MIT/LCS/TM-148

SPACE-BOUNDED SIMULATION OF MULTITAPE TURING MACHINES

Leonard M. Adleman

Michael C. Loui

January 1980

Space-Bounded Simulation of Multitape Turing Machines

Leonard M. Adleman and Michael C. Loui

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

December 1979

Abstract. A new proof of a theorem of Hopcroft, Paul, and Valiant is presented: every deterministic multitape Turing machine of time complexity $T(n)$ can be simulated by a deterministic Turing machine of space complexity $T(n)/\log T(n)$. The proof includes an overlap argument.

Key Words: Turing machine, time complexity, space complexity, overlap.

This report was prepared with the support of National Science Foundation Grants No. MCS-77-19754 and No. MCS-78-04343 and the Fannie and John Hertz Foundation.

1. Introduction

We present a new, direct proof of a theorem of Hopcroft, Paul, and Valiant [1]: every deterministic multitape Turing machine that runs in time $T(n)$ can be simulated by a deterministic Turing machine that uses space $T(n)/\log T(n)$. Earlier results [2], [4] apply to Turing machines with only one tape. Paul and Reischuk [6] establish the strongest theorems known about simulations by space-bounded deterministic Turing machines: every logarithmic cost random access machine that runs in time $T(n)$ can be simulated in space $T(n)/\log T(n)$; every multidimensional Turing machine that runs in time $T(n)$ can be simulated in space $T(n)(\log \log T(n))/\log T(n)$.

A multitape Turing machine has a finite number of worktapes, each with a single head, and a separate two-way read-only input tape. The worktapes are infinite in both left and right directions. Initially, the worktapes are blank, and the head on the input tape is positioned on the leftmost symbol of the input word. The machine accepts the word by entering an accepting state and halting.

The class of languages accepted by deterministic multitape Turing machines of time complexity $T(n)$ on inputs of length n is denoted $DTIME(T(n))$. The class of languages of space complexity $S(n)$ is denoted $DSPACE(S(n))$. The corresponding classes for nondeterministic machines are $NTIME(T(n))$ and $NSPACE(S(n))$.

2. Simulation

Let M be a deterministic Turing machine of time complexity $T(n)$. Assume that M reads all of its input: $T(n) \geq n$. Set $S(n) = T(n)/\log_2 T(n)$. We present a simulation of M by a nondeterministic machine M' of space complexity $O(S(n))$, and subsequently, in Section 3, we make the simulation deterministic without increasing the space.

To achieve a space-efficient simulation, M' keeps only part of the contents of the tapes of M and recomputes the contents of individual tape cells whenever they are needed. Assume for simplicity that M has just one worktape; at the end of this section we handle multiple worktapes.

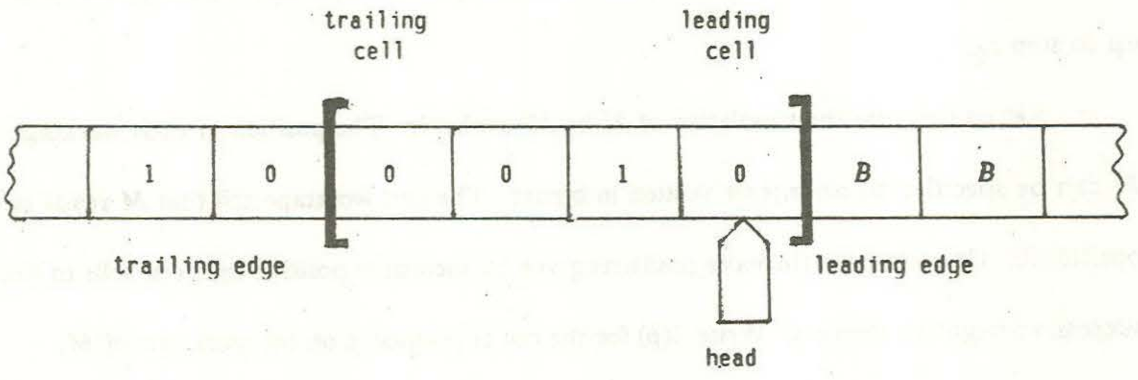
We first describe M' informally; a precise description appears below. On an input word of length n, machine M' partitions $O(S(n))$ of its tape cells into several windows of fixed size. Each window holds the contents of contiguous tape cells of M. See Figure 1. Starting from the initial configuration, as the worktape head of M moves rightward across the leading edge, window 1 shifts to the right; M' forgets the symbols that drop off the trailing edge at the left. As the worktape head proceeds to the right, every new tape cell that it encounters is initially blank. Although the worktape head of M may move leftward, M' will have the contents of the cell read by the head as long as it remains within window 1.

