Open Problems Proposed at DCFS 2015

Compiled by Jeffrey Shallit

July 6, 2015

1 State complexity of shuffle

Proposed by Janusz Brzozowski, University of Waterloo, brzozo@cs.uwaterloo.ca.

Let K_m (respectively, L_n) be a regular language with state complexity m (respectively, n). Here we demand that the corresponding automaton be complete (i.e., that there is a transition on each letter) and that K_m and L_n be defined over the same alphabet Σ.

Question: What is a tight bound on the worst-case state complexity of the shuffle language $K_m \sqcup L_n$?

Remark: In [Campeanu, Salomaa, Yu, Tight lower bound for the state complexity of shuffle of regular languages, *J. Automata, Languages and Combinatorics*, **7** (2002) 303–310] they found a lower bound where the automata were allowed to be incomplete.

For $m = 1$ a tight bound is $2^{n-2} + 1$ provided that $|\Sigma| \geq n - 2$ [Okhotin, On the state complexity of scattered substrings and superstrings, *Fund. Inform.* **99**(3), 325–338].

My undergraduate student Bo Liu found an upper bound in 2012:

$$2^{mn-1} + 2^{(m-1)(n-1)}(2^{m-1} - 1)(2^{n-1} - 1).$$

In 2015 Aayush Rajasekaran showed that $|\Sigma| \geq mn - 1$ is required if the bound is to be met, but this bound on the alphabet size is not tight. He also enumerated all pairs of DFAs with $m = n = 2$ and found that there was only *one* witness meeting the bound (up to isomorphism).

In 2015 Marek Szykuła found witnesses for $m = 2$, $n = 3$, and sent me *26 pages* of such witnesses. He also verified that the bound holds for $m, n \leq 5$, and has a proof that the bound is tight if $m = 3$ and n is arbitrary.

Galina Jirásková and Jozef Jirásek verified that there are witnesses for $m, n \leq 5$ and $m = 4$, $n = 6$. There is no proof of reachability of all the subsets required by the bound, but Galina has a proof of distinguishability.

We plan to post a note about this problem on arXiv soon.

2 State complexity of permutation-full strings

Proposed by Jeffrey Shallit, University of Waterloo, shallit@cs.uwaterloo.ca.

Let $\Sigma_n = \{1, 2, \ldots, n\}$. By "$x$ is a subsequence of y", we mean that we can obtain the word x by striking out 0 or more symbols of y. ("Subsequence" is often called "subword" in Europe.)

For $n \geq 1$ define

$$L_n = \{x \in \Sigma_n^* \ : \ x \text{ contains, as a subsequence, all permutations of the string } 123\cdots n \ \}.$$

Thus, for example, $L_1 = 1^+$ and $L_2 = (1^+2^+1 + 2^+1^+2)(1 + 2)^*$.

Question: what is $\mathrm{sc}(L_n)$? Find good upper and lower bounds on this quantity. Here sc denotes state complexity, the smallest number of states in any deterministic finite automaton accepting L_n.

Remark: A trivial upper bound on $\mathrm{sc}(L_n)$ is $(n+1)^{n!}$. Such a DFA can be constructed by maintaining, for each of the $n!$ permutations p, an integer between 0 and n counting the length of the longest prefix of p seen so far.

Remark: This trivial upper bound is likely to be quite far from the truth. For example, it is known that $\mathrm{sc}(L_1) = 2$, $\mathrm{sc}(L_2) = 6$, $\mathrm{sc}(L_3) = 44$, and $\mathrm{sc}(L_4) = 2014$.

Remark: Rusins Freivalds points out that a somewhat better upper bound can be derived by first creating an NFA accepting $\overline{L_n}$. Here states are suffixes of permutations and the NFA attempts to "guess" a permutation that does *not* appear as a subsequence of the input and then verify the guess. So states of this NFA are all possible suffixes of all length-n permutations. The initial states are all the length-n permutations, and all states are accepting except the state denoting the empty string. This NFA has $n!(1/0!+1/1!+\cdots+1/n!)$ states, or about $e \cdot n!$, and the resulting DFA for $\overline{L_n}$ (and hence, by complementing, for L_n) has about $2^{e \cdot n!}$ states. This is still likely to be quite far from the truth.

Remark: Rusins Freivalds shows how to get an upper bound of $2^{n \cdot 2^{n-1}}$, as follows. The idea is essentially the same as in the previous paragraph, but now states of the NFA are of the form (m, S), where $S \subseteq \{1, 2, \ldots, n\} - \{m\}$ and $m \in \{1, 2, \ldots, n\}$. The initial states are of the form $(m, \emptyset)$ and there are transitions from (m, S) to (m, S) on all inputs $i \neq m$, and from (m, S) to $(m', S \cup \{m\})$ on input m, for all $m' \in \{1, 2 \ldots, n\} - (S \cup \{m\})$.

Remark: The sequence $\mathrm{sc}(L_n)$ is sequence A259482 in the Online Encyclopedia of Integer Sequences; see https://oeis.org/A259482.

Remark: The radius r_n of the minimal automaton for L_n (i.e., length of shortest string in L_n) is sequence A062714 in the OEIS. It is known that $r_n = n^2(1 - o(1))$; see the references there and

http://mathoverflow.net/questions/165303/shortest-supersequence-of-all-permutations-of-n-elements

for more information. Nobody currently knows the exact value of r_n for $n \geq 8$.

3 Shuffle languages

Proposed by Jean-Éric Pin, LIAFA, Paris, France, Jean-Eric.Pin@liafa.jussieu.fr.

This is a problem due to A. Restivo.

