Universality and Prefixes, Suffixes, and Factors

J. Shallit[1] N. Rampersad[2] Z. Xu[3]

[1]University of Waterloo, Waterloo, Ontario, Canada

[2]University of Winnipeg, Winnipeg, Manitoba, Canada

[3]University of Western Ontario, London, Ontario, Canada

WORDS 2009, Salerno

Universality problems

A *universality problem* consists of the following: given a subset S of some universal set U, represented in some fashion, determine if $S = U$.

- Long studied in computer science.

- Often harder than emptiness problems.

- Example: the decision problem,

  Given a context-free grammar G, is $L(G) = \Sigma^*$?

is recursively unsolvable (Bar-Hillel, Perles, Shamir, 1961).

More universality problems

Meyer and Stockmeyer proved the decision problem

> *Given an NFA M over the alphabet Σ, is $L(M) = \Sigma^*$?*

is PSPACE-complete.

Their proof: based on constructing the set of all words which do not represent an accepting computation.

More universality problems

Kozen proved the decision problem

Given n DFA's M_1, M_2, ..., M_n, each with input alphabet Σ, is $\bigcup_{1 \leq i \leq n} L(M_i) = \Sigma^*$?

is PSPACE-complete.

Outline of the talk

In this talk I will discuss

1. Some new universality problems;

Outline of the talk

In this talk I will discuss

1. Some new universality problems;
2. Connections with Černý's celebrated conjecture on synchronizing words;

Outline of the talk

In this talk I will discuss

1. Some new universality problems;
2. Connections with Černý's celebrated conjecture on synchronizing words;
3. Some hardness results;

Outline of the talk

In this talk I will discuss

1. Some new universality problems;
2. Connections with Černý's celebrated conjecture on synchronizing words;
3. Some hardness results;
4. Bounds on the length of the shortest "counterexample" to the universality property.

Infinite Words

(See the splendid book of Perrin and Pin, *Infinite Words*.)

We write Σ^ω for the set of right-infinite words over Σ, that is, words of the form $a_0 a_1 a_2 \cdots$.

We write ${}^\omega\Sigma$ for the set of left-infinite words over Σ, that is, words of the form $\cdots a_{-2} a_{-1} a_0$.

We write ${}^\omega\Sigma^\omega$ for the set of bi-infinite words over Σ, that is, words of the form $\cdots a_{-2} a_{-1} a_0 a_1 a_2 \cdots$.

Omega-powers

L^ω denotes the set of all right-infinite words formed from concatenations of words from L.

${}^\omega L$ denotes the set of all left-infinite words formed from concatenations of words from L.

${}^\omega L^\omega$ denotes the set of all bi-infinite words formed from concatenations of words from L.

Universality for infinite words

Let S be a finite set of finite words. Under what conditions do the following hold?

- $S^* = \Sigma^*$? — not interesting, since this holds iff $\Sigma \subseteq S$.

Universality for infinite words

Let S be a finite set of finite words. Under what conditions do the following hold?

- $S^* = \Sigma^*$? — not interesting, since this holds iff $\Sigma \subseteq S$.
- $S^\omega = \Sigma^\omega$?

Universality for infinite words

Let S be a finite set of finite words. Under what conditions do the following hold?

- $S^* = \Sigma^*$? — not interesting, since this holds iff $\Sigma \subseteq S$.
- $S^\omega = \Sigma^\omega$?
- ${}^\omega S = {}^\omega \Sigma$?

Universality for infinite words

Let S be a finite set of finite words. Under what conditions do the following hold?

- $S^* = \Sigma^*$? — not interesting, since this holds iff $\Sigma \subseteq S$.
- $S^\omega = \Sigma^\omega$?
- ${}^\omega S = {}^\omega \Sigma$?
- ${}^\omega S^\omega = {}^\omega \Sigma^\omega$?

$S^\omega = \Sigma^\omega$

Claim. Let S be a finite set of finite words. Then $S^\omega = \Sigma^\omega$ iff $\mathrm{Pref}(S^*) = \Sigma^*$.

Proof. Suppose $S^\omega = \Sigma^\omega$. Then for all $x \in \Sigma^*$, there exists a sequence s_i of elements of S such that

$$s_1 s_2 s_3 \cdots = x\ 0^\omega.$$

Since x is finite, it follows that x is a prefix of $s_1 s_2 \cdots s_i$ for some i. So $\mathrm{Pref}(S^*) = \Sigma^*$.

The other direction

$\mathrm{Pref}(S^*) = \Sigma^* \implies S^\omega = \Sigma^\omega$: use an argument based on König's infinity lemma. This lemma states that every tree with infinitely many vertices, each of finite degree, has at least one infinite simple path.

Suppose $\mathrm{Pref}(S^*) = \Sigma^*$.

Let $x = a_1 a_2 a_3 \cdots$ be any word in Σ^ω. Consider the sequence of prefixes of x: $a_1, a_1 a_2, a_1 a_2 a_3, \ldots$. Each prefix is the prefix of some element of S^*. Make an infinite tree out of all the factorizations of S^* thus produced, connecting two factorizations if one is a prefix of the other and no other factorizations come between them in this ordering. Since S is finite, each vertex is of finite degree, so there is an infinite path, which corresponds to a factorization of $\mathbf{x}$. ■

More on infinite words

Note that in the statement

$$S^\omega = \Sigma^\omega \quad \textit{iff} \quad \mathrm{Pref}(S^*) = \Sigma^*$$

the requirement that S be finite is essential.

