we have the following upper bound, essentially due to McNaughton and Yamada [1960].

Theorem. If L is a regular language, accepted by a DFA or NFA $M = (Q, \Sigma, \delta, q_1, F)$ where $|Q| = n$ and $|\Sigma| = k$, it can be specified by an RE E such that $|\mathrm{alph}(E)| \leq nk4^n$.

Proof. Use the McNaughton-Yamada algorithm, defining a sequence of regular expressions

$$\alpha_{i,j}^{(l)}$$

that specify all words taking the automaton from state q_i to state q_j without passing through a state numbered higher than q_l. ∎

DFA,NFA $\rightarrow$ RE conversion

The upper bound of $nk4^n$ can be improved in certain special cases. For example, if the transition diagram of the NFA has no long paths, we can get a better bound, as the following theorem shows.

Theorem. Let G be a edge-labeled directed graph with n vertices and outdegree bounded above by D. Suppose the number of edges in a longest simple path (not repeating vertices or edges) in G that starts with vertex u is at most k. (By the length of the path we mean the number of edges.) Then for all vertices v, there is a regular expression E denoting all walks from u to v with $|\mathrm{alph}(E)| = O(D^{k+1}n^k)$.

We observe that if $k = o(n/\log n)$, then this bound is superior to the McNaughton-Yamada bound.

Planar DFA,NFA $\rightarrow$ RE conversion

Another improvement arises from other special classes of transition diagrams.

Theorem. Let $M = (Q, \Sigma, \Delta, q_0, F)$ be an NFA with r states, such that its transition diagram can be drawn in the plane with no edges crossing. Then there exists a regular expression E for $L(M)$ with $|\alpha(E)| \leq e^{O(\sqrt{r})}$.

Proof. The basic idea is divide-and-conquer.

We use the following well-known theorem of Lipton and Tarjan if G is an undirected planar graph with r vertices, then the vertices of G can be written as the (not necesssarily disjoint) union of two sets A and B such that

(a) there are no edges from $A - B$ to $B - A$,

(b) $|A - B|, |B - A| \leq 2r/3$, and

(c) $|A \cap B| \leq \sqrt{8r}$.

Planar DFA,NFA $\rightarrow$ RE conversion

We apply the Lipton-Tarjan theorem to the underlying undirected graph of the transition diagram for M for all sufficiently large M.

Here is the idea: we find a planar partition of the vertices of the transition diagram for M into A and B. Let $C = A \cap B$, $t = |C|$, and $C = \{c_1, c_2, \ldots c_t\}$. Now suppose q_u, q_v are both vertices of A; if $q_u \in A$ and $q_v \in B - A$ a similar argument applies.

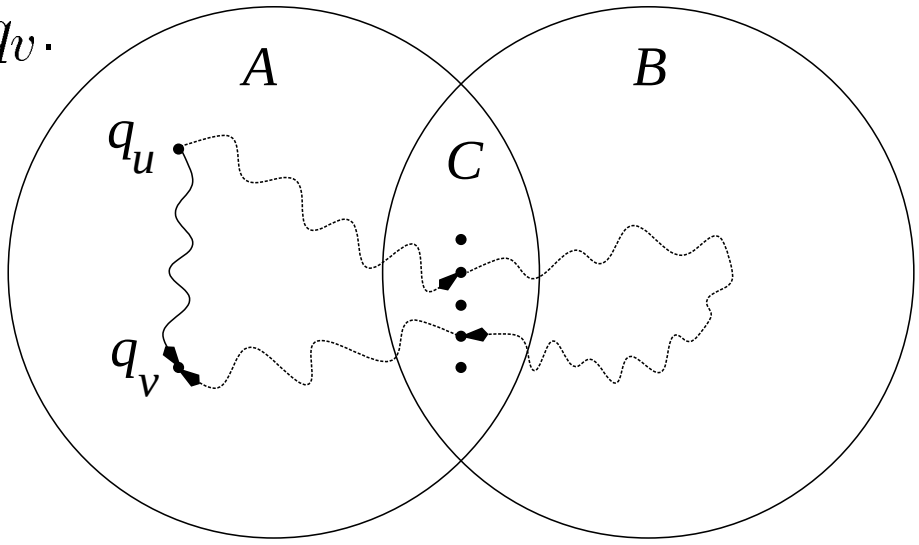
Planar DFA,NFA $\rightarrow$ RE conversion

We now decompose any walk connecting q_u to q_v as follows: either

(a) we go from q_u to q_v without leaving A; or

(b) we eventually leave A and enter $B - A$.