Finally, at step s_1 of its computation, M reads cell C to the left of the trailing edge of window 1. See Figure 2. At this step M' begins to use window 2, which moves leftward. Using window 1, M' repeats the entire simulation up to step s_1 to initialize window 2. Knowing the contents of C in window 2, M' simulates M at step s_1. Whenever M reads the tape cell to the left of the leading (left) edge of window 2, it uses window 1 to reconstruct the contents of that cell.

Windows are unidirectional. Window 1 always shifts to the right, window 2 to the left.

Figure 1.

(a) Window 1 shifts to the right. The symbol *B* denotes a blank on a cell not yet visited.



(b) As the worktape head of *M* moves to the right past the leading edge, the window shifts rightward.

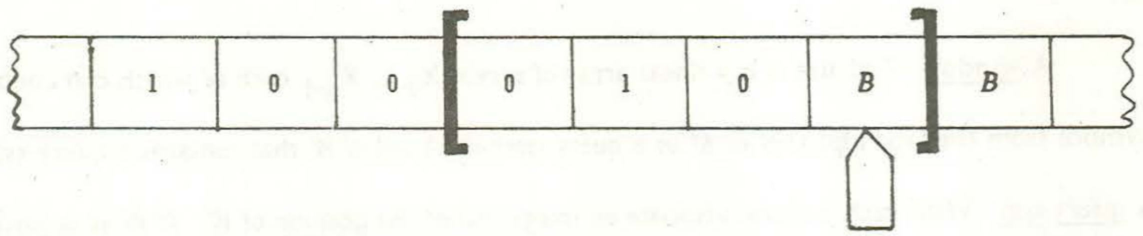
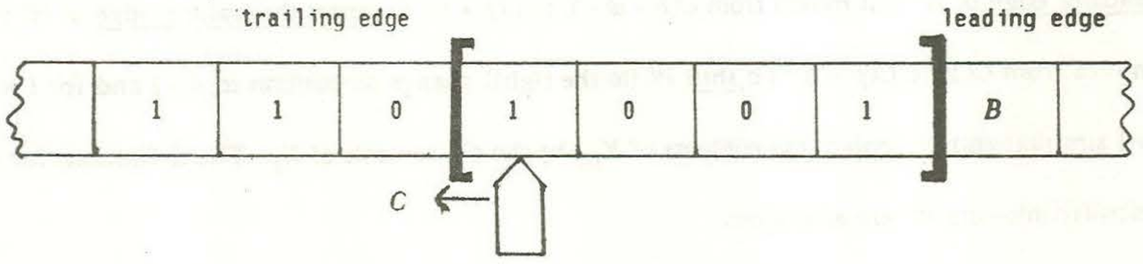


Figure 2. The worktape head of *M* moves out of window 1 onto cell *C*.



At step s_2 of the computation of M, when the worktape head of M reads a cell to the right of the trailing edge of window 2, M' begins to use window 3. It employs windows 1 and 2 to reconstruct the contents of cells beyond the leading edge of window 3 by repeating the simulation up to step s_2.

Let us describe the simulation of M by M' explicitly. The position of each worktape cell of M can be specified by an integer written in binary. The first worktape cell that M reads is at position 0. Cells to the right have positions given by successive positive integers, cells to the left by successive negative integers. Write $C(p)$ for the cell at position p on the worktape of M.

The tape alphabet of M' comprises the tape alphabet of M, symbols for the states of M, and a query-symbol Q_σ for each σ in the tape alphabet of M. Let B represent the blank symbol used by M.

A window W of size w is a linear array of w cells $X_0, \ldots, X_{w-1}$, each of which can contain a symbol from the tape alphabet of M or a query-symbol. A cell of W that contains a query-symbol is a query-cell. With each window associate an integer called the position of W. If W is at position p, then for $i = 0, \ldots, w-1$, cell X_i represents cell $C(p + i)$. Every window has a fixed initial position and a fixed direction (*left* or *right*). Suppose W, which moves rightward, is at position p. The leading cell of W is X_{w-1}, which represents $C(p + w - 1)$; the trailing cell is X_0. The head of M crosses the leading edge of W if it moves from $C(p + w - 1)$ to $C(p + w)$; it crosses the trailing edge of W if it moves from $C(p)$ to $C(p - 1)$. To shift W (to the right), change its position to $p + 1$ and for $i = 1, \ldots, w-1$ simultaneously, replace the contents of X_{i-1} by the old contents of X_i. The definitions for a leftward moving W are analogous.

