On multiply-exponential write-once Turing machines

Maciej Zielenkiewicz   Jacek Chrząszcz   Aleksy Schubert

Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland

January 31, 2015

Write-once Turing machine.

- First studied by Hao Wang [1] in 1957.
- A Turing machine.
- Alphabet: 0 (blank), 1.
- 0 can be changed to 1.
- 1 **can't** be changed to 0.
- We assume tape infinite in one direction (easy reduction).
- Notation: "D" for deterministic, "N" for non-deterministic and "A" for alternating

Known results

- a value can be stored in a way which permits t updates at the expense of increasing the storage size asymptotically $t/\log t$ times (Rivest and Shamir [2])
- an algorithm with a given complexity using k space can be rewritten to the write-only (deterministic) RAM machine model (with a constant amount of additional multiple-write memory) with an increase of running time by a factor of $O(\log n/\log\log n)$ or by a factor of $O(\log k)$ (Irani et al, [3])
- WO–PSpace $=$P (Irani et al, [3])

Our reductions

- k–ExpTime = k–DExpWOSpace = k–DExpWOTime
- Same for nondeterministic variants.
- k–AExpWOTime = k–AExpTime = k-1–ExpSpace.

k–ExpTime $\subseteq$ k–DExpWOSpace

- Suppose input length n, running time $f(n) = \overbrace{2^{2^{\cdot^{\cdot^{\cdot^{n^t}}}}}}^{k+2}$.
- This means a tape of length at most $f(n)$ and at most $f(n)$ updates.
- We can easily represent it in $O(f(n) \cdot f(n)) = O(\overbrace{2^{2^{\cdot^{\cdot^{\cdot^{n^t+1}}}}}}^{k+2})$ memory, which is in k–DExpWOSpace.

k–ExpTime $\supseteq$ k–DExpWOSpace

- We calculate how many writes a machine could make. . .

Lemma 1

A k-DExpWOSpace machine $\mathcal{A}$ with space complexity $f(n)$ started in a configuration in which it eventually stops makes at most $O(f(n) \cdot s)$ steps between any two consecutive writes.

- $\mathcal{A}$: s states, acts as two-way DFS automaton between writes.
- Uses at most $f(n)$ tape.
- Visits a losation at most s times (otherwise the sequence of states between the two visits of the same location in the same state could be repeated any number of times $\rightarrow$ nonterminating Ψ).

Lemma 1N

A k-NExpWOSpace machine $\mathcal{A}$ with space complexity $f(n)$ started in a configuration in which it eventually stops has a run that makes at most $O(f(n) \cdot s)$ steps between any two consecutive writes.

- We shorten the runs instead of showing nontermination.
- we can simulate the operations of the non-deterministic two-way finite automaton between two consecutive writes.
- In case the automaton is twice in the same state in a given location on the tape we can just skip the computation steps between the two visits.
- Iterating the procedure results in a run that visits each location in each state at most once. The number of steps is therefore $f(n) \cdot s$.

k–DExpWOTime = k–ExpTime

- k–DExpWOTime $\subseteq$ k–ExpTime is trivial
- k–DExpWOTime $\supseteq$ k–ExpTime: simulation.
  - Suppose that the k–ExpTime machine has time complexity of $f(n) = \overbrace{2^{2^{\cdot^{\cdot^{\cdot^{n^n}}}}}}^{k+2}$.
  - We copy the entire tape on each write that changes a marked cell to blank.
  - Cost of simulation of a single write: $f(n)^2$, and reads can be performed without any additional steps.
  - Running time will be $O((f(n))^3)$, which does not drive us outside the k–ExpTime complexity class.

Automata writing at the end of tape

- If we restrict writing to the end of the tape, such machines are not universal.
- Let's have a bigger alphabet (≥ 2 non-blank symbols).
- Suppose that blanks are only at the end (other – replaced with dummy symbol).
- Then we can simulate it without having a full copy of memory.
- First we convert our machine so that it always goes from left to right and then to the beginning of the tape and so on. Then we show a pumping lemma operating on the automaton's memory.

Lemma 2

For a given WO-automaton $\mathcal{A}$ with k states which writes only at the end of the tape, suppose s is a state in which $\mathcal{A}$ writes to the tape. Then there exists an automaton $\mathcal{A}_s$ which simulates the run of $\mathcal{A}$ from s to the next writing state, which moves to the beginning of the tape and then only reads the tape from left to right. When running from s to the next writing state, $\mathcal{A}$ does no writes, so can be seen as a DFA. by results of Vardi [?] it can be converted to one-way automaton at the expense of increasing the number of states to $O(\exp k)$.

Pumping lemma

For write-only automaton $\mathcal{A}$ which writes only at the end and has k states, we can pump any word of length at least $p(k) = O(2^{\exp(k)^k})$ with respect to runs between writes with a single pumping scheme for all states.

We have at most k writing states. Using Lemma 2 we build automata for all of these states. Then we pump the product automaton of all those automata, which has $K = (\exp(k^2))^k$ states, so words longer than $O(2^K)$ can be pumped down.

Pumping

- Use pumping lemma to shorten the runs between writes, length - depends on the number of states.
- $\ldots +$ some operations to do the pumping.
- We run the automaton for $2 \cdot p(n)$ steps from the beginning and for $p(n)$ steps backwards from the end (this needs nondeterminism, but we can simulate it with deterministic automaton at an exponential cost in term of the number of states) and find the state which occurs at least twice (positions i_1, i_2) in the normal run and at least once (i_3) in the backwards run.
- Then we can pump down the tape between positions i_1 and i_3; we stop simulating read run.
- We do not store the pumped down content.
- Simulation needs only an amount of memory which depends only on the number of states plus the size of the input, so the acceptance problem for this class of machines can be decided in PSpace.

Hao Wang.
A variant to Turing's theory of computing machines.
J. ACM, 4(1):63–92, January 1957.

Ronald L. Rivest and Adi Shamir.
How to reuse a "write-once" memory.
Inf. and Control, 55(1-3):1 – 19, 1982.

Sandy Irani, Moni Naor, and Ronitt Rubinfeld.
On the time and space complexity of computation using write-once memory or is pen really much worse than pencil?
Math. syst. theory, 25(2):141–159, 1992.