New Directions in 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

State Complexity

- Let L be a regular language
- Let $\mathrm{sc}(L)$ denote its state complexity, that is, the smallest number of states in any deterministic finite automaton (DFA) accepting L.
- We assume the DFA is complete, that is, the transition δ function is defined for all elements in $Q \times \Sigma$.

Main Themes of State Complexity

- Examining how the number of states (or transitions, or ...) changes when the representation of the language changes
  - Typical example: NFA to DFA conversion has $n \longrightarrow 2^n$ blowup for alphabet size $k \geq 2$
- Examining how $\mathrm{sc}(L)$ changes when various operations are performed on L
  - Typical example: $\mathrm{sc}(L_1 \cap L_2) \leq \mathrm{sc}(L_1)\mathrm{sc}(L_2)$, and this bound is tight for alphabet size $k \geq 2$

History of State Complexity

Representational state complexity has its origins in the work of

- V. A. Uspenskii, Yu. L. Ershov (1962), G. M. Korpelevich (1963)
  - Transducers versus ordinary automata
- O. B. Lupanov (1963), Frank R. Moore (1971)
  - NFA to DFA conversion
- Ju. I. Ljubich (1964), Marek Chrobak (1986, 2003)
  - unary NFA to DFA conversion

Representational State Complexity

- A. R. Meyer and M. J. Fischer (1971)
- Mandl (1973)
- Sakoda & Sipser (1978)
- Leiss (1981)

Operational State Complexity

Operational state complexity has its origins in the work of

- A. N. Maslov (1970)
  - state complexity of union, concatenation, Kleene *
- Maslov (1973)
  - cyclic shift
- Sheng Yu and co-authors (1992)
- Jean-Camille Birget (1992)
- M. Domaratzki (2001)
- Jirásková and co-authors (2005)

Other variations of interest

- Ambiguity
  - Ravikumar & Ibarra (1989)
  - Leung and co-authors (1993)
- Nondeterministic state complexity
  - Ellul (2002) and Holzer & Kutrib (2003)
- Transition complexity of NFA's
  - M. Domaratzki and K. Salomaa (2006)
- Finite languages
  - Câmpeanu, Culik, K. Salomaa, Yu (1999)
- Range instead of upper and lower bounds
  - Iwama and co-authors (2000)
  - Jirásková (2001)

Outline of the Talk

In this talk I will discuss two problems of state complexity:

- The first deals with languages represented by NFA's where there are restrictions on the states:
  - One initial state, and all states final;
  - One final state, and all states initial;
  - All states both initial and final.
- Joint work with Jui-Yi Kao, Andrew Malton, and Narad Rampersad

Outline of the Talk

- The second problem deals with a transformation on languages called decimation: it writes the elements of a language in radix order and then extracts a linearly-indexed subsequence.
- Joint work with Dalia Krieger, Avery Miller, Narad Rampersad, Bala Ravikumar

The First Problem: Motivation

- Andrew Malton, my colleague in computer science at Waterloo, studies the complexity of data modeling.
- He uses directed graphs to encode information and model data in software engineering and database design.
- Graphical languages used include
  - entity-relationship diagrams of Chen (1975);
  - higraphs of Harel (1988);
  - unified modeling language of Rumbaugh (2005).