If (b), the walk must leave A through a vertex of C, since there are no edges connecting $A - B$ to $B - A$. Now the walk is in $B - A$. Since $q_v \in A$, at some point the walk must leave $B - A$ and return to A, and it can do so only through a vertex of C. This ping-pong between A and $B - A$ can occur arbitrarily many times. Eventually we leave $B - A$ and return to A, and finally we enter q_v.



Planar DFA,NFA $\rightarrow$ RE conversion

We create an NFA $M' = (Q', \Delta, p_0, \delta, F)$ that encodes this walk in its transitions, with a single symbol representing a sequence of transitions for M. This NFA has $O(t)$ states and $O(t^2)$ symbols denoting walks from x to y with vertices in S.

It now follows from the McNaughton-Yamada bound that we can construct a regular expression E for $L(M')$ with at most $O(t^2)4^{O(t)}$ alphabetic symbols. Now for each symbol we recursively determine regular expressions specifying the labels of walks from x to y with internal nodes in S.

If we let $T(n)$ denote the length of the resulting regular expression denoting all walks between two vertices in an NFA with n states, we get

$$T(n) \leq 4^{O(\sqrt{n})} T(2n/3 + \sqrt{8n}),$$

which has the solution $T(n) \leq e^{O(\sqrt{n})}$. ∎

DFA,NFA $\rightarrow$ RE conversion

We note that the same bound holds for any family of NFA's whose transition diagrams are of bounded genus or exclude any complete graph K_i as a minor.

DFA,NFA $\rightarrow$ RE conversion

As an example, consider the languages

$$L_n = \{x \in \{a,b\}^* \ : \ |x|_a \equiv |x|_b \equiv 0 \ (\mathrm{mod}\, n)\}.$$

The language L_n can be accepted by an NFA with n^2 states, as follows:

