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 $s_2$. 

1
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 \( \text{INF}_{\text{CTM}} \). Without too much effort, one can show \( \text{INF}_{\text{CTM}} \in \text{NP} \). To the best of my knowledge, it is not known if \( \text{INF}_{\text{CTM}} \) is \( \text{NP} \)-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 \( k \log(n) \log \log(n) \) states.

Consider the fixed parameter problem where \( T \) contains at most \( k \) triples. We denote this problem by \( k \)-\( \text{INF}_{\text{CTM}} \). It follows that \( k \)-\( \text{INF}_{\text{CTM}} \) can be deterministically solved in \( O(n^k) \) time and \( k \)-\( \text{INF}_{\text{CTM}} \) 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-\( \text{INF}_{\text{CTM}} \) which we will also denote by \( \text{S2W}_{\text{CTM}} \). Notice that \( \text{S2W}_{\text{CTM}} \in \text{P} \), but we don’t know if \( \text{S2W}_{\text{DFA}} \) is solvable in polynomial time. One might think that \( \text{S2W}_{\text{DFA}} \) 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 \( k \)-\( \text{INF}_{\text{CTM}} \) 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 \( \text{NTIGU}(f(n), g(n)) \) 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 \( k \)-\( \text{INF}_{\text{CTM}} \) can be deterministically solved in \( O(n^k) \) time and \( k \)-\( \text{INF}_{\text{CTM}} \) can be non-deterministically solved in \( n^{O(1)} \) time using \( O(k \log(n)) \) non-deterministic guesses.

If it happens to be the case that \( k \)-\( \text{INF}_{\text{CTM}} \notin \text{DTIME}(n^{O(1)}) \), then there is an immense gap between \( \text{P} \) and \( \text{NP} \). In particular, for every \( g(n) = \omega(\log(n)) \), \( \text{NTIGU}(\text{poly}(n), g(n)) \notin \text{P} \). However, one might be able to show that \( \text{P} \neq \text{NP} \) implies that such a gap exists.

For an arbitrary function \( g(n) \), what can we say about the relationship between \( \text{NTIGU}(\text{poly}(n), g(n)) \) and \( \text{NTISP}(\text{poly}(n), g(n)) \)? If \( \text{P} = \text{NL} \), then one can space efficiently simulate polynomial time verifiers to get \( \text{NTIGU}(\text{poly}(n), g(n)) \subseteq \text{NTISP}(\text{poly}(n), g(n)) \). Also, it’s worth mentioning that although we do not show that \( k \)-\( \text{INF}_{\text{CTM}} \) is com-
plete for NTIGU(poly(n), O(log(n))), there exist natural problems that are complete for NTISP(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(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