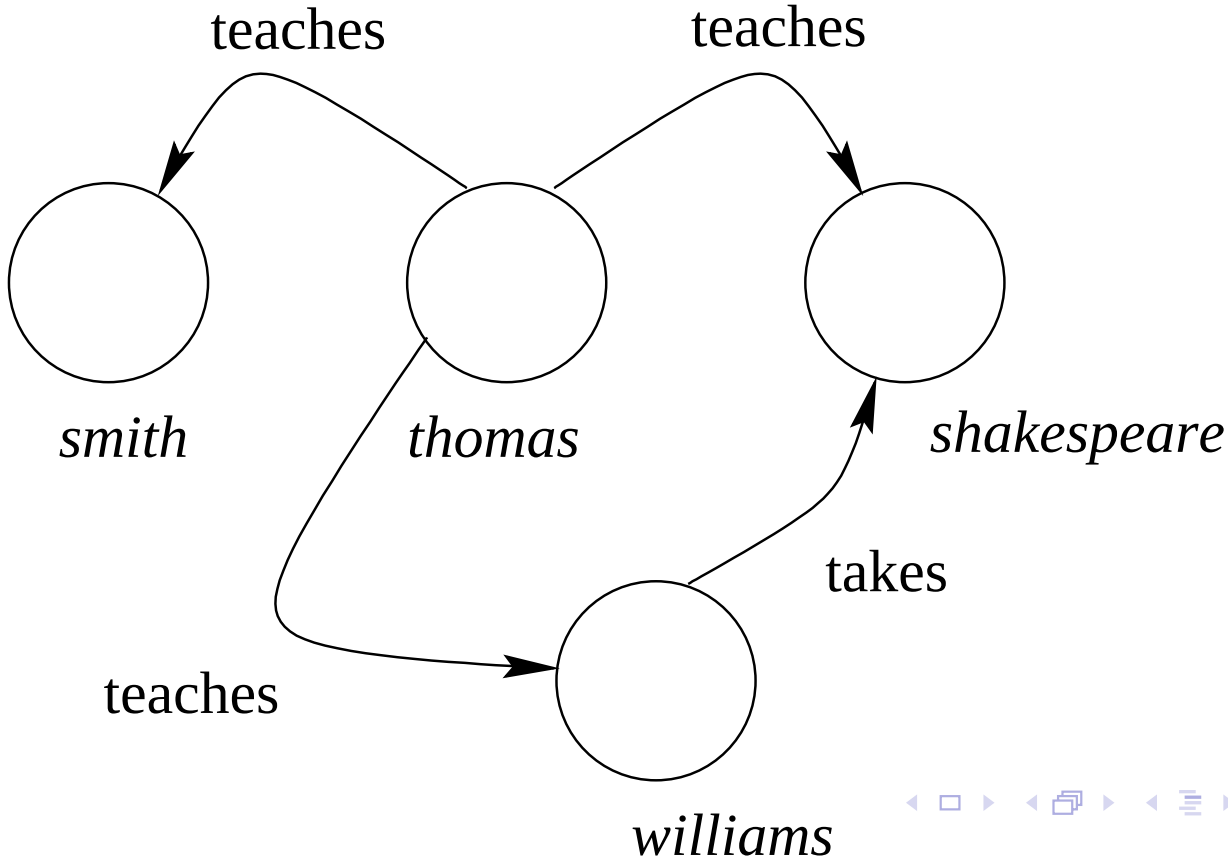
The First Problem: Motivation

These ideas are used to

- describe data structures
- model the relationships between the things the data refer to
- plan the cooperation and dependency between implementing procedures, classes, processes, packages, and so forth.

Diagrams as Directed Graphs

A diagram in such languages is a directed graph of some kind, in which the nodes stand for concepts or entities and the edges stand for directed relationships between them.



Malton was interested in the conformance problem: "are two given meta-models S and S' equivalent?" in the sense that every model described by S is described by S' and vice versa.

This amounts to determining, given two directed graphs with labels on the edges, whether the sets of labels obtained by starting and ending at any vertex coincide.

Considering these graphs as NFA's with all states both initial and final, then, the question is, are the languages accepted by these NFA's identical?

It turns out that this problem, and related ones, is PSPACE-complete.

Hardness results

First, we discuss the case of NFA's with a single initial state, and where all states are final.
Consider the following decision problem:

NFA-INEQUIVALENCE-ASF(k): Given two NFA's M_1 and M_2, over an alphabet with k letters, each having the property that all states are final states, is $L(M_1) \neq L(M_2)$?

We will show

Theorem

NFA-INEQUIVALENCE-ASF *(k) is PSPACE-complete for $k \geq 2$, but solvable in polynomial time for $k = 1$.*

$k = 1$: Let M be a unary NFA (over the alphabet $\Sigma = \{a\}$) with all states final.
Then $L(M)$ is either finite or Σ^*, depending on whether there is a cycle in the directed graph G given by the transitions of M.
If M has n states, then $a^n \in L(M)$ iff G has a cycle reachable from q_0.
Therefore, we can determine $L(M)$ efficiently by checking first if a^n is accepted.
If it isn't, we then successively check whether a^{n-1}, $a^{n-2}, \ldots, a^1, \epsilon$ are accepted.
If the first string in this list that is accepted is a^i, then $L(M) = \{\epsilon, a, \ldots, a^i\}$.
Thus we can check whether $L(M_1) \neq L(M_2)$ efficiently.

$k \geq 2$:

NFA-INEQUIVALENCE-ASF is in PSPACE:

a nondeterministic Turing machine can determine the sets of states encountered in both machines M_1 and M_2 on a guessed series of inputs.

If $L(M_1) \neq L(M_2)$, some input will lead to a set of states in one machine that contains a final state, whereas the other machine will not be in a set containing a final state.

Now we need to see that NFA-INEQUIVALENCE-ASF is PSPACE-hard.

To do so, we consider the specialization NFA-NONUNIVERSALITY-ASF:

NFA-NONUNIVERSALITY-ASF(k): Given an NFA M over an alphabet Σ with k letters, having the property that all states are final states, is $L(M) \neq \Sigma^*$?

Lemma

NFA-NONUNIVERSALITY-ASF(k) is PSPACE-hard for $k \geq 2$.

First, let's consider the case where $k \geq 3$. We reduce from the following decision problem:

NFA-NONUNIVERSALITY(k): Given an NFA M over an alphabet Σ with k letters, is $L(M) \neq \Sigma^*$?