For example, if

$S = \{0, 10, 110, 1110, \ldots\}$,

then $\mathrm{Pref}(S^*) = \Sigma^*$, but $S^\omega \neq \Sigma^\omega$ (since S^ω omits the word $111\cdots$).

${}^\omega S = {}^\omega \Sigma$

A similar argument works to prove

${}^\omega S = {}^\omega \Sigma$ iff $\mathrm{Suff}(S^*) = \Sigma^*$

and

${}^\omega S^\omega = {}^\omega \Sigma^\omega$ iff $\mathrm{Fact}(S^*) = \Sigma^*$.

Deciding if $\mathrm{Pref}(S^*) = \Sigma^*$

Given a finite list of words S, how quickly can we check if $\mathrm{Pref}(S^*) = \Sigma^*$?

Claim: the following algorithm works in linear time:

- Create a trie out of the words of S, discarding any word for which there is a proper prefix already in S.

Deciding if $\mathrm{Pref}(S^*) = \Sigma^*$

Given a finite list of words S, how quickly can we check if $\mathrm{Pref}(S^*) = \Sigma^*$?

Claim: the following algorithm works in linear time:

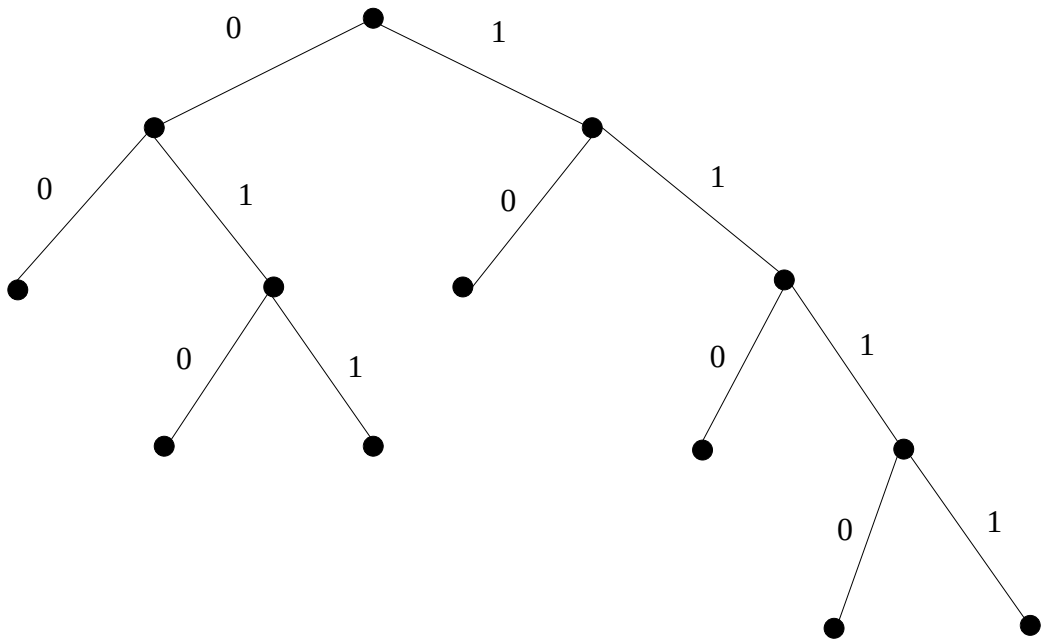
- Create a trie out of the words of S, discarding any word for which there is a proper prefix already in S.
- If every node in the resulting tree has 0 or $|\Sigma|$ children, answer "yes", otherwise answer "no".

An example of the algorithm

Suppose

$$S = \{00, 10, 010, 011, 110, 1110, 1111\}.$$

When we make a trie we discover each node has degree 0 or 2.
Thus $\mathrm{Pref}(S^*) = \{0, 1\}^*$.



Deciding if $\mathrm{Suff}(S^*) = \Sigma^*$

Similarly, we can decide if $\mathrm{Suff}(S^*) = \Sigma^*$ in linear time by applying the trie algorithm to the reversal of all words of S.

Deciding if $\mathrm{Fact}(S^*) = \Sigma^*$

I don't know the complexity of this problem.

Note that it is not true that

$\mathrm{Fact}(S^*) = \Sigma^*$ implies that one of $\mathrm{Pref}(S^*) = \Sigma^*$, $\mathrm{Suff}(S^*) = \Sigma^*$ holds.

A counterexample is $S = \{0, 01, 10, 111, 1011\}$.

Then it is easy to check that

$\mathrm{Fact}(S^*) = \Sigma^*$.

But neither $\mathrm{Pref}(S^*) = \Sigma^*$ nor $\mathrm{Suff}(S^*) = \Sigma^*$ hold.

For example, $110 \notin \mathrm{Pref}(S^*)$ and $0011 \notin \mathrm{Suff}(S^*)$.