Characterize the smallest set of languages containing the singletons (or, equivalently, the finite languages) and closed under Boolean operations (i.e., union, complement, intersection), product (concatenation), and shuffle.

Remark: It is known that if shuffle is omitted then the smallest class is exactly the star-free regular languages.

Partial results can be found in

J. Berstel, L. Boasson, O. Carton, J.-E. Pin et A. Restivo, The expressive power of the shuffle product, *Information and Computation* **208** (2010), 1258–1272.

4 Self-verifying automata

Proposed by Jozef Jirásek, Jr., Slovakia, jirasekjozef@gmail.com.

Consider *self-verifying finite automata* (SVFAs). These are nondeterministic finite automata M over an alphabet Σ, where the states can be either accepting, rejecting, or neutral. Let $L_a(M)$ (resp., $L_r(M)$) be the set of words taking M from the start state to an accepting state (resp., rejecting state). Then M is said to be self-verifying if $L_a(M) \cup L_r(M) = \Sigma^*$ and $L_a(M) \cap L_r(M) = \emptyset$. That is, the words reaching an accepting state and the words reaching a rejecting state form a disjoint partition of Σ^*. The language accepted by the SVFA M is $L_a(M)$.

Remark: Given a nondeterministic automaton M with accepting, rejecting, and neutral states, it is PSPACE-complete to determine if M is an SVFA.

Question: What is the computational complexity of the following problem: Given an automaton M as above that is *promised* to be an SVFA, and a word w, is $w \in L(M)$?

Remark: For more about SVFAs, see Jirásková and Pighizzini, Converting self-verifying automata into deterministic automata, LATA 2009, LNCS, vol. 5457, pp. 458–469; Jirásková and Pighizzini, Optimal simulation of self-verifying automata by deterministic automata, *Inform. Comput.* **209** (2011) 528–535; and Jirásek, Jirásková, and Szabari, Operations on self-verifying finite automata, CSR 2015, LNCS, vol. 9139, pp. 231–261.

Remark: Markus Holzer and Sebastian Jakobi report that they can prove that the membership problem for SVFA's is NL-complete.

5 Dissectible languages

Proposed by Tomoyuki Yamakami, University of Fukui, Japan, TomoyukiYamakami@gmail.com.

A language C *dissects* an infinite language S if

$$|C \cap S| = |\overline{C} \cap S| = \infty,$$

where $|A| = \infty$ means that A is an infinite set. We say a class $\mathcal{C}$ of languages is *REG-dissectible* if every member of the class $\mathcal{C}$ is dissected by a certain regular language. It is relatively easy to show that CFL (class of context-free languages) is REG-dissectible.

Question 1: Is the class co-CFL (the class of all complements of context-free languages) REG-dissectible?

Question 2: Is CFL(k) (the class of languages formed by the k intersections of context-free languages) REG-dissectible?

Question 3: Is CFL_k (the kth level of the Boolean hierarchy over the class of context-free languages) REG-dissectible?

Comment: For more information on this notion and its related notion (i-separation), see Tomoyuki Yamakami and Yuichi Kato, *Information Processing Letters*, **113** (2013), 116–122.

6 Regular realizability

Proposed by Alexander Rubtsov, Moscow Institute of Physics and Technology, Russia, rubtsov99@gmail.com.

Question: Is there a language L such that NRR(L) does not reduce in log space to RR(L)?

Here NRR(L) denotes the set of nondeterministic finite automata $\mathcal{A}$ such that $L(\mathcal{A}) \cap L \neq \emptyset$, and RR($L$) denotes the set of deterministic finite automata $\mathcal{A}$ such that $L(\mathcal{A}) \cap L \neq \emptyset$. These sets (decision problems) called regular realizability problems for filter L.

Remark: If such L is a regular language, then it implies that L $\neq$ NL – the separation of deterministic and nondeterministic log space complexity classes. It is known, that if such a regular language L exists, then it is a bounded language.

Remark: In case of non-regular languages, I have no conjecture about the answer. Evidence for existance of such a language: there is an exponential gap between DFA and NFA descriptions. Evidence against: a language $L \subseteq \{a, b\}^*$ is not such a language, if $\mathrm{RR}(L \sqcup \{\#\}^*) \leq_{\log} \mathrm{RR}(L)$ – this condition looks very weak.

For more information about NRR and RR see my paper in the DCFS 2015 proceedings and arXiv:1503.05879 paper.

7 Group languages and partial commutations

Proposed by Jean-Éric Pin, LIAFA, Paris, France, Jean-Eric.Pin@liafa.jussieu.fr.

Is the closure of a regular group language under a given partial commutation always regular?

A regular language is called a "group language" if it is accepted by a permutation automaton. A permutation automaton is one in which the transitions on each letter form a permutation of the state set.

A partial commutation refers to a set of relations of the form $ab \sim ba$ for letters a, b.

The result has been proved for various partial commutations, including the case of a total commutation ($ab \sim ba$ for all letters a, b). However, I conjecture the answer is "no" in general but no example is known. More details can be found in A. Cano, G. Guaiana et J.-É. Pin, Regular languages and partial commutations, *Information and Computation* **230** (2013), 76–96.

8 Unix regular expressions

Proposed by Cezar Câmpeanu, University of PEI, Canada, ccampeanu@upei.ca.

Consider the set of Unix regular expressions with back referencing. Study the properties of such expressions that are "deterministic" or "unambiguous"; that is, there is exactly one way to use the to parse the strings they apply to.

For more about the properties of such regular expressions, see Câmpeanu, Salomaa, and Yu, A formal study of practical regular expressions, *Int. J. Found. Comput. Sci.* **14** (2003), 1007–1018.