Given an NFA M over an alphabet of size k, we transform it to an NFA M' with all states final, over an alphabet of size $k + 1$, as follows:

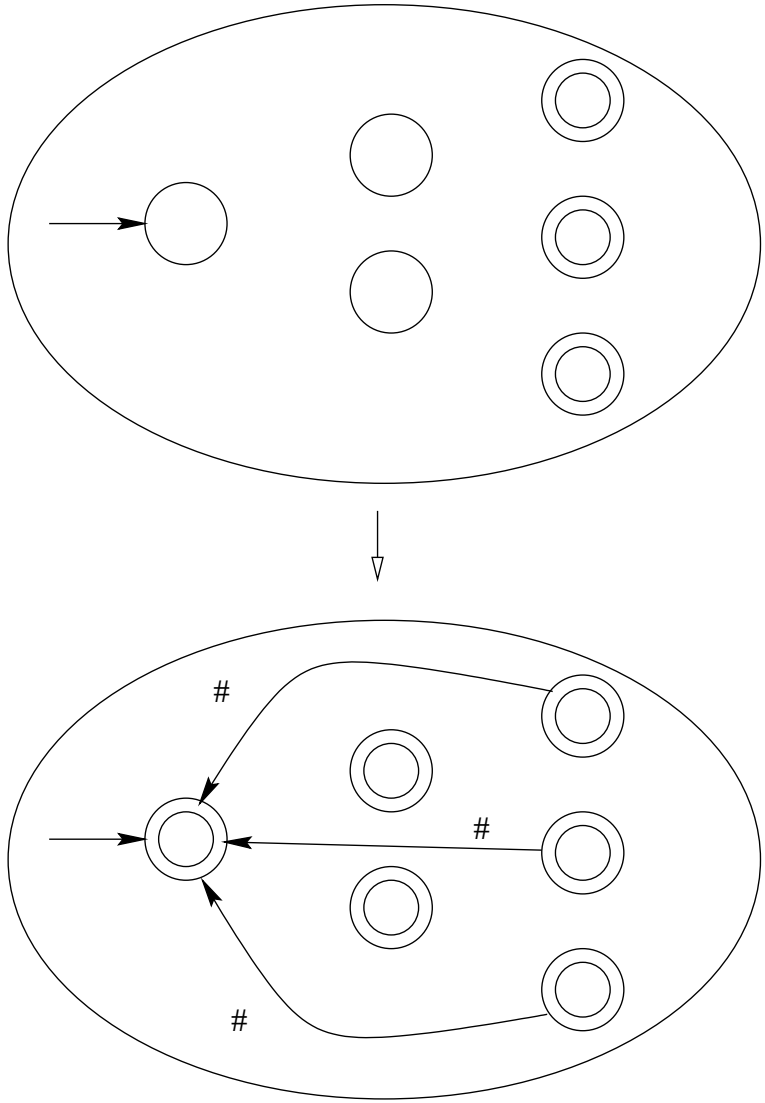


Figure: The transformation of M to M'.

Let $\Delta = \Sigma \cup \{\#\}$.
We now claim that $L(M) \neq \Sigma^*$ iff $L(M') \neq \Delta^*$, or, equivalently, $L(M) = \Sigma^*$ iff $L(M') = \Delta^*$.

Suppose $L(M) = \Sigma^*$. Then for each string $w \in \Sigma^*$, there exists a final state $p(w)$ of M such that $p \in \delta(q_0, w)$. Let $x \in \Delta^*$. If $x \in \Sigma^*$, clearly M' accepts x. Otherwise write

$$x = x_1 \# x_2 \# x_3 \# \cdots \# x_n,$$

where each $x_i \in \Sigma^*$. Now there exists an accepting computation for x in M', which starts in q_0, follows x_1 to the state $p(x_1)$, then follows the transition on $\#$ back to q_0 of M', then follows x_2 to $p(x_2)$, etc. Thus $L(M') = \Delta^*$.

Now suppose $L(M') = \Delta^*$. Then, in particular, M' accepts all strings of the form $w\#$ where $w \in \Sigma^*$. In order for M' to accept $w\#$, it must be the case that there is a transition from a state $p \in \delta(q_0, w)$ on $\#$ in M'. But then this state is final in M, by construction, so w is accepted by M. Thus $L(M) = \Sigma^*$.

This completes the reduction.

Note that our construction increases the size of the alphabet by 1, so that we have shown that

$$\texttt{NFA-NONUNIVERSALITY}(k)$$

reduces to

$$\texttt{NFA-NONUNIVERSALITY-ASF}(k+1).$$

Since NFA-NONUNIVERSALITY is PSPACE-hard for $k \geq 2$, we have proved our result for $k \geq 3$.

It remains to show NFA-NONUNIVERSALITY-ASF(2) is PSPACE-hard.
To do this, we show by recoding that NFA-NONUNIVERSALITY-ASF(4) reduces to NFA-NONUNIVERSALITY-ASF(2).
Given an machine M over the input alphabet $\Sigma = \{0, 1, 2, 3\}$ with all states final, we create a new machine M' over the input alphabet $\Delta = \{0, 1\}$.