A Variation on the Problem

Instead of starting with a finite set of words, let's start with a regular language specified by a DFA M.

We get three decision problems:

- Given a DFA M with input alphabet Σ, is $\mathrm{Pref}(L(M)) = \Sigma^*$?

A Variation on the Problem

Instead of starting with a finite set of words, let's start with a regular language specified by a DFA M.

We get three decision problems:

- Given a DFA M with input alphabet Σ, is $\mathrm{Pref}(L(M)) = \Sigma^*$?
- Given a DFA M with input alphabet Σ, is $\mathrm{Suff}(L(M)) = \Sigma^*$?

A Variation on the Problem

Instead of starting with a finite set of words, let's start with a regular language specified by a DFA M.

We get three decision problems:

- Given a DFA M with input alphabet Σ, is $\mathrm{Pref}(L(M)) = \Sigma^*$?
- Given a DFA M with input alphabet Σ, is $\mathrm{Suff}(L(M)) = \Sigma^*$?
- Given a DFA M with input alphabet Σ, is $\mathrm{Fact}(L(M)) = \Sigma^*$?

Deciding if $\mathrm{Pref}(L(M)) = \Sigma^*$

Easy - because we can create a DFA for $\mathrm{Pref}(L(M))$ by changing the set of final states to those states q from which there exists a path from q to a formerly final state.

The set of these states q can be determined through depth-first search in the graph formed by turning all the transitions around.

So this question can be settled in linear time.

Deciding if $\mathrm{Suff}(L(M)) = \Sigma^*$

Hard! It turns out to be PSPACE-complete.

The decision problem is in PSPACE because the more general problem of testing universality for NFA's is in PSPACE.

Testing universality for NFA's is in PSPACE

Given an NFA M with n states, we can test if $L(M) \neq \Sigma^*$ as follows:

We "guess" a word w that is not accepted, and we simulate M on w. This can be done in nondeterministic polynomial space provided w is not too long.

To see that w is not too long, note that we can convert M to a DFA M' with at most 2^n states with the subset construction, and then to a DFA M'' accepting $\overline{L(M)}$ by changing final states to non-final, and vice versa. If $L(M'') \neq \emptyset$, then M'' must accept a word of length at most 2^n.

So in our test, $|w| < 2^n$. Since we can count up to 2^n in polynomial space, we have a NPSPACE algorithm. But NPSPACE = PSPACE by Savitch's theorem.

Deciding if $\mathrm{Suff}(L(M)) = \Sigma^*$ is in PSPACE

To complete the proof that deciding $\mathrm{Suff}(L(M)) = \Sigma^*$ is in PSPACE, we need to see how to take a DFA for M and convert it to an NFA for $\mathrm{Suff}(L(M))$.

To do this, we take M and change q_0, the initial state, into a set of initial states S. A state q is in S iff there is a path from q_0 to q. This creates a "generalized NFA" with a set of initial states instead of a single state. We can easily convert this to an ordinary NFA by adding a new initial state q_0' and ϵ-transitions to states of S, then using the usual algorithm to replace ϵ-transitions with ordinary ones.

Deciding if $\mathrm{Suff}(L(M)) = \Sigma^*$ is PSPACE-hard

Reduce from the following well-known PSPACE-complete problem:

Given n DFA's $M_0, M_1, \ldots, M_{n-1}$, is there a word accepted by all of them?

More precisely, we reduce from the following problem: given n DFA's $M_0, M_1, \ldots, M_{n-1}$, is the union of all their languages equal to Σ^*?

