Formal Languages and Number Theory

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

State Complexity

The *state complexity* of a regular language L, $\mathrm{sc}(L)$, is the minimum number of states needed to accept it by a DFA.

The problem:
Given languages L, L' with state complexity n, n' respectively, what are good bounds on the state complexity of $L \cup L'$, LL', L^*, etc.?

For the state complexity of intersection, we have the following bound:

Proposition. *We have*

$$\mathrm{sc}(L \cap L') \leq \mathrm{sc}(L)\mathrm{sc}(L').$$

Proof. Use the usual direct product construction.

State Complexity of Intersection

The upper bound of $\mathrm{sc}(L)\mathrm{sc}(L')$ can be achieved if L, L' are over an alphabet of size at least 2:

Proposition. $(\mathrm{S.\ Yu.})$ *Define*

$$L := \{x \in \{a,b\}^* \ : \ |x|_a \equiv 0 \pmod{n}\};$$

$$L' := \{y \in \{a,b\}^* \ : \ |y|_b \equiv 0 \pmod{n'}\}.$$

Then

$$\mathrm{sc}(L \cap L') = nn'.$$

But what if L, L' are *unary*, that is, defined over an alphabet of one symbol?

Clearly if $\gcd(n, n') = 1$ then the bound nn' can again be achieved, by taking $L = (a^n)^*$ and $L' = (a^{n'})^*$.

But what if $\gcd(n, n') > 1$?

State Complexity of Intersection for Unary Languages

A connected unary DFA has the property that its transition diagram consists of

- a *tail* of $t \geq 0$ states and
- a *cycle* of $c \geq 1$ states.

It is then not hard to prove that

Theorem. *Let M, M' be unary DFA's with tails of size t, t' and cycles of size c, c', respectively. If L, L' are the corresponding languages, we have*

$$\mathrm{sc}(L \cap L') \leq \max(t, t') + \mathrm{lcm}(c, c'). \qquad (1)$$

Furthermore, for all $t, t' \geq 0$ and $c, c' \geq 1$ there exist unary languages for which the bound (1) is achieved.

For example, if $t \geq t'$, take

$$\begin{aligned} L &= a^{t+c-1}(a^c)^*; \\ L' &= a^r(a^{c'})^*; \\ r &= t - 1 \bmod c'. \end{aligned}$$

Two New Number-Theoretic Functions

Thus, to estimate the worst-case behavior for the state complexity of intersection of unary languages with n and n' states, respectively, we must estimate the function

$$F(n,n') = \max_{\substack{1\le c\le n\\ 1\le c'\le n'}} \left(\max(n-c, n'-c') + \operatorname{lcm}(c,c')\right).$$

This in turn suggests studying the somewhat simpler and more natural function

$$G(n,n') = \max_{\substack{1\le c\le n\\ 1\le c'\le n'}} \operatorname{lcm}(c,c').$$

- The asymptotic behavior of F and G is still not known precisely
- There is a relation to JACOBSTHAL'S function $g(n)$, which is the *least integer r such that every set of r consecutive integers contains at least one integer relatively prime to n.*
- IWANIEC proved [1978] using the linear sieve that $g(n) = O((\log n)^2)$.
- It then follows that if $n \le n'$, we have $F(n,n') \ge G(n,n') \ge nn' - c(\log n)^2 n$ for some constant c.

Automatic Complexity: Introduction

- We're interested in a measure of complexity for finite strings x over a finite alphabet, typically $\{0, 1\}$.
- Any such measure should reflect, in some sense, how "complicated" the string x is.
- Old idea: KOLMOGOROV-CHAITIN complexity
  - $K(x) =$ size of the shortest pair (M, y) where M is a Turing machine description and y is an input, such that M outputs x on input y.
  - Extremely useful; see book by LI and VITÁNYI.
  - The map $x \to (M, y)$ can be viewed as the best possible way to compress x.
  - Unfortunately not computable; no effective procedure for finding the pair (M, y).
  - Also depends on the particular model of universal TM chosen.
- Open problem: is it true that $K(xx) > K(x)$ for almost all strings x? Note: it is easy to see that $K(xx) = K(x) + O(1)$.