Each transition out of a state A is recoded, and two new final states are introduced, so that

- a transition on 0 is replaced by a transition on 0 followed by 0
- a transition on 1 is replaced by a transition on 0 followed by 1
- a transition on 2 is replaced by a transition on 1 followed by 0
- a transition on 3 is replaced by a transition on 1 followed by 1

Recoding

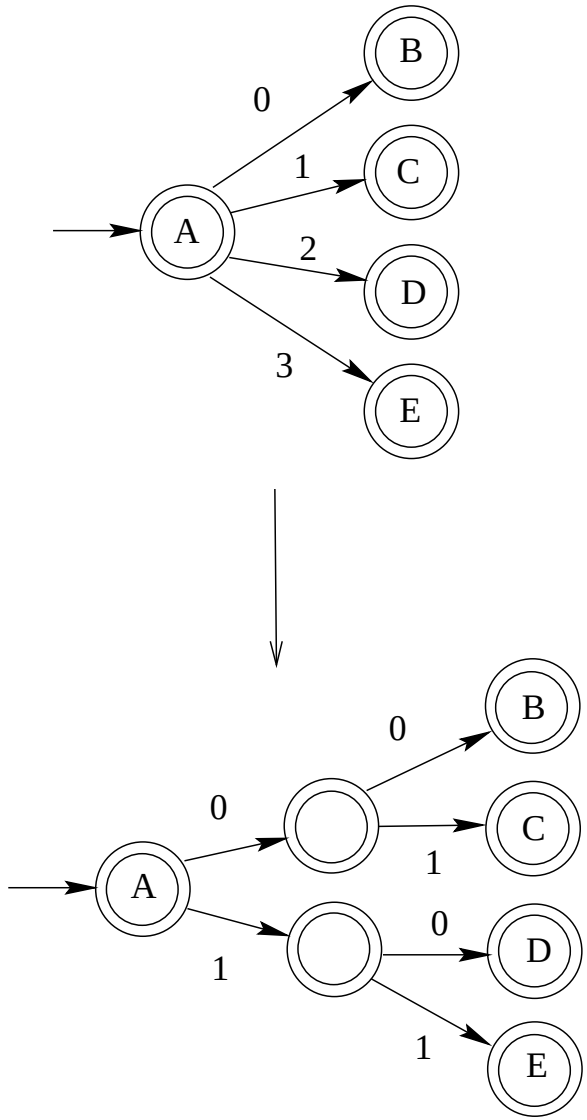


Figure: The transformation of M to M'.

All states both initial and final

Our original motivation involved generalized NFA's where all states are both initial and final.
Consider the following decision problem:

`NFA-INEQUIVALENCE-ASIF`(k): Given two NFA's M_1 and M_2, over an alphabet with k letters, each having the property that all states are both initial and final, is $L(M_1) \neq L(M_2)$?

Theorem

`NFA-INEQUIVALENCE-ASIF` *(k) is PSPACE-complete for* $k \geq 2$*, but solvable in polynomial time for* $k = 1$*.*

The idea is similar to that in the proof of Theorem 1.

Once again, we work with the "easier" problem

`NFA-NONUNIVERSALITY-ASIF`(k): Given an NFA M over an alphabet with k letters, having the property that all states are both initial and final, is $L(M) \neq \Sigma^*$?

We can show that NFA-NONUNIVERSALITY(k) reduces to NFA-NONUNIVERSALITY-ASIF($k+1$) using a simple variant of our previous proof.

Given M, an NFA over an alphabet Σ of k symbols, we modify it to obtain M', an NFA over an alphabet Δ of $k+1$ symbols, as follows:

First, we delete all states of M not reachable from q_0, the start state. Next, we introduce a new symbol $\#$ and transitions on $\#$ from each of the final states of M to q_0. Finally, we change all states to be both initial and final. We claim that $L(M) = \Sigma^*$ iff $L(M') = \Delta^*$.

The direction $L(M) = \Sigma^* \implies L(M') = \Delta^*$ is exactly as before.

For the other direction, suppose $L(M') = \Delta^*$. Then, in particular, for all $x \in \Sigma^*$, the string $\#x\#$ is accepted by M'.
Consider an accepting path for this string in M'. It starts at some state (since all states are initial) and then follows a transition on $\#$ to q_0. It now processes x and arrives at some state q. In order for it to reach a final state on the last symbol, $\#$, there must be a transition on $\#$ from q to q_0. But this can only be the case if q was final in M. Thus we have found an accepting path for x in M, and so $L(M) = \Sigma^*$.

Thus we have shown NFA-NONUNIVERSALITY-ASIF(k) is PSPACE-complete for $k \geq 3$, and thus, that NFA-INEQUIVALENCE-ASIF(k) is PSPACE-complete for $k \geq 3$.

To complete the proof of the theorem, we now show that NFA-NONUNIVERSALITY-ASF(3) reduces to NFA-NONUNIVERSALITY-ASIF(2). This proof is more complicated.

Characterization of the languages accepted by special NFA's

We define

- $\mathrm{pref}(L)$ to be the language of all prefixes of strings of L;
- $\mathrm{suff}(L)$ to be the language of all suffixes of strings of L; and
- $\mathrm{subword}(L)$ to be the language of all subwords (aka "factors") of strings of L.

A language L is

- prefix-closed if $L = \mathrm{pref}(L)$;
- suffix-closed if $L = \mathrm{suff}(L)$;
- factorial if $L = \mathrm{subword}(L)$.