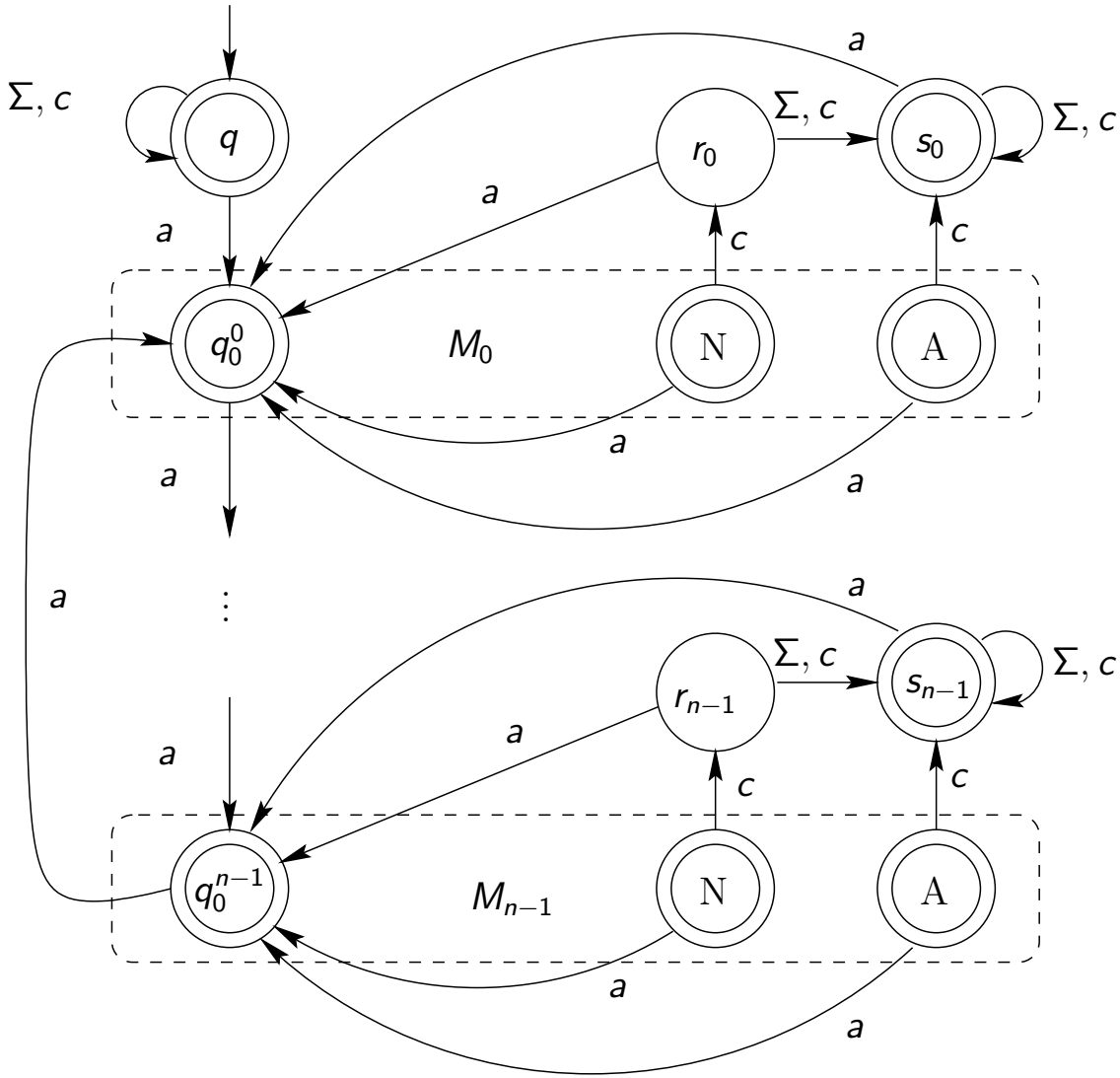
Deciding if $\mathrm{Suff}(L(M)) = \Sigma^*$ is PSPACE-hard

Suppose $M_i = (Q_i, \Sigma, \delta_i, q_0^i, F_i)$ for $0 \leq i < n$.

Without loss of generality, we assume no M_i has transitions into the initial state; if this condition does not hold, we alter M_i to add a new initial state and transitions out of this initial state that coincide with the original initial state.

Let a, c be letters not in Σ, and let $\Delta = \Sigma \cup \{a, c\}$. We create a new DFA $M = (Q, \Delta, \delta, q, F)$ as in the next slide.

Deciding if $\mathrm{Suff}(L(M)) = \Sigma^*$ is PSPACE-hard



Universality for factors

Theorem.

The decision problem

Given a DFA M with input alphabet Σ, is $\mathrm{Fact}(L(M)) = \Sigma^*$?

is solvable in polynomial-time.

Terminology and observations about DFA's

A state q is *dead* if no accepting state can be reached from q via any path (including the empty path).

If a DFA has a dead state d, then every state reachable from it is also dead.

So there is an equivalent DFA with only one dead state and all transitions from that dead state lead to itself.

Terminology and observations about DFA's

A state r is *universal* if no dead state is reachable from it via a possibly empty path.

A state is *reachable* if there is some path to it from the start state.

A DFA is *initially connected* if all states are reachable.

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ has a *synchronizing word* w if $\delta(p, w) = \delta(q, w)$ for all states p, q.

Synchonizing words

A huge literature exists on this topic, much of it dealing with *Černý's conjecture*: if a DFA has a synchronizing word, it has such a word of length at most $(n-1)^2$.

The best upper bound known so far is $(n^3 - n)/6$ (Pin).

For a good survey, see Volkov's paper in the LATA 2008 proceedings.

Testing if a DFA has a synchronizing word

Proposition. (Černý, 1964). A DFA $M = (Q, \Sigma, \delta, q_0, F)$ has a synchronizing word if and only if for all $q, q' \in Q$ there exists a word $w \in \Sigma^*$ such that $\delta(q, w) = \delta(q', w)$.

Now build a new DFA M' based on M whose states are subsets of Q of cardinality at most 2, and two such subsets S, S' are connected with a directed arrow labeled a if $\delta(S, a) = S'$.

Then M has a synchronizing word if and only if every state of cardinality 2 in M' has a path to some state of cardinality 1.

This can be settled using depth-first search in quadratic time.

Universal states

Lemma. If a DFA M has a reachable universal state, then $\mathrm{Fact}(L(M)) = \Sigma^*$.

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$.

Let q be a reachable universal state, and let x be such that $\delta(q_0, x) = q$.

Consider any word y, and let $\delta(q, y) = r$.

Then no dead state is reachable from r, for otherwise it would be reachable from q.

So there exists a word z such that $\delta(r, z) = s$, and s is an accepting state.