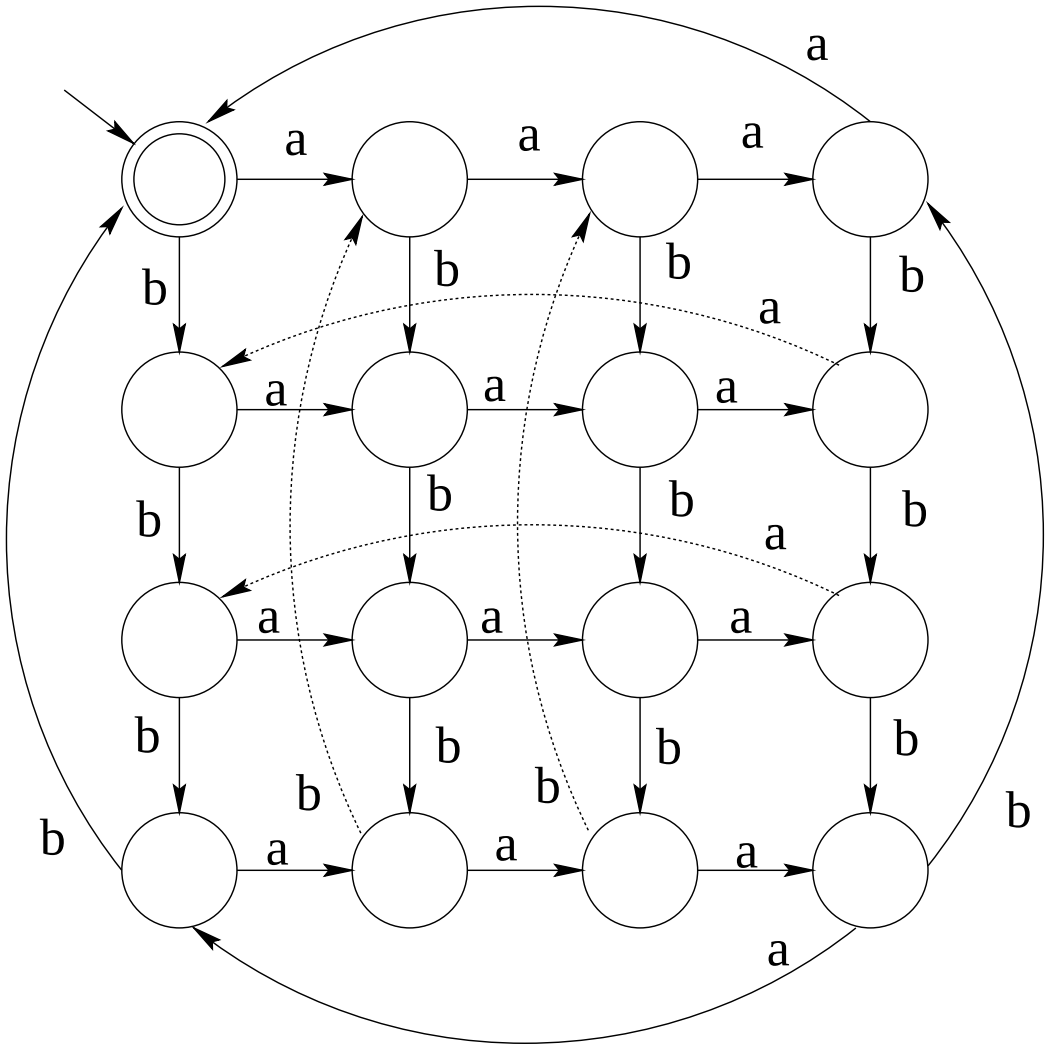


Figure 1: NFA for the language $L_4 = \{x \in \{a,b\}^* \ : \ |x|_a \equiv |x|_b \equiv 0 \ (\mathrm{mod}\ 4)\}$

DFA,NFA $\rightarrow$ RE conversion

Since the transition diagram can be embedded in a torus, it follows that there are regular expressions of size $e^{O(n)}$ for

$$L_n = \{x \in \{a,b\}^* \ : \ |x|_a \equiv |x|_b \equiv 0 \ (\mathrm{mod}\, 4)\}.$$

Divide-and-conquer, combined with a heuristic graph separator algorithm or an approximation algorithm will likely be a powerful technique for creating relatively short regular expressions for arbitrary automata in software packages such as Grail.

DFA,NFA $\rightarrow$ RE conversion

Theorem. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA or NFA with t states, over an alphabet of size $k = |\Sigma|$. Write $Q = \{q_0, q_1, \ldots, q_{t-1}\}$. Then for all $q_i, q_j \in Q$ and $n \geq 1$ there is a regular expression E denoting all walks of length n from q_i to q_j with $|\mathrm{alph}(E)| < k(t+1)n^{(\log_2 t)+1}$. The same bound holds for a regular expression denoting all walks of length $\leq n$.

Proof. Create a $t \times t$ matrix $M = (m_{i,j})_{0 \leq i,j < t}$ such that $m_{i,j}$ is a regular expression specifying all labeled edges taking A from state q_i to state q_j. If $M^k = (m_{i,j}^{(k)})_{0 \leq i,j < t}$, then $m_{i,j}^{(k)}$ is a regular expression specifying the labels of all length-k walks from q_i to q_j.

Now define $S(n) = \max_{0 \leq i,j < t} |\mathrm{alph}(m_{i,j}^{(n)})|$. Then

$$S(n) \leq t(S(\lfloor n/2 \rfloor) + S(\lceil n/2 \rceil)).$$

The solution to the recurrence

$$S(n) = t(S(\lfloor n/2 \rfloor) + S(\lceil n/2 \rceil))$$

is

$$2bt^{a+1} + (2^a - b)t^a,$$

where $n = 2^a + b$ with $0 \leq b < 2^a$.

We get

$$\begin{aligned} S(n) &\leq k(2bt^{a+1} + (2^a - b)t^a) \\ &< k(nt^{1+\log_2 n} + nt^{\log_2 n}) \\ &= kn(t+1)t^{\log_2 n} \\ &= kn(t+1)n^{\log_2 t} \\ &= k(t+1)n^{(\log_2 t)+1}, \end{aligned}$$

as desired. Here we have used the inequalities $b < n/2$, $2^a - b \leq n$, and $a \leq \log_2 n$.