Theorem

(a) *A nonempty regular language is prefix-closed if and only if it is accepted by some NFA with all states final;*

(b) *A nonempty regular language is suffix-closed if and only if it is accepted by some generalized NFA with all states initial and one final state.*

(c) *A nonempty regular language is factorial if and only if it is accepted by some generalized NFA with all states both initial and final.*

Proof. We prove only (a), as the others are similar. Suppose L is nonempty, and accepted by an NFA M with all states final. Let $x \in L$. Consider an accepting path for x. Then each state encountered on this path is final, so all prefixes of x are accepted by M.

On the other hand, suppose L is regular, nonempty, and prefix-closed.
Take any DFA M for L. Delete all states not reachable from the initial state q_0. If there are any non-final states in M, they must be "dead" states (i.e., from them, it is only possible to reach non-final states), for otherwise L would not be prefix-closed. Now delete all these non-final states and their associated transitions. Since $L \neq \emptyset$, at least one state remains. Evidently the result is an NFA that accepts L.

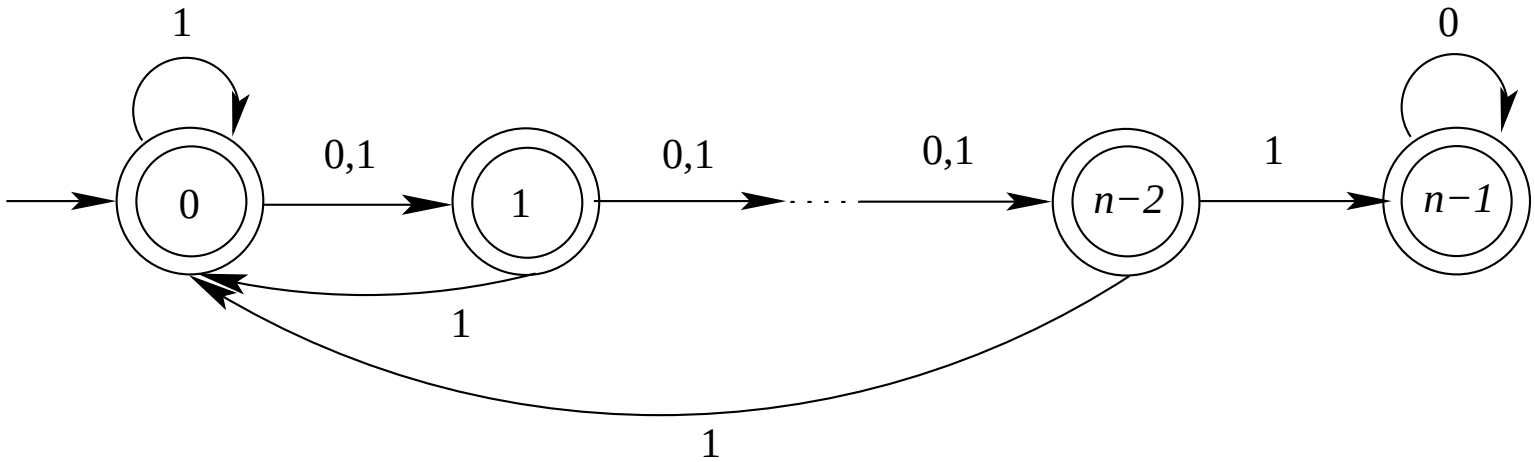
State complexity results

- Well known: for all $n \geq 1$, there exists an NFA with n states such that the minimal equivalent DFA has 2^n states.
- This blow-up can still be achieved for alphabets of size ≥ 2, if we demand that all states be
  - final
  - initial
  - both initial and final

Theorem

For $n = 1$ and every $n \geq 3$ there exists an NFA M over a binary alphabet with n states, all of which are final, such that the minimal DFA accepting $L(M)$ has 2^n states. No such binary NFA exists for $n = 2$, although over a ternary alphabet one exists.

Proof. We use an automaton like the following:

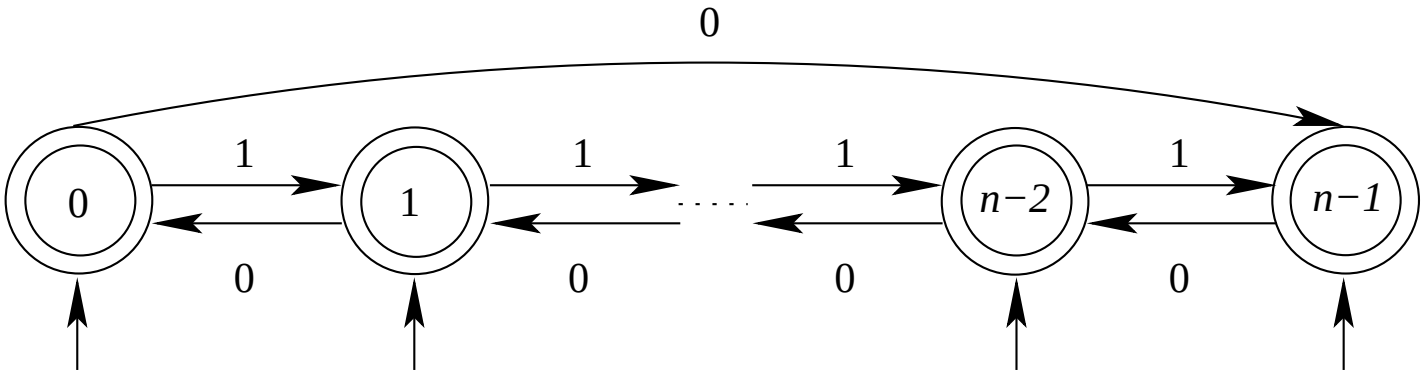


All states both initial and final

Theorem

For every $n \geq 1$ there exists an NFA M over a binary alphabet with n states, each of which is both initial and final, such that the minimal DFA accepting $L(M)$ has 2^n states.

Proof. Use an automaton like the following:

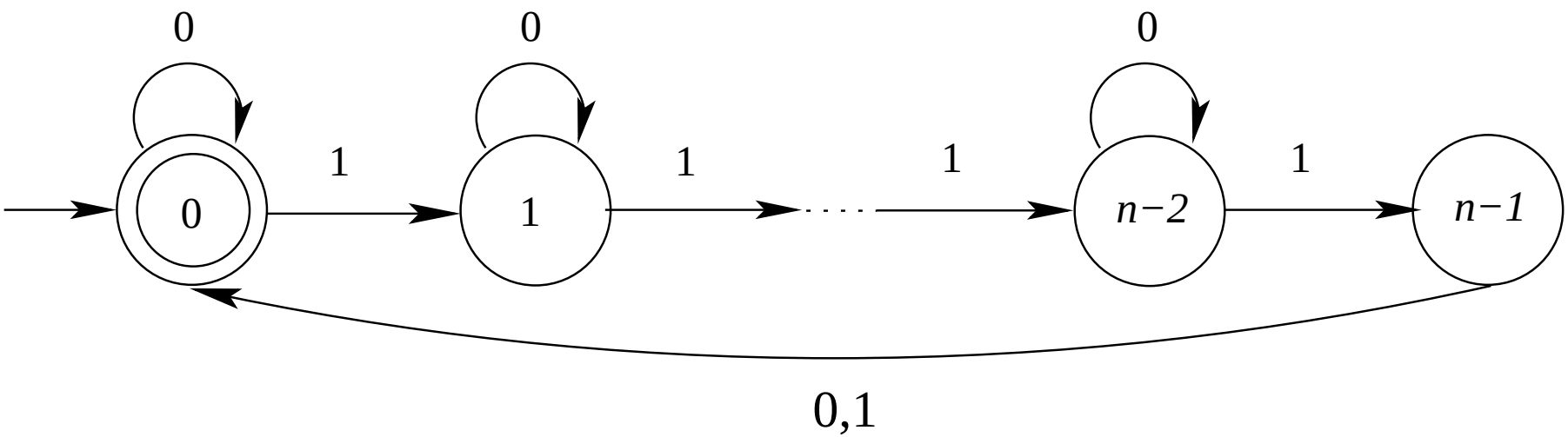


State complexity of $\mathrm{pref}(L)$, $\mathrm{suff}(L)$, $\mathrm{subword}(L)$

If the state complexity of L is n, the state complexity of $\mathrm{pref}(L)$ is also at most n, as can be seen from the standard construction for $\mathrm{pref}(L)$ where we change every state from which a final state can be reached to final.
The state complexity of $\mathrm{suff}(L)$ is more interesting.

Theorem

Let M be a DFA with n states. Then $\mathrm{suff}(L(M))$ *can be accepted by a DFA with at most* $2^n - 1$ *states, and this bound is tight.*



We now turn to the state complexity of $\mathrm{subword}(L)$:

Theorem

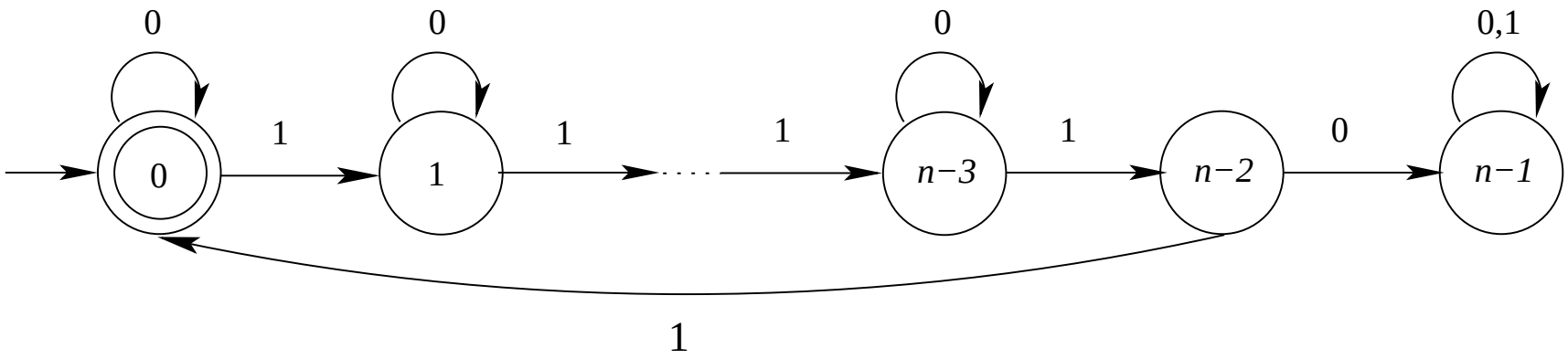
Let M be a DFA with n states. Then $\mathrm{subword}(L(M))$ can be accepted by a DFA with at most 2^{n-1} states, and this bound is tight.