Then $\delta(q_0, xyz) = s$, so xyz is accepted, and hence $y \in \mathrm{Fact}(L(M))$. But y was arbitrary, so $\mathrm{Fact}(L(M)) = \Sigma^*$. ■

Synchronizing words

Lemma. Suppose the DFA M is initially connected, has no universal states, and has exactly one dead state. Then there exists $x \notin \mathrm{Fact}(L(M))$ if and only if there is a synchronizing word for M.

Proof. Suppose M has a synchronizing word x.

Then there exists a state q such that for all all states p we have $\delta(p, x) = q$.

Since, as noted above, all transitions from the unique dead state d must go to itself, we must have $q = d$.

Then for all states p we have $\delta(p, x) = d$.

So $x \notin \mathrm{Fact}(L(M))$, because every path labeled x goes to a state from which one cannot reach a final state.

Proving the other direction

Now suppose there is $x \notin \mathrm{Fact}(L(M))$.

Then for all y, z we have $yxz \notin L(M)$.

In other words, no matter what state we start in, xz leads to a nonaccepting state.

Then no matter what state we start in, x leads to a state from which no accepting state can be reached.

But there is only one such state, the dead state d.

So it must be the case that x always leads to d, and so x is a synchronizing word. ■

This result was also found independently by Elena Pribavkina.

Testing if $\mathrm{Fact}(L(M)) = \Sigma^*$ in polynomial time

The following algorithm decides whether $\mathrm{Fact}(L(M)) = \Sigma^*$ in polynomial time:

1. Remove all states not reachable from the start state by a (possibly empty) directed path.

Testing if $\mathrm{Fact}(L(M)) = \Sigma^*$ in polynomial time

The following algorithm decides whether $\mathrm{Fact}(L(M)) = \Sigma^*$ in polynomial time:

1. Remove all states not reachable from the start state by a (possibly empty) directed path.
2. Identify all dead states via depth-first search. If M has at least one dead state, modify M to replace all dead states with a single dead state d.

Testing if $\mathrm{Fact}(L(M)) = \Sigma^*$ in polynomial time

The following algorithm decides whether $\mathrm{Fact}(L(M)) = \Sigma^*$ in polynomial time:

1. Remove all states not reachable from the start state by a (possibly empty) directed path.
2. Identify all dead states via depth-first search. If M has at least one dead state, modify M to replace all dead states with a single dead state d.
3. Identify all universal states via depth-first search. If there is a universal state, answer "Yes" and halt.

Testing if $\mathrm{Fact}(L(M)) = \Sigma^*$ in polynomial time

The following algorithm decides whether $\mathrm{Fact}(L(M)) = \Sigma^*$ in polynomial time:

1. Remove all states not reachable from the start state by a (possibly empty) directed path.
2. Identify all dead states via depth-first search. If M has at least one dead state, modify M to replace all dead states with a single dead state d.
3. Identify all universal states via depth-first search. If there is a universal state, answer "Yes" and halt.
4. Using the quadratic procedure mentioned previously decide if M has a synchronizing word. If it does, answer "No"; otherwise answer "Yes".

Why the algorithm works

We already observed that we can replace all dead states by a single dead state without changing the language accepted by M.

If a DFA has no universal states, then it has at least one dead state (for otherwise every state would be universal).

So when we reach step 4 of the algorithm, we are guaranteed that M has exactly one dead state, and we can apply our previous lemma.

Universality for NFA's

We consider some universality problems again, but now we represent our regular language by an NFA.

Theorem.
The decision problem

Given an NFA M with input alphabet Σ, is $\mathrm{Pref}(L(M)) = \Sigma^*$?

is PSPACE-complete.