For $i = 1, 2, \ldots, O(S(n)/\log T(n))$, M' sets up a data structure D_i that contains the following information, including a window W_i of size w_i:

Fixed information

- Size w_i of W_i (the number of tape cells that it contains)
- Direction of W_i (*left* or *right*)
- Initial position of W_i

Changing information

- Contents of W_i
- Position of W_i
- State of the machine M
- Position of the input head of M
- Position of the worktape head of M
- Step counter (to specify the simulated time step)
- Status of W_i –
  - *current*,
  - *suspended* (waiting for the contents of the query-cell),
  - *initializing* (waiting for contents of all cells), or
  - *inactive*
- Last-written step (the last step at which the query-cell was written)

Throughout the computation of M' every window has at most one query-cell, and at most one window is *current*.

Simulation of M

Phase 0. Set up windows. Nondeterministically partition $O(S(n))$ tape into $O(S(n)/\log T(n))$ windows. For each window set its status *inactive* and guess its direction and its initial position.

Phase 1. Initialize D_1. Set the tape contents of W_1 to blanks. Set the step counter of D_1 to 1, the head positions to 0, and the state to the initial state of M. Make W_1 *current*; set $i \leftarrow 1$.

Phase 2. Single step. Simulate one step of M in the current window W_i: after reading cell X, which in W_i represents the cell read by M, write a new symbol σ on X. For $j > i$, if Y is the query-cell of W_j, and Y represents the same cell as X, and the last-written step of D_j is less than the value of the step counter of D_i, then write the query-symbol Q_σ on Y in W_j and copy the step counter of D_i into the the last-written step of D_j. Increment the step counter of D_i by 1 and change the state and head positions of D_i to record the new state and head positions. If the worktape head of M crosses the leading edge of W_i, then go to Phase 3. If the head crosses the trailing edge of W_i, then go to Phase 4. Otherwise, continue with Phase 2.

Phase 3. Recomputation. If $i = 1$, then W_1 is the current window; in this case shift W_1, make the leading cell blank, and go to Phase 2. Otherwise, shift W_i, set its status *suspended*, write Q_B in its leading cell, and set the last-written step of D_i to 0. For all $j < i$ set the status of W_j *inactive*. Go to Phase 1.

Phase 4. Next window. Make W_i *inactive*. Put $i \leftarrow i + 1$.

Case 4a: W_i is *suspended*. Convert the query-symbol Q_σ to the corresponding symbol σ, make the status of W_i *current*, and resume the simulation at Phase 2.

Case 4b: W_i is *initializing*. Convert the query-symbol Q_σ on the query-cell of W_i to the corresponding symbol σ. If the query-cell is the last cell of W_i, then make the status of W_i *current*, and go to Phase 2. Otherwise, make the next cell X of W_i its query-cell; write Q_B on X and set the last-written step of D_i to 0. For all $j < i$ set the status of W_j *inactive*. Go to Phase 1.

Case 4c: W_i is *inactive*. Make the status of W_i *initializing*. Verify that when W_i is placed according to its initial position p, the position of the worktape head of M is between p and $p + w - 1$; if not, then reject the input word and halt. Copy the state, step counter, and head positions of D_{i-1} into the corresponding parts of D_i. Write Q_B on the first cell of W_i, and set the last-written step of D_i to 0. For all $j < i$ set the status of W_j *inactive*. Restart at Phase 1. □

Machine M' halts when M halts in Phase 2, when some window has an inappropriate initial position in Phase 4, or when M' runs out of windows. In the latter two cases M' reports a failure. Section 3 proves that for every input word some computation of M' simulates M until M halts.

Since M uses space $O(T(n))$, the position of every cell visited by M can be specified in space $O(\log T(n))$. The position of the input head can be recorded in space $O(\log n) \leq O(\log T(n))$. Each of the $T(n)$ steps of the computation of M can be specified in space $O(\log T(n))$. Consequently, the $O(S(n)/\log T(n))$ data structures D_i occupy

$$O(S(n)/\log T(n))O(\log T(n)) + \sum_i w_i = O(S(n))$$

tape cells. Therefore, M' uses space $O(S(n))$.