Proof. Let $M = (Q, \Sigma, \delta, 0, F)$, where $Q = \{0, \ldots, n-1\}$. Let us assume that M contains no unreachable states. Suppose that every state of M can reach a final state. Then $\mathrm{subword}(L(M)) = \Sigma^*$ and is accepted by a one state DFA.

Let us suppose then that there exists $q \in Q$ such that q cannot reach a final state. Then we may remove the state q and any associated transitions to obtain a equivalent NFA with $n-1$ states. Then $\mathrm{subword}(L(M))$ is accepted by the generalized NFA $N = (Q \setminus \{q\}, \Sigma, \delta, Q \setminus \{q\}, P)$, where $P \subseteq Q$ is the set of states that can reach a final state.

The minimal DFA equivalent to N thus has at most 2^{n-1} states.

To show the bound 2^{n-1} is tight, consider the DFA M:



Abrupt Change of Direction: Decimation

- Let $k \geq 1$ and let $\Sigma = \{a_0, a_1, \ldots, a_{k-1}\}$ be a finite alphabet.
- Put an ordering on the symbols of Σ by defining $a_0 < a_1 < \cdots < a_{k-1}$.
- This ordering can be extended to the radix order on Σ^* by defining $w < x$ if
  - $|w| < |x|$, or
  - $|w| = |x|$, where $w = a_0 a_1 \cdots a_{n-1}$, $x = b_0 b_1 \cdots b_{n-1}$, and there exists an index r, $0 \leq r < n-1$ such that $a_i = b_i$ for $0 \leq i < r$ and $a_{r+1} < b_{r+1}$.

Extracting words from a language

Given a language $L = \Sigma^*$, we can consider the elements of L in radix order, say

$$L = \{w_0, w_1, w_2, \ldots, \},$$

where $w_0 < w_1 < \cdots$.
Let $I \subseteq \mathbb{N}$ be an index set. Given an infinite language L, we can extract the elements corresponding to the indices of I, and we write $L[I]$.
Let $I \subseteq \mathbb{N}$ be an index set. We say I is *ultimately periodic* if there exist integers $r \geq 0, m \geq 1$ such that for all $i \in I$ with $i \geq r$ we have
$i \in I \implies i + m \in I$.

Decimation defined

For a language L, we define the (m, r)-*decimation* $\mathrm{dec}_{m,r}(L)$ to be $L[I]$, where $I = \{im + r \ : \ i \geq 0\}$.

Two particular decimations of interest are $\mathrm{even}(L) = \mathrm{dec}_{2,0}(L)$ and $\mathrm{odd}(L) = \mathrm{dec}_{2,1}(L)$.

Lemma

Let $w \in \Sigma^*$, *where* $\Sigma = \{a_0, a_1, \ldots, a_{k-1}\}$, *and let* $F(w)$ *the set of strings that are less than* w *in radix order. Then*

$$F(wa_j) = \{\epsilon\} \cup F(w)\Sigma \cup \{w\}\{a_0, \ldots, a_{j-1}\},$$

and this union is disjoint.

Lemma

Let $L \subseteq \Sigma^*$ be a finite language, and let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Define $M(L)$ to be the matrix such that the entry in row i and column j is the number of strings $x \in L$ such that $\delta(q_i, x) = q_j$. Then

$$\begin{aligned} M(F(wa_j)) &= M(\{\epsilon\}) \\ &+ M(F(w))(M_0 + M_1 + \cdots + M_{k-1}) \\ &+ M(\{w\})(M_0 + \cdots + M_{j-1}). \end{aligned}$$

Theorem

Let $I \subseteq \mathbb{N}$ be an index set. Then $L[I]$ is regular for all regular languages L if and only if I is either finite or ultimately periodic.

Theorem

Suppose $m \geq 1$ and $0 \leq r < m$. If L is regular, accepted by an n-state DFA, then the state complexity of $\mathrm{dec}_{m,r}(L)$ is $\leq nm^n$.

Proof. Since L is regular, it is accepted by a deterministic finite automaton (DFA) $M = (Q, \Sigma, \delta, q_0, F)$, where, as usual,

- Q is a finite nonempty set of states,
- δ is the transition function,
- q_0 is the start state, and
- F is the set of final states.

We show how to construct a new DFA M' that accepts $L[I]$ where $I = \{jm + r \ : \ j \geq 0\}$.