Now consider a regular expression for all strings of length $\leq n$. Our analysis above can then be repeated without change, except for the following: we add ϵ to each entry on the diagonal of M. ∎

Corollary. If A is a DFA or NFA with r states over a k-letter alphabet, and $L(A)$ is finite, then there is a regular expression E specifying $L(A)$ with

$$|\mathrm{alph}(E)| \leq kr(r+1)(r-1)^{(\log_2 r)+1}.$$

Proof. Such a machine accepts no strings of length $\geq r$. Use the previous Theorem with $t = r$ and $n = r-1$. Finally, the strings accepted by A correspond to paths from the initial state to all final states, so we get an extra multiplicative factor of r. ∎

DFA,NFA $\rightarrow$ RE conversion

After this abundance of upper bounds, we state the following

Open Problem: Do there exist a constant c, a fixed alphabet Σ, and a family of languages L_n $(n \geq 1)$ defined over Σ, accepted by DFA's (resp. NFA's) with $f(n)$ states, such that the smallest corresponding regular expression has size $2^{cf(n)}$?

In fact, for fixed alphabet size there is no non-trivial lower bound known. Ehrenfeucht and Zeiger [1976] gave a family of examples with n states achieving exponential blow-up, but their alphabet size grew quadratically with n.

GRE $\rightarrow$ RE conversion

What is maximum possible size blow-up in going from a GRE to RE? Surprisingly, the blow-up is not even elementary. (By "elementary", we mean a function of the form

$$2^{2^{2^{\cdots 2^n}}}$$

where the number of 2's is bounded.) This seems to be a "folklore" result apparently first published by Dang (1973).

Theorem. The worst-case size blow-up from GRE to RE is not elementary.

Proof. (A. Meyer) Suppose every GRE R of size n has an equivalent RE R' having $\leq f(n)$ alphabetic symbols. where $f(n)$ is an elementary function of the form $2^{2^{\cdots 2^n}}$ for some finite number of levels of exponents.

- By Leiss's theorem, R' can be converted to an equivalent DFA having at most $\leq 2^{f(n)+1}$ states.

- But a DFA M of t states accepts the empty language if and only if it accepts no string of length $< t$.
- Hence we have the following algorithm A for determining if $L(R) = \emptyset$: for each string x of length $\leq 2^{f(n)+1}$, determine if $x \in L(R)$. If the answer is no for all these x, return "yes"; otherwise, return "no".
- Since we can decide if $x \in L(R)$ in time polynomial in $x + |R|$, the running time of algorithm A is elementary.
- But this contradicts a well-known result of Stockmeyer that there is no algorithm that runs in elementary time which decides if the language specified by a GRE is empty. ∎

Operations on Regular Expressions

How do various operations on regular languages affect the length of the regular expression representing them?

In particular, what is maximum blow-up in going from an RE for L to the shortest possible RE for $\overline{L} = \Sigma^* - L$? For the unary case optimal results are known.

Theorem. If E is a regular expression with $|\mathrm{alph}(E)| = n$ specifying a unary regular language L, then there exists a regular expression E' of length $e^{O(\sqrt{n \log n})}$ specifying $\overline{L}$. Furthermore, there exist infinitely many regular languages for which this bound is achieved.

Proof. Convert the RE to an NFA using the usual method; convert the NFA to a DFA using the subset construction, interchange accepting and non-accepting states, and convert the resulting DFA to an RE. The first, third, and fourth steps

involve only a linear blow-up, while the second can increase the size of the resulting DFA by $e^{\sqrt{n \log n}}$, as is well-known. This gives a bound of $e^{O(\sqrt{n \log n})}$.