To prove that M' simulates M, it suffices to observe that throughout the computation of M', for every j, if W_j is *current* or *suspended*, then at the step of the computation of M specified by the step counter of D_j, the state and head positions of D_j are the state and head positions of M, and the contents of the cells of W_j (except the query-cell) are the contents of the cells of M that they represent. We show that M' recomputes the contents of cells of M correctly. Let s be the first simulated step of M at which M' uses window W_j. Suppose W_j is *suspended*, waiting for the contents of cell C at simulated step t after s. Since W_j moves in only one direction, it could not have included a representative of C before. Thus, M does not visit C between s and t. By induction on j, machine M' correctly computes the contents of C at all steps prior to step s when it reconstructs the computation of M before s. The last symbol that M writes on C before s is the symbol that M' uses to simulate step t. If M does not visit C before step s, then C holds a blank B, and M' converts the query-symbol Q_B to a blank.

This simulation can be modified to handle multitape machines. Let M have r worktapes, numbered 1, ..., r. Use the same number of windows for each worktape; for each j, windows W_{1j}, ..., W_{rj} are used to simulate the same steps. As before, the total number of windows is $O(S(n)/\log T(n))$. Begin the simulation with windows W_{11}, ..., W_{r1}. When for some a the head on worktape a crosses the trailing edge of W_{a1}, start using W_{12}, ..., W_{r2}. Employ W_{11}, ..., W_{r1} to determine the initial contents of $W_{12}, ..., W_{r2}$ and to recompute for each a the contents of cells on worktape a when its head crosses the leading edge of window W_{a2}. In general, whenever a worktape head crosses the trailing edge of a current window, the current windows for all worktapes are changed.

3. Proofs

We prove that some partition into windows permits a successful simulation of a machine of time complexity $T(n)$ by a machine of space complexity $T(n)/\log T(n)$.

Intervals of steps of a computation are denoted

$$[t_0, t_1] = \{t: t_0 \leq t \leq t_1\}.$$

Interval $[t_0, t_1]$ immediately precedes interval $[t_2, t_3]$ if $t_2 = t_1 + 1$. For interval $J = [t_0, t_1]$, define $\min(J) = t_0$ and $\max(J) = t_1$. On a Turing machine tape a loop is a sequence of cells $(C_0, ..., C_k)$, $k \geq 0$, such that:

(i) C_i is adjacent to C_{i-1} for each i;

(ii) $C_0 = C_k$, but $C_0 \neq C_i$ for $0 < i < k$.

A tape head H traces a loop L during $[t_0, t_1]$ if L is the sequence of cells that H reads at steps t_2, $t_2 + 1, ..., t_3$ for some t_2, t_3 in $[t_0, t_1]$. At steps t_2 and t_3 tape head H reads the same cell C_0. Let t_e be a step in $[t_2, t_3]$ at which H reads a cell C_e farthest from C_0; call t_e an extreme step for L and C_e the extreme cell. The other cells of L are its interior cells. The width of L is the number of its interior cells. If there is only one distinct cell in L, then L has no interior cells, and the width of L is 0. A window of size $w + 1$ can contain representatives of the $w + 1$ cells in a loop of width w.

When a tape head traces a loop of nonzero width w, it reads the w interior cells of the loop both before and after each extreme step. This observation leads to an overlap argument [5].

During a computation of a Turing machine, if two successive visits to the same worktape cell C occur at steps u_0 and u_1, where $u_0 < u_1$, then the pair (u_0, u_1) is an overlap pair, and C is the overlap cell for the pair. For intervals J_0, J_1, let $\omega_a(J_0, J_1)$ be the set of overlap pairs (u_0, u_1) for which $u_0 \in J_0$ and $u_1 \in J_1$ and the overlap cell is on worktape a.

Lemma 1. During a computation, if the head on worktape a traces a loop L of width w during $[t_0, t_1]$ and t_e is an extreme step of L, then $|\omega_a([t_0, t_e], [t_e + 1, t_1])| \geq w$.

Lemma 2. For every m and every set $\{J_{i,j}: i = 0, ..., m; j = 0, ..., 2^i-1\}$ of intervals of steps during a T step computation of a Turing machine with r worktapes, if for every i, j, interval $J_{i,j}$ immediately precedes interval $J_{i,j+1}$ and $J_{i,2j} \cup J_{i,2j+1} = J_{i-1,j}$, then

$$\Sigma_a \Sigma_i \Sigma_j |\omega_a(J_{i,2j}, J_{i,2j+1})| \leq r\,T. \tag{1}$$

Proof. Fix a worktape a. By definition of the intervals $J_{i,j}$, the sets $\omega_a(J_{i,2j}, J_{i,2j+1})$ are pairwise disjoint. Consequently, since there are at most T overlap pairs for tape a,

$$\Sigma_i \Sigma_j |\omega_a(J_{i,2j}, J_{i,2j+1})| \leq T,$$

and (1) follows. □

Theorem 1. If $T(n) \geq n$, then $DTIME(T(n)) \subseteq NSPACE(T(n)/\log T(n))$

Proof. Let deterministic machine M with r worktapes run in time $T(n)$. Set $S(n) = 2rT(n)/\log_2 T(n)$. We demonstrate that for every input word of length n, some partition of $O(S(n))$ tape into at most $T(n)^{1/2} = O(S(n)/\log T(n))$ windows enables the nondeterministic machine M' presented in Section 2 to simulate M in space $O(S(n))$.

