Verifying Time Complexity of Turing Machines

David Gajser
IMFM, Jadranska 19, 1000 Ljubljana, Slovenia
E-mail: david.gajser@fmf.uni-lj.si

Thesis summary

Keywords: Turing machine, relativization, NP-completeness, crossing sequence, decidability, lower bound, time complexity, running time, linear time

Received: November 4, 2016

This paper presents a summary of the doctoral dissertation [1] of the author, which analyzes in detail the following problem. For a function $T : \mathbb{N} \to \mathbb{R}_{\geq 0}$, how hard is it to verify whether a given Turing machine runs in time at most $T(n)$? Is it even possible?

Povzetek: Prispevek predstavlja povzetek doktorske disertacije [1] avtorja, v kateri je podrobneje obravnavan naslednji problem. Naj bo $T : \mathbb{N} \to \mathbb{R}_{\geq 0}$ poljubna funkcija. Kako težko je preveriti, ali je časovna zahtevnost danega Turingovega stroja $T(n)$? Je to sploh mogoče preveriti?

While it is tempting to argue about a Turing machine's time complexity, we cannot algorithmically tell even whether a given Turing machine halts on the empty input (a folkloric result). Can we perhaps algorithmically check whether it is of a specified time complexity? While the answer is *no* in most cases, there is an interesting case where the answer is *yes*: verifying a time bound $T(n) = Cn + D$, $C, D \in \mathbb{N}$, for a given one-tape Turing machine.

There are at least two natural types of questions about whether a Turing machine obeys a given time bound:

- For a function $T : \mathbb{N} \to \mathbb{R}_{>0}$, does a given Turing machine run in time $\mathrm{O}(T(n))$?
- For a function $T : \mathbb{N} \to \mathbb{R}_{>0}$, does a given Turing machine run in time $T(n)$, i.e., does it make at most $T(n)$ steps on all computations on inputs of length n for all n?

It is a folklore that it is undecidable whether a Turing machine runs in time $\mathrm{O}(1)$, thus the first question is undecidable for all practical functions T. We state a generalization of this well known fact in the dissertation and prove it using standard techniques. However, for the second question, it is not hard to see that it is decidable whether a given Turing machine runs in time C for some constant $C \in \mathbb{N}$: we just need to simulate the given Turing machine on all the inputs up to the length C [2]. It would be interesting if the second question were decidable also for linear functions T. However, we prove that it is decidable whether a multi-tape Turing machine runs in time $T(n)$ if and only if we have the "eccentric" case $T(n_0) < n_0 + 1$ for some $n_0 \in \mathbb{N}$. The time bound $n + 1$ is special because it minimally enables a multi-tape Turing machine to mark time while simulating another Turing machine. The timekeeping can be done on the input tape by just moving the head to the right until the blank symbol at the end marks $n + 1$ steps, while the other tapes are used for the simulation. But what if the simulation has to be performed on the same tape as the timekeeping, i.e., how much time do we need for a one-tape Turing machine to count steps and simulate another Turing machine? We show in [2] that $\Omega(n \log n)$ time is enough:

Let $T : \mathbb{N} \to \mathbb{R}_{>0}$ be a function such that $T(n) = \Omega(n \log n)$ and, for all $n \in \mathbb{N}$, it holds $T(n) \geq n+1$. Then it is undecidable whether a given one-tape Turing machine runs in time $T(n)$.

We also provide a nice contrast [2]:

For any "nice" function $T : \mathbb{N} \to \mathbb{R}_{>0}$, $T(n) = \mathrm{o}(n \log n)$, it is decidable whether a given one-tape Turing machine runs in time $T(n)$.

Hence, a one-tape Turing machine that runs in time $T(n) = \mathrm{o}(n \log n)$ cannot count steps while simulating another Turing machine. There is another well known fact about one-tape Turing machines that makes the time bounds $\Theta(n \log n)$ special: these bounds are the tightest that allow a one-tape Turing machine to recognize a non-regular language [4].

An interesting fact is that one-tape Turing machines that run in time $\mathrm{o}(n \log n)$ actually run in linear time [2, 4]. Thus, we can conclude that the most natural algorithmically verifiable time bounds for one-tape Turing machines are the linear ones. This motivates the analysis of the computational complexity of the following problems parameterized by integers $C, D \in \mathbb{N}$. The problem $\textsc{Halt}^1_{Cn+D}$ is defined as

Given a one-tape NTM*, does it run in time $Cn + D$?

*NTM is an abbreviation for non-deterministic Turing machine. Until

and the problem D-HALT^1_{Cn+D} is defined as

Given a one-tape DTM†, does it run in time $Cn + D$?

For the analyses of the problems HALT^1_{Cn+D} and D-HALT^1_{Cn+D}, we fix an input alphabet Σ, $|\Sigma| \geq 2$, and a tape alphabet $\Gamma \supset \Sigma$. It follows that the length of most standard encodings of q-state one-tape Turing machines is $\text{O}(q^2)$. To make it simple, we assume that each code of a q-state one-tape Turing machines has length $\Theta(q^2)$ and when we will talk about the complexity of the problems HALT^1_{Cn+D}, we will always use q as the parameter to measure the length of the input. We prove the following [3].

For all integers $C \geq 2$ and $D \geq 1$, all of the following holds.

1. *The problems* HALT^1_{Cn+D} *and* D-HALT^1_{Cn+D} *are* co-NP-complete.
2. *The problems* HALT^1_{Cn+D} *and* D-HALT^1_{Cn+D} cannot *be solved in time* $\text{o}(q^{(C-1)/4})$ *by multi-tape NTMs.*
3. *The complements of the problems* HALT^1_{Cn+D} *and* D-HALT^1_{Cn+D} can *be solved in time* $\text{O}(q^{C+2})$ *by multi-tape NTMs.*
4. *The complement of the problem* HALT^1_{Cn+D} cannot *be solved in time* $\text{o}(q^{(C-1)/2})$ *by multi-tape NTMs.*
5. *The complement of the problem* D-HALT^1_{Cn+D} cannot *be solved in time* $\text{o}(q^{(C-1)/4})$ *by multi-tape NTMs.*

To put the theorem in short, the problems HALT^1_{Cn+D} and D-HALT^1_{Cn+D} are co-NP-complete with a non-deterministic and co-non-deterministic time complexity lower bound $\Omega(q^{0.25C-1})$ and a co-non-deterministic time complexity upper bound $\text{O}(q^{C+2})$.

The main techical tools in the analyses of one-tape Turing machines that we used were crossing sequences and diagonalization. We argue that our main results are proved with techniques that relativize.

References

[1] D. Gajser. *Verifying Time Complexity of Turing Machines*. FMF UL, 2015. Thesis also available on http://eccc.hpi-web.de/static/books/Verifying_Time_Complexity_of_Turing_Machines/.

[2] D. Gajser. Verifying time complexity of Turing machines. *Theor. Comput. Sci.*, 600:86 – 97, 2015.

[3] D. Gajser. Verifying whether one-tape Turing machines run in linear time. *ECCC*, TR15-036, 2015.

[4] G. Pighizzini. Nondeterministic one-tape off-line Turing machines and their time complexity. *J. Autom. Lang. Comb.*, 14(1):107–124, 2009.

this point it was irrelevant whether the Turing machines were deterministic or non-deterministic.

†DTM is an abbreviation for deterministic Turing machine.