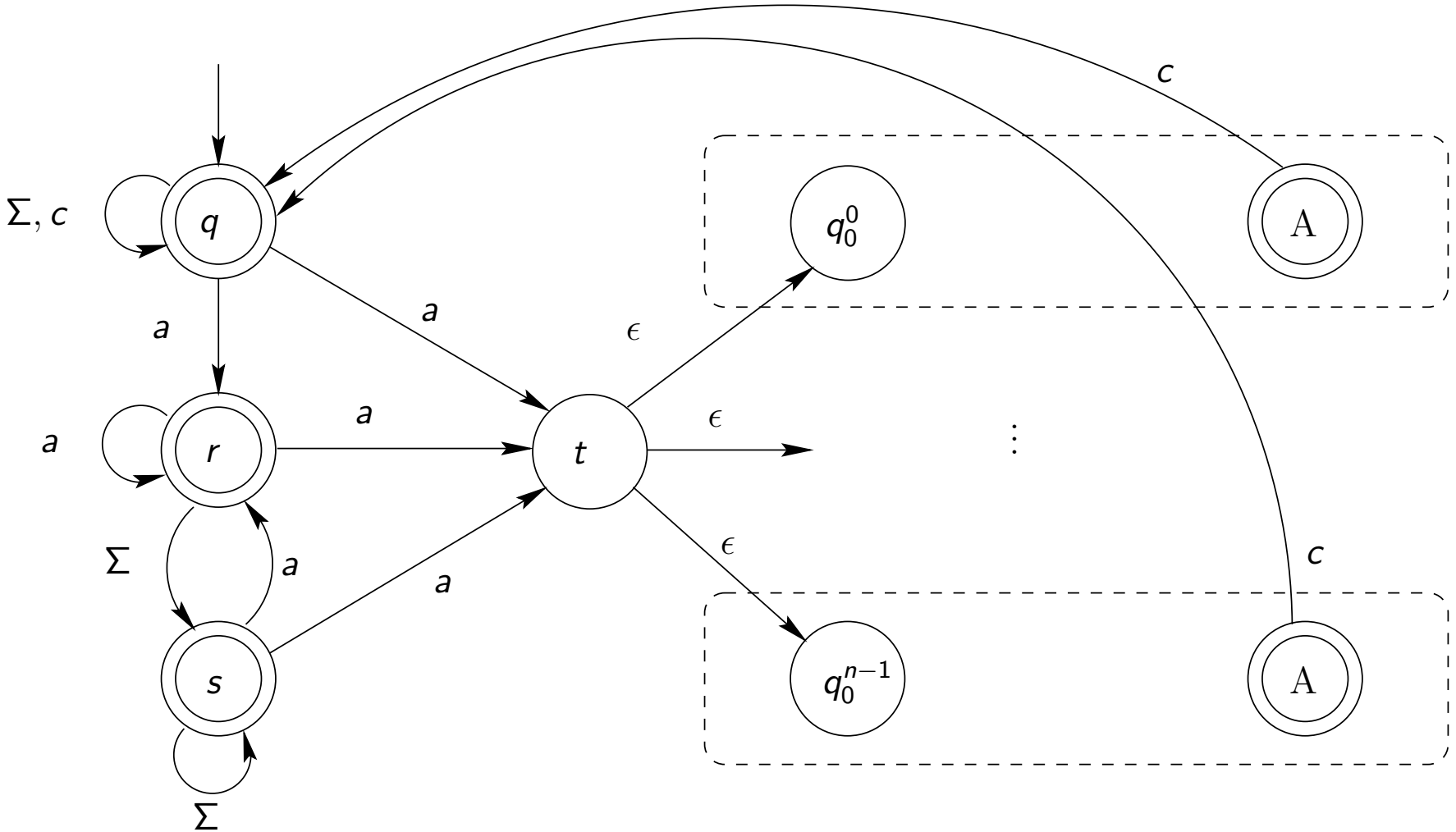
Proof. In fact, this decision problem is even PSPACE-complete when M is restricted to be of the form A^R, where A is a DFA.
To see this, note that our construction for suffix universality for DFA's given above, when reversed, gives an NFA M with the property that $\mathrm{Pref}(L(M)) = \Sigma^*$ if and only if $\bigcup_{0 \leq i < n} L(M_i) = \Sigma^*$. ■

Universality for factors

Although, as we have seen, universality for $\mathrm{Fact}(L(M))$ is testable in polynomial-time when M is a DFA, the same decision problem becomes PSPACE-complete when M is an NFA.

To see this, we again reduce from the universality problem for n DFA's.

Universality for factors



An application to boolean matrices

Our result about factors has an interesting interpretation in terms of Boolean matrices.

Given an NFA $M = (Q, \Sigma, \delta, q_0, F)$, construct $|\Sigma|$ different matrices M_a, for each $a \in \Sigma$, as follows: M_a has a 1 in row i and column j if $q_j \in \delta(q_i, a)$, and a 0 otherwise.

Then for all words $w = c_1 c_2 \cdots c_k$,

$$M_w := M_{c_1} M_{c_2} \cdots M_{c_k}$$

has a 1 in row i and column j iff $q_j \in \delta(q_i, w)$.

An application to boolean matrices

Assume that M is an NFA in which every state is reachable from the start state and that a final state can be reached from every state. (If M does not fulfill these conditions, we delete the appropriate states.)

Then form M_a for each $a \in \Sigma$.

We claim that some product of the M_a equals the all-zeros matrix iff $\mathrm{Fact}(L(M)) \neq \Sigma^*$.

An application to boolean matrices

Suppose there is some product, say M_y for $y = c_1 \cdots c_k$, that equals the all-zeros matrix.

Then no matter what state we start in, reading y takes us to no state, so xyz is rejected for all x, z.

Hence $y \notin \mathrm{Fact}(L(M))$.

An application to boolean matrices

On the other hand, if $\mathrm{Fact}(L(M)) \neq \Sigma^*$, then there must be some $y \notin \mathrm{Fact}(L(M))$.

We claim M_y is the all-zeros matrix.

If not, there exist i, j such that M_y has a 1 in row i and column j.

Then since every state is reachable from the start state, there exists x such that $\delta(q_0, x) = q_i$.

Since a final state can be reached from every state, there exists z such that $\delta(q_j, z) \in F$.

Then $\delta(q_0, xyz) \in F$, so M accepts xyz and $y \in \mathrm{Fact}(L(M))$, contradicting our assumption.

An application to boolean matrices

We have therefore shown

Corollary. The decision problem

Given a finite list of square Boolean matrices of the same dimension, is some product equal to the all-zeros matrix?

is PSPACE-complete.

Universality and subwords

We say a word x is a *subword* of a word w if we can obtain x by deleting one or more symbols from w.