This upper bound can be achieved using an expression like

$$\epsilon + r_1(00)^* + r_2(000)^* + r_4(00000)^* + \cdots + r_{p-1}(0^p)^*$$

where

$$r_n = \{0, 00, 000, \ldots, 0^n\}$$

and p is the largest prime $\leq n$. This regular expression is of length $t = O(n^2/(\log n))$. However, the regular expression for the complement is of length $e^{O(\sqrt{t \log t})}$, since the shortest string in the complement is $0^{p_1 p_2 \cdots p_k}$, where $p_1, p_2, \ldots, p_k = p$ are the primes $\leq n$, and by the prime number theorem we have $p_1 p_2 \cdots p_k = e^{n(1+o(1))}$. ∎

Complement of a Regular Expression

How about the case of larger alphabets? For the upper bound, the best result we currently know is doubly-exponential. First convert the RE to a DFA, interchange accepting and non-accepting states; and, finally, convert back to an RE. This gives an upper bound of the form c^{2^n} for a constant c.

Complement of a Regular Expression

For a lower bound, we have the following examples.

Theorem. Define

$$E_n = (0+1)^* \, 0 \, (0+1)^n \, 0 \, (0+1)^*.$$

Then

(a) E_n is a regular expression with $|\mathrm{alph}(E_n)| = 2n + 6$;

(b) $L(E_n)$ is accepted by a minimal DFA with $2^{n+1} + 1$ states;

(c) $L(E_n)$ is accepted by an NFA with $n + 3$ states;

(d) $\overline{L(E_n)}$ is not accepted by any NFA with $< 2^n$ states;

(e) $\overline{L(E_n)}$ is not specified by any regular expression E with $|\mathrm{alph}(E)| < 2^n - 1$.

Complement of a Regular Expression

Proof.

(a) $E_n = (0+1)^* \, 0 \, (0+1)^n \, 0 \, (0+1)^*$. is a regular expression with $|\mathrm{alph}(L_n)| = 2n + 6$

Clear.

(b) $L(E_n)$ is accepted by a minimal DFA with $2^{n+1} + 1$ states

Follows because we can accept L_n with $2^{n+1} + 1$ states by using a state labeled with every string of length $n+1$, indicating the last $n+1$ symbols seen, plus one more state for the accepting state once we detect $0(0+1)^n 0$. The initial state is $[111 \cdots 1]$.

To see this is minimal, use the Myhill-Nerode theorem. Note that for the states labeled with strings, say w and w', we must have w and w' differ in some letter. Without loss of generality, assume w has a 0 in position i but w' does

not. Now append enough 1's and then a 0 to make the 0 in position i occur n symbols from the added 0. Now w will be accepted and w' won't be. To see the additional accepting state is inequivalent from all the labeled states, note that ϵ distinguishes them.

(c) $L(E_n)$ is accepted by an NFA with $n+3$ states;

Easy. Use a state that loops on everything, followed by a transition on 0, followed by n transitions on everything, followed by a transition on 0, followed by a state that loops on everything.

(d) $\overline{L(E_n)}$ is not accepted by any NFA with $< 2^n$ states

Use the result of Birget mentioned above. If $w \in \{0,1\}^*$, let $\tilde{w}$ be the string obtained from w by changing every 0 to a 1 and vice-versa. In Birget's theorem let S be the set of pairs

$$\{(w, \tilde{w}) \ : \ w \in \{0,1\}^n\}.$$

Clearly $w\tilde{w} \in L(E_n)$, since any two symbols that are n positions apart are different. Now if $w \neq x$ then (say) there is a 0 in a position in w that has a 1 in the corresponding position of x. Then $w\tilde{x} \notin L$ since there are two 0's n positions apart.

(e) $\overline{L(E_n)}$ is not specified by any regular expression E with $|\mathrm{alph}(E)| < 2^n - 1$.

If it were specified by a regular expression with $2^n - 1$ symbols there would be an NFA with 2^n states by Leiss's theorem, a contradiction. ∎

This leads to

Open Problem: What is the maximum achievable blow-up in going from a regular expression for L to a shortest regular expression for $\overline{L}$?

Intersection of Two Regular Expressions

We now turn to intersection. Given RE's E_1, E_2, with m and n alphabetic symbols, respectively, what is the shortest regular expression for $L(E_1) \cap L(E_2)$? By converting to an NFA, forming the direct product, and converting back to an RE, we get an upper bound of the form $c^{(m+1)(n+1)}$. However, we do not currently know an example where the blow-up exceeds cmn.