Alternatives to Kolmogorov Complexity

- Can we find a computable measure of complexity?
- Idea: replace Turing machine with a less powerful model.
- Example: replace the Turing machine with a context-free grammar (CFG).
- We choose some measure of the complexity of a context-free grammar, and then ask for the "smallest" grammar G such that $L(G) = \{x\}$.
- If we demand that the CFG be in Chomsky normal form (i.e., all productions are of the form $A \to BC$ or $A \to a$ where A, B, C are variables and a is a terminal), then we get a well-known measure of complexity associated with "word chains".
- Studied by Diwan; Berstel & Brlek; Roth; Arnold & Brlek, Althöfer, etc.
- Open problem: what is the complexity of computing a minimum word chain or its length?
  - Probably a difficult problem to study, since in the unary case we get the well-studied area of "addition chains", which are discussed in Knuth, Volume II.

Automatic Complexity

- Alternate idea: consider replacing the Turing machine with a deterministic finite automaton (DFA).
- First attempt: find a smallest DFA M such that $L(M) = \{x\}$.
- Uninteresting: if $|x| = n$, a smallest such DFA always has exactly $n + 2$ states.
- If a DFA M accepts a string x, but no other strings of length $|x|$, we say M *accepts* x *uniquely*.

Definition. $A(x)$, the *automatic complexity* of x, is the smallest number of states in any DFA M that accepts x uniquely.

- There may be many such DFA's with the smallest number of states.
- We do not care how M behaves on strings that are shorter or longer than x.
- We can view the map $x \to (M, |x|)$ as a compression algorithm for x.

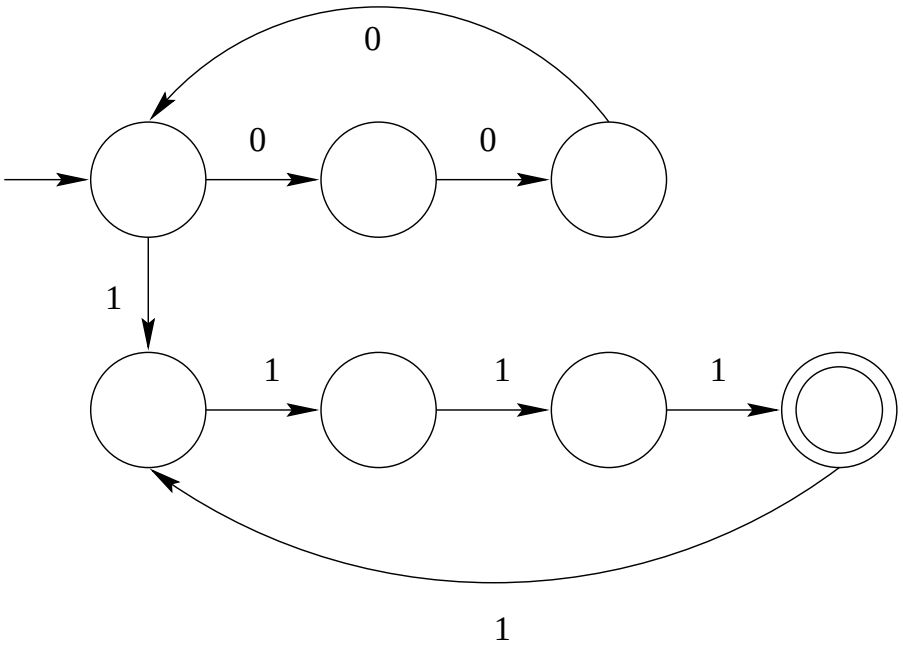
Basic Properties of Automatic Complexity

- $A(x) \leq |x| + 2$, since we can accept a string with a chain of $|x| + 1$ states, plus one more for a dead state.
- Hence A is computable, since we can simply
  - Enumerate all DFA's with $\leq |x| + 2$ states, and
  - For each such automaton, we check if it accepts x and if it rejects all other strings of length $|x|$.
- Better algorithms are possible — for example, it is possible, given a DFA and a string x, to determine efficiently if x is accepted uniquely.
- But we still do not know an efficient algorithm for computing A, or have a proof that computing A is, say, NP-hard
- On the other hand, given a description of a DFA M which uniquely accepts x, and the length $n = |x|$, we can efficiently recover x.

