Finding the Smallest Turing Machine Using $k \log(n)$ Non-deterministic Guesses

Michael Wehar
University at Buffalo
mwehar@buffalo.edu

May 4, 2014

Consider that we are given a number m and two disjoint finite sets of strings A and R. Does there exist a DFA with at most m states that accepts the strings in A and rejects the string in R? We refer to this problem as the inference problem for DFA's and denote it by INF_{DFA}. It was shown by E. Mark Gold in [4] that INF_{DFA} is NP-hard. To the best of my knowledge, it is not known whether INF_{DFA} remains NP-Hard when restricting A and R such that both sets contain exactly one string. We refer to this problem as separating two words and denote it by S2W_{DFA}. Separating two words is related to constructing a minimum DFA that accepts one string and rejects another. From a combinatorial point of view, this problem has been well studied and several upper bounds have been given for the size of a minimum DFA in terms of the length of the string to accept and the string to reject [8]. If the strings have length at most n, it is an open problem to resolve whether a minimum DFA always has $O(\log(n))$ states.

Let's consider the separating two words problem for computational models with memory. Consider that we are given a number m and two bit strings s_1 and s_2. Does there exist a 2PDA with at most m states that accepts s_1 and rejects s_2? We denote this problem by S2W_{2PDA}. It was shown that if s_1 and s_2 have length at most n, then there exists a 2PDA with $O(\log(n))$ states that accepts s_1 and rejects s_2 [3]. Notice that there are at most $2^{O(\log(n) \log\log(n))}$ 2PDA's with $\log(n)$ states. Therefore, S2W_{2PDA} can be deterministically solved in $2^{O(\log(n) \log\log(n))}$ time by brute force search. One can non-deterministically solve S2W_{2PDA} in $n^{O(1)}$ time using $O(\log(n) \log\log(n))$ non-deterministic guesses. We will improve on this result by showing that there exists a Turing machine with at most $O(\frac{\log(n)}{\log\log(n)})$ states that accepts s_1 and rejects s_2.

We will now consider the inference problem for clocked Turing machines introduced by Manuel Blum in [1]. Consider that we are given a number m and a finite set T of triples of the form (s, b, t) where s is a bit string, b is a single bit, and t is number represented in unary. A Turing machine M is said to match a triple (s, b, t) if M halts on input s in at most t steps and M accepts s if and only if $b = 1$. Does there exist a Turing machine with at most m states that matches all triples in T? We denote this problem by $\mathrm{INF}_{\mathrm{CTM}}$. Without too much effort, one can show $\mathrm{INF}_{\mathrm{CTM}} \in \mathrm{NP}$. To the best of my knowledge, it is not known if $\mathrm{INF}_{\mathrm{CTM}}$ is NP-Hard. We will show that if there exists a Turing machine that matches all triples in T and T has size k, then there is a Turing machine that matches all triples in T with at most $k\frac{\log(n)}{\log\log(n)}$ states. Consider the fixed parameter problem where T contains at most k triples. We denote this problem by k-INF_{CTM}. It follows that k-INF_{CTM} can be deterministically solved in $O(n^k)$ time and k-INF_{CTM} can be non-deterministically solved in $n^{O(1)}$ time using $O(k\log(n))$ non-deterministic guesses.

If we restrict ourselves to only two triples, we get 2-$\mathrm{INF}_{\mathrm{CTM}}$ which we will also denote by $\mathrm{S2W}_{\mathrm{CTM}}$. Notice that $\mathrm{S2W}_{\mathrm{CTM}} \in \mathrm{P}$, but we don't know if $\mathrm{S2W}_{\mathrm{DFA}}$ is solvable in polynomial time. One might think that $\mathrm{S2W}_{\mathrm{DFA}}$ is easier because DFA's are computationally much simpler than Turing machines. However, this may not be the case because there always exists a small Turing machine that separates two given strings. Therefore, we need only search through polynomially many Turing machines to find a smallest one that matches both triples.

From a computational complexity point of view, resolving whether the k-INF_{CTM} problems are deterministically solvable in $n^{O(1)}$ time could shed light on the relationship between deterministic time and non-deterministic time. Consider the following complexity class for an arbitrary pair of functions $f(n)$ and $g(n)$. Let $\mathrm{NTIGU}(f(n), g(n))$ denote the set of problems solvable by a non-deterministic Turing machine in at most $f(n)$ time using at most $g(n)$ non-deterministic guesses. We show that k-$\mathrm{INF}_{\mathrm{CTM}} \in \mathrm{NTIGU}(n^{O(1)}, k\log(n))$ and k-$\mathrm{INF}_{\mathrm{CTM}} \in \mathrm{DTIME}(n^k)$. If it happens to be the case that k-$\mathrm{INF}_{\mathrm{CTM}} \notin \mathrm{DTIME}(n^{O(1)})$, then there is an immense gap between P and NP. In particular, for every $g(n) = \omega(\log(n))$, $\mathrm{NTIGU}(\mathsf{poly}(n), g(n)) \nsubseteq \mathrm{P}$. However, one might be able to show that $\mathrm{P} \neq \mathrm{NP}$ implies that such a gap exists.

For an arbitrary function $g(n)$, what can we say about the relationship between $\mathrm{NTIGU}(\mathsf{poly}(n), g(n))$ and $\mathrm{NTISP}(\mathsf{poly}(n), g(n))$? If $\mathrm{P} = \mathrm{NL}$, then one can space efficiently simulate polynomial time verifiers to get $\mathrm{NTIGU}(\mathsf{poly}(n), g(n)) \subseteq \mathrm{NTISP}(\mathsf{poly}(n), g(n))$. Also, it's worth mentioning that although we do not show that k-$\mathrm{INF}_{\mathrm{CTM}}$ is complete for NTIGU($\mathsf{poly}(n), O(\log(n))$), there exist natural problems that are complete for NTISP($\mathsf{poly}(n), O(\log(n))$). In particular, for any fixed k, intersection non-emptiness for k acyclic DFA's, those without directed cycles, is complete for NTISP($\mathsf{poly}(n), O(\log(n))$).

Acknowledgments

I greatly appreciate the help and suggestions that I received from Joseph Swernofsky. In addition, I would like to thank Manuel Blum for introducing me to inductive inference and for the many discussions that we had on the subject.

References

[1] Manuel Blum. *Timed Inductive Inference*. Unpublished, 2012.

[2] James Currie, Holger Petersen, John Michael Robson, and Jeffrey Shallit. Separating words with small grammars. *J. Automata, Languages, and Combinatorics*, 4:101–110, 1999.

[3] Erik D. Demaine, Sarah Eisenstat, Jeffrey Shallit, and David A. Wilson. Remarks on separating words. In Markus Holzer, Martin Kutrib, and Giovanni Pighizzini, editors, *Descriptional Complexity of Formal Systems*, volume 6808 of *Lecture Notes in Computer Science*, pages 147–157. Springer Berlin Heidelberg, 2011.

[4] E Mark Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302 – 320, 1978.

[5] Gregor Gramlich and Ralf Herrmann. Learning unary automata. *Presented at Descriptional Complexity of Formal Systems*, 2005.

[6] Leonard Pitt and Manfred K. Warmuth. The minimum consistent dfa problem cannot be approximated within any polynomial. *J. ACM*, 40(1):95–142, January 1993.

[7] J. M. Robson. Separating words with machines and groups. *RAIRO - Theoretical Informatics and Applications - Informatique Thorique et Applications*, 30(1):81–86, 1996.

[8] Jeffrey Shallit. The separating words problem. *Presented at McMaster University optimization seminar*, 2010.