Open Problem: What is the maximum achievable blow-up in going from regular expressions for L_1, L_2 to a shortest regular expression for $L_1 \cap L_2$?

In the unary case we can get an upper bound of $O((mn)^2)$ using Chrobak normal form. Can this be achieved?

Intersection of Two Regular Expressions

There are some interesting variations on these problems.

Zhivko Nedev of Wilfrid Laurier University (personal communication, June 2001), asked, what is the worst-case in going from a regular expression R to a regular expression for $L(R) \cap \Sigma^n$?

From a previous theorem, we know that we can find a regular expression E for $L(R) \cap \Sigma^n$ with

$$|\mathrm{alph}(E)| \leq k(r+1)(r+2)n^{\log_2(r+1)+1},$$

where $k = |\Sigma|$ and $r = |\mathrm{alph}(R)|$.

A similar bound holds for $L(R) \cap \Sigma^{\leq n}$.

Shortest String Not Specified by a Regular Expression

Suppose we have a regular expression E with $|\mathrm{alph}(E)| = n$ over a finite alphabet Σ with $L(E) \neq \Sigma^*$. How long can the shortest string *not* specified by E be?

An upper bound of 2^n can be obtained as follows: first, convert E to a DFA of at most $2^n + 1$ states. Next, interchange accepting and non-accepting states, and look for the shortest string accepted. This is of length at most 2^n. Can this bound be achieved?

We can get 2^{cn} as follows: we create a regular expression that represents all strings except the numbers $1, 2, \ldots, 2^n - 1$ (suitably encoded) listed in increasing order, separated by a special delimiter symbol #. To verify that a string is not of this form, we only need verify either that it is syntactically incorrect, or that it contains a substring of the form $\#x\#w\#$ where w does not

denote a binary number that is 1 more than that denoted by x.

To make life even easier, we will assume that the representations for n also include those for $n-1$. We do this by treating the base-2 representations of both numbers in parallel, writing one expansion above the other and using the alphabet $\{0,1\} \times \{0,1\}$. Thus for $n = 3$ the only string not specified will be

$$\#\begin{matrix}0\\0\end{matrix}\begin{matrix}0\\0\end{matrix}\begin{matrix}0\\1\end{matrix}\#\begin{matrix}0\\0\end{matrix}\begin{matrix}0\\1\end{matrix}\begin{matrix}1\\0\end{matrix}\#\begin{matrix}0\\0\end{matrix}\begin{matrix}1\\1\end{matrix}\begin{matrix}0\\1\end{matrix}\#\begin{matrix}0\\1\end{matrix}\begin{matrix}1\\0\end{matrix}\begin{matrix}0\\0\end{matrix}\#\begin{matrix}1\\1\end{matrix}\begin{matrix}0\\0\end{matrix}\begin{matrix}0\\1\end{matrix}\#\begin{matrix}1\\1\end{matrix}\begin{matrix}0\\1\end{matrix}\begin{matrix}1\\0\end{matrix}\#\begin{matrix}1\\1\end{matrix}\begin{matrix}1\\1\end{matrix}\begin{matrix}0\\0\end{matrix}\#\begin{matrix}1\\1\end{matrix}\begin{matrix}1\\1\end{matrix}\begin{matrix}1\\1\end{matrix}\#.$$

We now recode this over the alphabet $\{a, b, c, d, e\}$, as follows:

$$\begin{matrix}0\\0\end{matrix} \to a; \qquad \begin{matrix}0\\1\end{matrix} \to b;$$

$$\begin{matrix}1\\0\end{matrix} \to c; \qquad \begin{matrix}1\\1\end{matrix} \to d;$$

$$\# \to e.$$

Thus, for $n = 3$ the only string not specified will

be

$$eaabeabceadbebccedabedbceddbe.$$

Now we construct the necessary regular expressions. We use the abbrevation $\Sigma = \{a, b, c, d, e\}$.