How to Save States

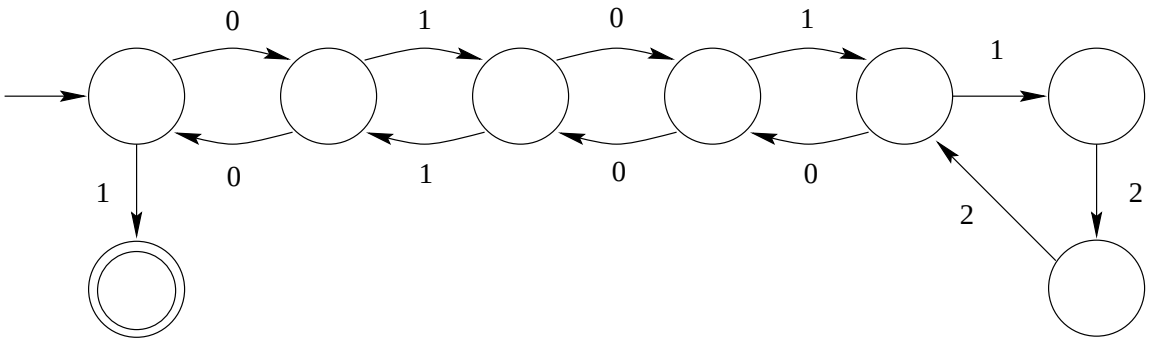
You can save states with loops:

For example, the automaton below shows that $A(0^9 1^8) \leq 8$. (Unspecified transitions go to a "dead state" which is not shown.)



How to Save States

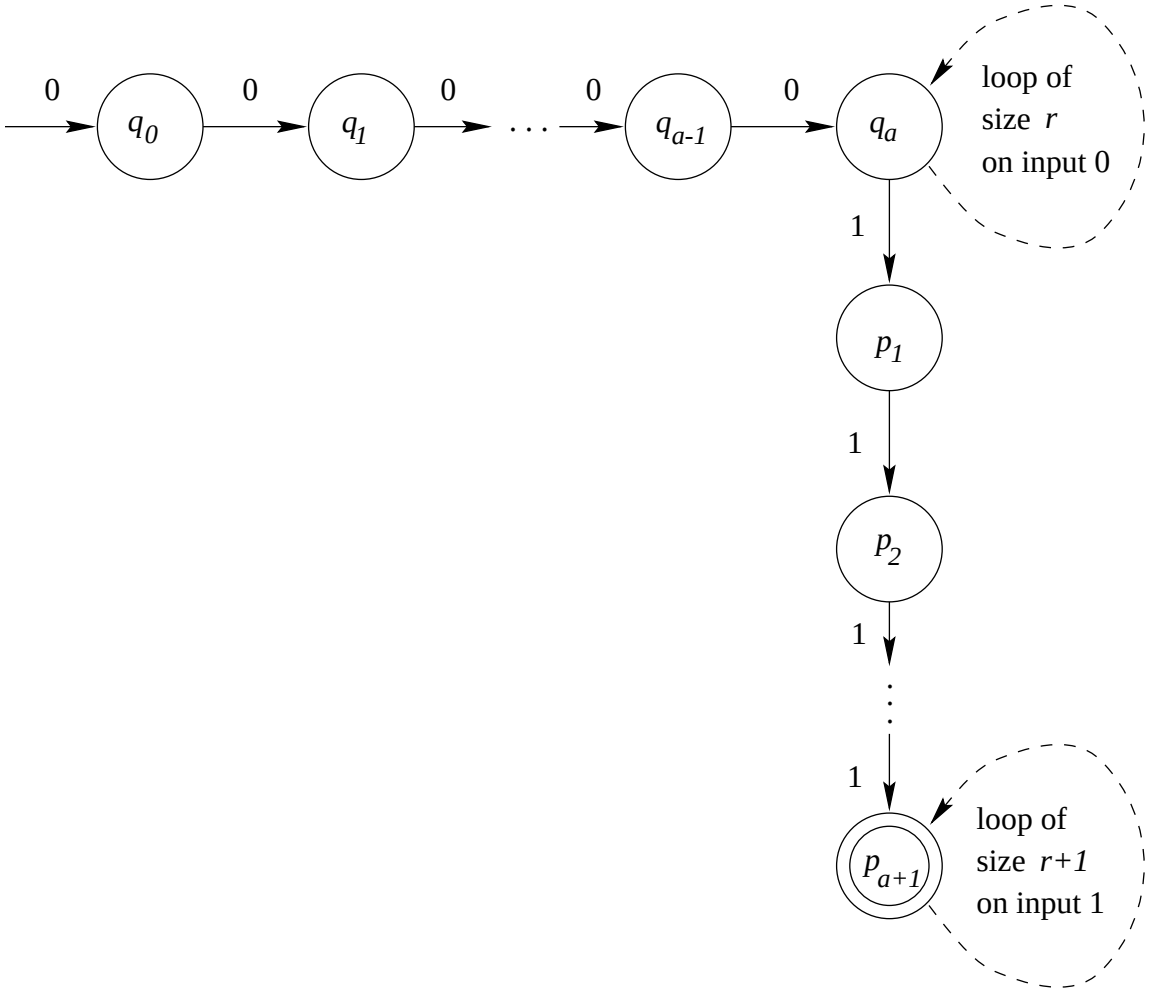
You can reuse states, if the string is of the form $x\, y\, z\, \overline{y}^R\, w$, as follows:



Some Specific Examples

Theorem. We have $A(0^n 1^n) = O(\sqrt{n})$.