Let's look at the problem of determining, given an NFA M, whether $\mathrm{Subw}(L(M)) = \Sigma^*$.

Lemma. Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA such that

(a) every state is reachable from q_0 and

Then $\mathrm{Subw}(L(M)) = \Sigma^*$ if and only if the transition diagram of M has a strongly connected component C such that, for each letter $a \in \Sigma$, there are two states of C connected by an edge labeled a.

Universality and subwords

We say a word x is a *subword* of a word w if we can obtain x by deleting one or more symbols from w.

Let's look at the problem of determining, given an NFA M, whether $\mathrm{Subw}(L(M)) = \Sigma^*$.

Lemma. Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA such that

(a) every state is reachable from q_0 and

(b) a final state is reachable from every state.

Then $\mathrm{Subw}(L(M)) = \Sigma^*$ if and only if the transition diagram of M has a strongly connected component C such that, for each letter $a \in \Sigma$, there are two states of C connected by an edge labeled a.

Proof of the result for subwords

Proof. Suppose the transition diagram of M has a reachable strongly connected component C with the given property.

Then to obtain any word w as a subword of a word in $L(M)$, use a word to enter the strongly connected component C, and then travel successively to states of C where there is an arrow out labeled with each successive letter of w.

Finally, travel to a final state.

Proof of the converse result for subwords

For the converse, assume $\mathrm{Subw}(L(M)) = \Sigma^*$, but the transition diagram of M has no strongly connected component with the given property.

Then since any directed graph can be decomposed into a directed acyclic graph on its strongly connected components, we can write any $w \in L(M)$ as $x_1 y_1 x_2 y_2 \cdots x_n$, where x_i is the word traversed inside a strongly connected component, and y_i is the word on an edge linking two strongly connected components.

Furthermore, $n \leq N$, where N is the total number of strongly connected components.

Proof of the converse result for subwords

If $\Sigma = \{a_1, a_2, \ldots, a_k\}$, then $\mathrm{Subw}(L(M))$ omits the word $w = (a_1 a_2 \cdots a_k)^{N+1}$, because the first component encountered has no transition on some letter a_i, so reading $a_1 a_2 \cdots a_k$ either forces a transition to (at least) the next component of the DAG, or in the case of an NFA, ends the computational path with no move.

Since there are only N strongly connected components, we cannot have w as a subword of any accepted word. ■

An algorithm for subwords

We can now prove

Theorem. Given an NFA M with input alphabet Σ, we can determine if $\mathrm{Subw}(L(M)) = \Sigma^*$ in linear time.

Proof. First, use depth-first search to remove all states not reachable from the start state.

Next, use depth-first search (on the transition diagram of M with arrows reversed) to remove all states from which one cannot reach a final state.

Next, determine the strongly connected components of the transition diagram of M (which can be done in linear time).

Finally, examine all the edges of each strongly connected component C to see if for all $a \in \Sigma$, there is an edge labeled a.

Shortest counterexamples

If a language L is not equal to the universal U, we might want to find good bounds on the length of the shortest *counterexample*, that is, a word in $U - L$.

Given that $\overline{\mathrm{Pref}(L(M))} \neq \Sigma^*$, what is the length of the shortest word in $\overline{\mathrm{Pref}(L(M))}$, as a function of the number of states of M?

We can ask the same question for suffixes, factors, and subwords.

Shortest counterexamples for prefixes

Theorem. Let M be a DFA or NFA with n states. Suppose $\mathrm{Pref}(L(M)) \neq \Sigma^*$. Then the shortest word in $\overline{\mathrm{Pref}(L(M))}$ is

(a) of length $\leq n-1$ if M is a DFA, and there exist examples achieving $n-1$;

Shortest counterexamples for prefixes

Theorem. Let M be a DFA or NFA with n states. Suppose $\mathrm{Pref}(L(M)) \neq \Sigma^*$. Then the shortest word in $\overline{\mathrm{Pref}(L(M))}$ is

(a) of length $\leq n-1$ if M is a DFA, and there exist examples achieving $n-1$;

(b) of length $\leq 2^n$ if M is an NFA, and there exist examples achieving 2^{cn} for some constant c.

Proof

(a) If M is a DFA with n states, our construction shows $\overline{\mathrm{Pref}(L(M))}$ can be accepted by a DFA M' with n states. If M' accepts a word, it accepts one of length $\leq n-1$.

An example achieving this bound is $L = a^{n-2}$, which can be accepted by an n-state DFA, and the shortest word not in $\mathrm{Pref}(L)$ is a^{n-1}.

Proof

(a) If M is a DFA with n states, our construction shows $\overline{\mathrm{Pref}(L(M))}$ can be accepted by a DFA M' with n states. If M' accepts a word, it accepts one of length $\leq n-1$.

An example achieving this bound is $L = a^{n-2}$, which can be accepted by an n-state DFA, and the shortest word not in $\mathrm{Pref}(L)$ is a^{n-1}.

(b) The upper bound is trivial (convert the NFA for M to one for $\mathrm{Pref}(L(M))$; then convert the NFA to a DFA and change accepting states to non-accepting and vice versa; such a DFA has at most 2^n states).