Consider a computation of M on an input word of length n. Let $J_{0,0}$ be the interval of all steps of the computation. Repeat the following for $i = 0, 1, ...,$ until

$$\Sigma_j\, w_{aij} \le S(n) \text{ for all } a. \tag{2}$$

> Stage i. Suppose intervals $J_{i,j}$ for $j = 0, ..., 2^i\text{-}1$ have been defined. For $a = 1, ..., r$, let w_{aij} be the width of the largest loop L_{aij} traced by the head on worktape a during $J_{i,j}$. If (2) does not hold, then $\Sigma_j\, w_{b_i ij} > S(n)$ for some b_i. For each j, let t_{eij} be an extreme step for loop $L_{b_i ij}$; set $J_{i+1,2j} = [\min(J_{i,j}), t_{eij}]$ and $J_{i+1,2j+1} = [t_{eij}+1, \max(J_{i,j})]$.

Let i_0 be the least i for which (2) holds. According to Lemma 1,

$$\Sigma_j\, |\omega_{b_i}(J_{i+1,2j}, J_{i+1,2j+1})| \ge \Sigma_j\, w_{b_i ij} > S(n)$$

for $i = 0, 1, ..., i_0\text{-}1$, hence

$$\Sigma_a\, \Sigma_{i<i_0}\, \Sigma_j\, |\omega_a(J_{i+1,2j}, J_{i+1,2j+1})| \ge i_0 S(n). \tag{3}$$

Lemma 2 and (3) together imply that

$$r\, T(n) = (S(n) \log_2 T(n))/2 \ge i_0 S(n),$$

$$i_0 \le (\log_2 T(n))/2.$$

With a partition into windows of sizes $w_{ai_0j} + 1$ for each worktape a, machine M' simulates M successfully. This partition has $2^{i_0} \le T(n)^{1/2}$ windows for each tape, and the sum of the sizes of these windows is

$$\Sigma_a\, \Sigma_j\, (w_{ai_0j} + 1) \le \Sigma_a\, (S(n) + 2^{i_0}) \le r(S(n) + T(n)^{1/2}) = O(S(n)).$$

To ensure that M' uses space $T(n)/\log T(n)$, appeal to constant-factor tape reduction [3, Theorem 10.1]. □

Theorem 2. If $T(n) \geq n$, then $DTIME(T(n)) \subseteq DSPACE(T(n)/\log T(n))$.

Proof. To circumvent the nondeterministic choices in Phase 0 of the simulation, try all partitions into windows by enumerating strings in $\{0,1\}^*$; the 0's separate the window sizes denoted by the 1's (in unary). Also, enumerate all combinations of directions of the windows and all possible initial positions for the windows. Finally, $S(n)$ may not be tape-constructible; successively try $S(n) =$ 1, 2, 3, ... until the deterministic machine has enough space to complete the simulation successfully. $\square$

Acknowledgments. Albert Meyer, Joel Seiferas, and Daniel Weise suggested expository improvements.

References

[1] J. Hopcroft, W. Paul, and L. Valiant, "On time versus space." *J. ACM* **24** (1977) 332-337.

[2] J.E. Hopcroft and J.D. Ullman, "Relations between time and tape complexities." *J. ACM* **15** (1968) 414-427.

[3] J.E. Hopcroft and J.D. Ullman, *Formal Languages and their Relation to Automata.* Addison-Wesley, Reading, Mass., 1969.

[4] M.S. Paterson, "Tape bounds for time-bounded Turing machines." *J. Comp. Sys. Sci.* **6** (1972) 116-124.

[5] M.S. Paterson, M.J. Fisher, and A.R. Meyer, "An improved overlap argument for on-line multiplication." In *Complexity of Computation*, SIAM-AMS Proc. vol. 7, ed. R. Karp, Amer. Math. Soc., 1974, pp. 97-111.

[6] W. Paul and R. Reischuk, "On time versus space II." *Proc. 20th Ann. Symp. on Foundations of Computer Science*, 1979, pp. 298-306.