Proof. Assume $n \geq 1$. Let $r = \lfloor \sqrt{n} \rfloor$, so $r^2 \leq n < (r+1)^2$. Write $n = r^2 + a$. Then $0 \leq a \leq 2r$ and $r \geq 1$. Then we can accept $0^n 1^n$ with an automaton of the form below. (Unspecified transitions go to a "dead state" which is not shown.)



This automaton does indeed accept 0^n1^n because

1. We go from state q_0 to state q_a on 0^a;
2. We then go around the loop at q_a r times;
3. Next on 1^a we go from q_a to p_{a+1};
4. Finally, we go around the loop at p_{a+1} $r-1$ times.

This path accepts $0^a(0^r)^r11^a(1^{r+1})^{r-1} = 0^{r^2+a}1^{r^2+a}$.

On the other hand, we claim that this DFA accepts no other string of length $2n$. Suppose it did. Then any accepting path must go around the loop on q_a b times and the loop on p_{a+1} c times. Then

$$2n = a + br + a + 1 + c(r+1).$$

Since $n = r^2 + a$, it follows that $2r^2 - 1 = br + c(r+1)$. Reducing modulo r, we get $c \equiv -1 \pmod{r}$. Thus $c \in \{r-1, 2r-1, \dots\}$. But if $c \geq 2r-1$ then the string would be of length $\geq (2r-1)(r+1) + 2a + 1 = 2r^2 + r - 1 + 2a + 1 \geq 2n + r > 2n$, a contradiction.

Finally, our automaton uses $a + 1 + r - 1 + a + 1 + r = 2r + 2a + 1 \leq 6r + 1 \leq 6\sqrt{n} + 1$ states. ∎

Lower Bound for $A(0^n1^n)$

We now show that the bound of $O(\sqrt{n})$ is tight. We use the following lemma.

Lemma. Let c, d be integers ≥ 1. Suppose the linear diophantine equation $N = xc + yd$ is solvable in integers, i.e., suppose $\gcd(c, d) \mid N$. If $N > 2cd - c - d$, then the linear diophantine equation $N = xc + yd$ has at least two solutions in non-negative integers x, y.

Theorem. Any DFA that uniquely accepts 0^n1^n must have at least $\sqrt{n} - 1$ states.

Generalization to $A(a_1^n a_2^n \cdots a_k^n)$

Can we generalize our result for $0^n 1^n$ to strings of the form $a_1^n a_2^n \cdots a_k^n$?

To do so, we would need loops on each of the symbols $a_1, a_2, \ldots, a_k$. If the sizes of the loop corresponding to a_i is s_i, and we go around it e_i times, then we want $e_i s_i$ to be less than but close to n, say $n - l_i$, and we want to have exactly one solution to the equation

$$\sum_{1 \le i \le k} e_i s_i = kn - \sum_{1 \le i \le k} l_i.$$

One way to do this is to pick k numbers, relatively prime in pairs, say n_1, n_2, ..., n_k, each less than $n^{1/k}$ but relatively close to it in size. Let $N = n_1 n_2 \cdots n_k$. Let the loop sizes be N/n_i for $1 \le i \le k$. It is now possible to show that the linear diophantine equation

$$e_1(N/n_1) + e_2(N/n_2) + \cdots + e_k(N/n_k) = C$$

has a unique solution

$$(e_1, e_2, \ldots, e_k) = (n_1 - 1, n_2 - 1, \ldots, n_k - 1)$$

for suitably chosen C.

Modulo the existence of the relatively prime numbers, we have shown

Generalization to $A(a_1^n a_2^n \cdots a_k^n)$

Theorem. Let $a_1, a_2, \ldots, a_k$ be k distinct symbols. Then $A(a_1^n a_2^n \cdots a_k^n) = O(n^{1-1/k})$.

But it still remains to show that one can find the k relatively prime numbers.

It suffices to show that for each integer $k \geq 1$ there exists a number $f(k)$ such that every $f(k)$ consecutive positive integers contains a subset of size k that is pairwise relatively prime. This can be done as follows:

Lemma. Let p_n denote the n'th prime, with $p_1 = 2$. Define $Q_n = p_1 p_2 \ldots p_n$ for $n \geq 1$. Then we may take $f(k) = Q_k$.