The examples achieving 2^{cn} for some constant c can be constructed using an n-state NFA M with all states final such that the shortest word not accepted is of length 2^{cn}. However, if all states are final, then $\mathrm{Pref}(L(M)) = L(M)$, so this construction provides the needed example. ■

Shortest counterexamples for suffixes

Theorem. Let M be a DFA or NFA with n states. Suppose $\overline{\mathrm{Suff}(L(M))} \neq \Sigma^*$. Then the shortest word in $\overline{\mathrm{Suff}(L(M))}$ is of length $\leq 2^n$. There exist DFA's achieving $e^{\sqrt{cn \log n(1+o(1))}}$ for a constant c, and there exist NFA's achieving 2^{dn} for some constant d.

Proof for suffixes

Proof. The upper bound of 2^n is just like the case for prefixes.

The example for DFA's achieving $e^{c\sqrt{n \log n(1+o(1))}}$ for some constant c can be constructed by using the construction for DFA suffixes we saw before, with each M_i a unary DFA accepting $b^{p_i-1}(b^{p_i})^*$ for primes $p_1 = 2$, $p_2 = 3$, etc.

The construction generates an automaton of $O(p_1 + p_2 + \cdots p_n)$ states, and the shortest word omitted as a suffix is of length $\geq p_1 p_2 \cdots p_n$.

Proof for suffixes

For NFA's, we take the construction in the proof of the result for prefixes, and construct the NFA for the reversed language. This can be done by reversing the order of each transition, changing the initial state to final and all final states to initial.
This creates a "generalized NFA" with a set of initial states, but this can easily be simulated by an ordinary NFA by adding a new initial state, adding ϵ-transitions to the former final states, and then removing ϵ-transitions using the usual algorithm.
This gives an example achieving 2^{dn} for some constant d. ■

Proof for factors

Theorem. Let M be a DFA or NFA with n states. Suppose $\mathrm{Fact}(L(M)) \neq \Sigma^*$. Then the shortest word in $\overline{\mathrm{Fact}(L(M))}$ is

(a) of length $O(n^2)$ if M is a DFA, and there exist examples achieving $\Omega(n^2)$;

Proof for factors

Theorem. Let M be a DFA or NFA with n states. Suppose $\mathrm{Fact}(L(M)) \neq \Sigma^*$. Then the shortest word in $\overline{\mathrm{Fact}(L(M))}$ is

(a) of length $O(n^2)$ if M is a DFA, and there exist examples achieving $\Omega(n^2)$;

(b) of length $\leq 2^n$ if M is an NFA, and there exist examples achieving 2^{cn} for some constant c.

Proof for suffixes

Proof.

(a) The bounds come from known results on synchronizing words.

Proof for suffixes

Proof.

(a) The bounds come from known results on synchronizing words.

(b) The upper bound is clear. For an example achieving 2^{cn}, we use a construction from Kao et al.
There the authors construct a "generalized" NFA M of n states with all states both initial and final, such that the shortest word not accepted is of length 2^{cn}.
Such an NFA can be converted to an ordinary NFA, as we have mentioned previously, at a cost of increasing the number of states by 1. But for such an NFA, clearly $\mathrm{Fact}(L(M)) = L(M)$, so the result follows. ■

Sets of finite words

Motivation for this work arose from testing if

(a) $S^\omega = \Sigma^\omega$,

We have resolved (a) and (b), giving fast algorithms for these problems. But (c) is still open.

Sets of finite words

Motivation for this work arose from testing if

(a) $S^\omega = \Sigma^\omega$,

(b) ${}^\omega S = {}^\omega \Sigma$, or

We have resolved (a) and (b), giving fast algorithms for these problems. But (c) is still open.

Sets of finite words

Motivation for this work arose from testing if

(a) $S^\omega = \Sigma^\omega$,

(b) ${}^\omega S = {}^\omega \Sigma$, or

(c) ${}^\omega S^\omega = {}^\omega \Sigma^\omega$ for a finite set of words S.

We have resolved (a) and (b), giving fast algorithms for these problems. But (c) is still open.

Shortest word omitted

We can also address the question of the shortest word not in $\mathrm{Fact}(S^*)$, given that $\mathrm{Fact}(S^*) \neq \Sigma^*$.

Theorem. For each $n \geq 1$ there exists a set of finite words of length $\leq n$, such that the shortest word not in $\mathrm{Fact}(S^*)$ is of length $n^2 + n - 1$.

Proof. Let $S = \Sigma^n - \{0^{n-1}1\}$. Then it is easy to verify that the shortest word not in $\mathrm{Fact}(S^*)$ is $0^{n-1}1(0^n1)^{n-1}$. ■

Stronger results have been found by Elena Pribavkina.

Open Problems

- What is the computational complexity of the following decision problem?

  Given a finite set of finite words S, decide if $\mathrm{Fact}(S^*) = \Sigma^*$

Open Problems

- What is the computational complexity of the following decision problem?

  Given a finite set of finite words S, decide if $\mathrm{Fact}(S^*) = \Sigma^*$

- Given that $\mathrm{Fact}(S^*) \neq \Sigma^*$, what is a good upper bound on the length of the shortest word in $\Sigma^* - \mathrm{Fact}(S^*)$, in terms of the size of S?