Let $Q = \{q_0, q_1, \ldots, q_{n-1}\}$. The states of M' are pairs of the form $[\mathbf{v}, q]$, where $\mathbf{v}$ is a vector with entries in $\mathbb{Z}/(m)$ and q is a state of Q. The intent is that if we reach the state $[\mathbf{v}, q]$ by a path labeled x, then the i'th entry of $\mathbf{v}$ counts the number (modulo m) of strings $y < x$ that take M from state q_0 to q_i.

More formally, let $M' = (Q', \Sigma, \delta', q_0', F')$, where the components are defined as follows:

For $0 \leq l < k$, define M_l to be the $n \times n$ matrix where the entry in row i and column j is 1 if $\delta(q_i, a_l) = q_j$, and 0 otherwise.

Let $M = \sum_{0 \leq l < k} M_l$.

Let

$Q' = (\mathbb{Z}/(m))^n \times Q$

$q_0' = [\mathbf{0}, q_0]$ (where $\mathbf{0}$ is the vector of length n of all 0's),

$$F' = \{[\mathbf{v}, q] \ : \ \sum_{\substack{i \\ q_i \in F}} \mathbf{v}[i] \equiv r \pmod{m} \text{ and } q \in F\},$$

$$\begin{aligned}\delta'([\mathbf{v}, q_j], \quad a_i) &= \mathbf{v}M \\ &+ \quad [10\cdots0] \\ &+ \quad [\overbrace{00\cdots0}^{j-1}1\overbrace{00\cdots0}^{n-j}](M_0 + M_1 + \cdots + M_{i-1})\end{aligned}$$

where the entries in the matrix product are computed over $\mathbb{Z}/(m)$.
We claim $L(M') = \mathrm{dec}_{m,r}(L)$.

Define

$$L_n = \{x \in \{0,1\}^* \ : \ |x|_1 \equiv 0 \pmod{n}\}.$$

Then L_n can be accepted by an n-state DFA.
$\mathrm{even}(L_n)$ requires $(n+1)2^{n-1}$ states when n is odd.

Decimations of context-free languages

Ravikumar asked, if L is a context-free language (CFL), need its decimation be context-free? We give an example where this is not the case.
Let B be the balanced parentheses language on the symbols $\{a, b\}$, i.e.,

$$B = \{\epsilon, ab, aabb, abab, aaabbb, aababb, aabbab, abaabb, ababab, aaaabbbb, \ldots\}.$$

This is a well-known CFL, generated by the context-free grammar

$$S \rightarrow aSbS \mid \epsilon$$

We will show that

$$\text{even}(B) = \{\epsilon, aabb, aaabbb, aabbab, ababab, \ldots\}$$

is not a CFL.

The proof is based on the following claims:

Lemma

The number of strings of length $2n$ in B is the Catalan number $C_n = \binom{2n}{n}/(n+1)$.

Now let $\nu_2(n)$ denote the exponent of the highest power of 2 dividing n.

Let $s_2(n)$ denote the number of 1's in the binary expansion of n.

Lemma

For $n \geq 0$ we have $\nu_2(n!) = n - s_2(n)$.

Lemma

For $n \geq 0$, C_n is odd if and only if $n = 2^i - 1$ for some integer $i \geq 0$.

Proof. We have

$$\begin{aligned}
\nu_2(C_n) &= \nu_2\left(\frac{\binom{2n}{n}}{n+1}\right) \\
&= \nu_2((2n)!) - 2\nu_2(n!) - \nu_2(n+1) \\
&= (2n - s_2(2n)) - 2(n - s_2(n)) - \nu_2(n+1) \\
&= s_2(n) - \nu_2(n+1).
\end{aligned}$$

Thus C_n is odd if and only if $s_2(n) = \nu_2(n+1)$, if and only if $n = 2^i - 1$ for some $i \geq 0$.

Lemma

For $n \geq 0$ define $D_n = \sum_{1 \leq i \leq n} C_n$. (Thus $D_0 = 0$.) Then D_n is even if and only if there exists $i \geq 0$ such that $2^{2i} - 1 \leq n < 2^{2i+1} - 1$.

We are now ready to prove

Theorem

The language $\mathrm{even}(B)$ *is not a context-free language.*

Proof. First, we observe that $(ab)^n$ is the lexicographically greatest string of length $2n$ in B.

It follows that $(ab)^n$ is, in the radix order, the $D_n = (\sum_{1 \leq i \leq n} C_i)$'th string in B. (Recall that we start indexing at 0.)

Now define the morphism $h : \{c\}^* \longrightarrow \{a, b\}^*$ by $h(c) = ab$.

By a well-known theorem $h^{-1}(B)$ is a context-free language.
But $h^{-1}(B) = \{c^n \ : \ D_n \text{ is even}\}$.

From a previous lemma, we have

$$h^{-1}(B) = \{c^n \ : \ \exists i \geq 0 \text{ such that } 2^{2i} - 1 \leq n < 2^{2i+1} - 1\}$$

Since $h^{-1}(B)$ is a unary CFL, by a well-known theorem it is actually regular.

But now picking the string $z = c^{2^{2i}-1}$ and applying the pumping lemma, we get a contradiction.