Proof. We claim that for each integer $r \geq 0$ and $n \geq 1$ the sets $U_r := \{rQ_n + 2, rQ_n + 3, \ldots, rQ_n + p_n\}$ and $V_r := \{(r+1)Q_n - 2, (r+1)Q_n - 3, \ldots, (r+1)Q_n - p_n\}$ are individually pairwise relatively prime. U_r. ∎

Jacobsthal's Function

We have seen that we have $f(k) \leq p_1 p_2 \cdots p_k$, the product of the first k primes. By the prime number theorem we know that $C = e^{k(\log k)(1+o(1))}$. This exponential bound can be significantly improved.

Let $n \geq 1$ be an integer, and let

$$c_1 = 1 < c_2 < \cdots < c_r$$

be the integers in the range $[1, n]$ that are relatively prime to n, where $r = \varphi(n)$.

Then Jacobsthal's function $g(n)$ is defined to be

$$\max_{2 \leq i \leq r} (c_i - c_{i-1}),$$

i.e., the size of the maximum gap between two consecutive numbers relatively prime to n.

Also define

$$C(r) = \max_{\omega(n)=r} g(n),$$

where ω counts the number of distinct prime divisors.

Jacobsthal's function

Let I be a set of consecutive positive integers. We call a subset of I that is pairwise relatively prime a $clique$.

Define $f(m)$ to be the least integer n such that every set of n consecutive positive integers contains a clique of size m.

Theorem. We have $f(m) = C(m-1)$ for $m \geq 1$.

Proof.

- Suppose $C(m-1) = s$.
- Then there exist $m-1$ prime numbers $p_1, p_2, \ldots, p_{m-1}$ such that $g(p_1 p_2 \cdots p_{m-1}) = s$.
- Then there exists an integer t such that the numbers $t+1, t+2, \ldots, t+s-1$ all have a factor in common with $p_1 p_2 \cdots p_{m-1}$, i.e., each integer in

  $$\{t+1, t+2, \ldots, t+s-1\}$$

  is divisible by at least one of the primes $p_1, p_2, \ldots, p_{m-1}$.
- Hence the largest clique in $\{t+1, t+2, \ldots, t+s-1\}$ is of size $\leq m-1$.
- Therefore $f(m) > s-1$, and so $f(m) \geq C(m-1)$.

Jacobsthal's function

- Now suppose $f(m) = s$.
- Then there exists a set $s-1$ consecutive positive integers, say $S = \{t+1, t+2, \ldots, t+s-1\}$, which contains no clique of size m.
- However

  $$S \cup \{t+s\} = \{t+1, t+2, \ldots, t+s-1, t+s\}$$

  must contain a clique of size m, so S must contain a clique of size $m-1$, say $T = \{a_1, a_2, \ldots, a_{m-1}\}$.
- Without loss of generality we may assume $1 \notin T$.
  - For if $1 \in T$, then $1 \in S$ and hence $t = 0$.
  - Then $\{1, 2, \ldots, s\}$ contains a clique of size m.
  - Now $\{2, \ldots, s\}$ cannot contain a clique of size $\geq m$, for if it did, $\{1, 2, \ldots, s\}$ would contain a clique of size $m+1$, and so $\{1, 2, \ldots, s-1\}$ would contain a clique of size m, a contradiction.
  - Thus in this case we may instead take $S = \{2, 3, \ldots, s\}$.
- Since $1 \notin T$, at least one prime divides each member of T.
- Further, we cannot have the same prime dividing two different members of T, for then T would not be a clique.

- Thus there is a set of $m-1$ distinct primes, say $P = \{p_1, p_2, \ldots, p_{m-1}\}$, such that each prime of P divides exactly one element of T.
- Now suppose there exists $a \in S$ such that no prime of P divides a.
- Then $T \cup \{a\}$ would be a clique of size m, a contradiction.
- It follows that $\gcd(a, p_1 p_2 \cdots p_{m-1}) > 1$ for all $a \in S$.
- But then we have $g(p_1 p_2 \cdots p_{m-1}) \geq s$.
- Now $C(m-1) \geq g(p_1 p_2 \cdots p_{m-1})$, so $C(m-1) \geq f(m)$.

Thus we have proved $f(m) = C(m-1)$. ∎

We now use a result of IWANIEC, which shows that $C(m) = O(m^2(\log m)^2)$. It follows that $f(m) = O(m^2(\log m)^2)$.