1. E_1: the first $n+2$ symbols are not $ea^{n-1}be$.
   For $n = 4$ this is

   $$\epsilon + e(\epsilon + a(\epsilon + a(\epsilon + a(\epsilon + b))))+$$
   $$((\epsilon + eaaab)(a + b + c + d) + e(\epsilon + a(\epsilon + a))$$
   $$(b + c + d + e)eaaa(a + c + d + e))\Sigma^*.$$

   We have $|\mathrm{alph}(E_1)| = 4n + 18$.

2. E_2: the last $n+2$ symbols are not $ed^{n-1}be$.
   For $n = 4$ this is

   $$\epsilon + (\epsilon + (\epsilon + (\epsilon + (\epsilon + d)d)d)b)e+$$
   $$\Sigma^*((a + b + c + d)(\epsilon + dddbe) +$$
   $$(a + c + d + e)e + (a + b + c + e)$$

   We have $|\mathrm{alph}(E_2)| = 3n + 20$.

3. E_3: the string has two consecutive e's: $\Sigma^* ee \Sigma^*$.
   We have $|\mathrm{alph}(E_3)| = 12$.

4. E_4: the string has a block of more than n consecutive digits: $\Sigma^*(a+b+c+d)^{n+1}\Sigma^*$. We have $|\mathrm{alph}(E_4)| = 4n+14$.

5. E_5: the string has a block of length $< n$ between two e's: $\Sigma^* e(a+b+c+d+\epsilon)^{n-1}e\Sigma^*$. We have $|\mathrm{alph}(E_5)| = 4n+8$.

6. E_6: the string has a block of the form $e\binom{x}{y}e$ where $[y]_2 \neq [x]_2 + 1$:

   $$\Sigma^* e(a+d) * (c+e+bc^*(a+b+d))\Sigma^*.$$

   We have $|\mathrm{alph}(E_6)| = 20$.

7. E_7: the string has a block of the form $e\binom{x}{y}e\binom{y'}{z}e$ where $y \neq y'$:

   $$\Sigma^*(\ (a+c)\Sigma^n(c+d) + (b+d)\Sigma^n(a+b)\ )\Sigma^*.$$

   We have $|\mathrm{alph}(E_7)| = 10n+18$.

Our regular expression

$$E_1 + E_2 + E_3 + E_4 + E_5 + E_6 + E_7$$

proves

Theorem. For each $n \geq 3$ there is a regular expression E over the alphabet $\{a, b, c, d, e\}$ with $|\mathrm{alph}(E)| = 25n + 110$ such that the shortest string not specified is of length $(2^n - 1)(n+1)+1$.

We can convert this example to an example over the alphabet $\{0, 1\}$, as follows. First, we apply the morphism sending $a \to 000$, $b \to 001$, $c \to 010$, $d \to 011$, and $e \to 111$ to each of the regular expressions $E_1, E_2, \ldots, E_7$. Next, we supplement the resulting regular expression with two additional ones:

8. E_8: strings of length divisible by three, containing one of the three excluded triples:

$$((0+1)(0+1)(0+1))^*(100+101+110)$$
$$((0+1)(0+1)(0+1))^*.$$

   We have $|\mathrm{alph}(E_8)| = 21$.

9. E_9: strings of length not divisible by three: $((0+1)(0+1)(0+1))^*(0+1)(0+1+\epsilon)$. We have $|\mathrm{alph}(E_9)| = 10$.

This gives

Theorem. For each $n \geq 3$ there is a regular expression E over the alphabet $\{0,1\}$ with $|\mathrm{alph}(E)| = 75n + 361$ such that the shortest string not specified is of length $3(2^n - 1)(n + 1) + 3$.

Open Problem: For a regular expression E over $\{0,1\}$, define

$$\mathrm{lssns}(E) = \begin{cases} \min\{|x| \ : \ x \in \Sigma^* - L(E)\}, \\ \qquad \text{if } L(E) \neq \Sigma^* \ ; \\ 0, \\ \qquad \text{if } L(E) = \Sigma^*. \end{cases}$$

Define

$$h(n) = \max_{|\mathrm{alph}(E)|=n} \mathrm{lssns}(E).$$

By Theorem we know $h(n) > 2^{n/75}$ for all n sufficiently large, and as above $h(n) \leq 2^n$. Find a closed form for $h(n)$ or better upper and lower bounds.