DETERMINISTIC SIMULATION OF NON-DETERMINISTIC TURING MACHINES (DETAILED ABSTRACT)

Walter J. Savitch
Department of Mathematics
University of California, Berkeley, California

Summary

Computations of non-deterministic Turing machines are shown to correspond to "solving" certain mazes. The storage needed to "solve" mazes is related to the storage needed to deterministically simulate non-deterministic Turing machines. In particular, it is shown that a non-deterministic $L(n)$-tape bounded Turing machine can be simulated by an $(L(n))^2$-tape bounded Turing machine, provided $L(n) \geq \log_2 n$.

Mazes

A maze is a set of rooms connected by one-way corridors. Certain rooms are designated goal rooms, and one room is designated the start room. The maze is threadable if there is a path from the start room to some goal room. More formally:

Definition 1. A maze over Σ (a finite alphabet) is a quadruple $\mathcal{M} = (X, R, z, G)$, where X is a finite set of strings over Σ (X is the set of rooms), R is a binary relation on X (giving the corridors), $z \in X$ (z is the start room), and $G \subseteq X$ (G is the set of goal rooms).

Definition 2. The maze $\mathcal{M}$ is threadable if there is a sequence $r_1, r_2, \ldots, r_e$ of rooms such that $r_1 = z$ (the start room), $R(r_i, r_{i+1})$ holds for $i = 1, 2, \ldots, e-1$, and $r_e \in G$.

Definition 3. Let), (, and * be three new symbols. A representation of $\mathcal{M}$ is a string of the form:

$$z(x_1 * y_1^1 * y_2^1 * \cdots * y_{n(1)}^1)(x_2 * y_1^2 * y_2^2 * \cdots * y_{n(2)}^2) \cdots$$

$$\cdots (x_w * y_1^w * y_2^w * \cdots * y_{n(w)}^w) u_1 * u_2 * \cdots * u_g$$

where, z is the start room, $X = \{x_1, x_2, \ldots, x_w\}$, and for $1 \leq i \leq w$, $y_1^i, y_2^i, \ldots, y_{n(i)}^i$ is an enumeration of all $y \in X$ such that $R(x_i, y)$ holds.

Definition 4. M_Σ denotes the set of all representations of threadable mazes over Σ.

In what follows, Turing machine will mean off-line Turing machine. That is, a finite-state control attached to a read only input tape, with end markers, and to finitely many read/write work tapes. A Turing machine Z (deterministic or non-deterministic) is said to accept the set A within storage $L(n)$ if for each string w in A, Z accepts w in a computation in which no work tape head scans more than $L(n)$ squares, where n is the length of w, and if Z does not accept any string not in A.

Theorem 1. There is a deterministic Turing machine, Z_M, which accepts M_Σ within storage $(\log_2 n)^2$.

Theorem 1 is proven by exhibiting an algorithm for Z_M. The algorithm is similar to that used by Lewis, Stearns and Hartmanis[1] to show that every context-free language is accepted by a deterministic Turing machine within storage $(\log_2 n)^2$. Although we will not present it here, a unified proof of Theorem 1 and the result on context-free languages can be given (see Cook and Savitch[2]). The general idea of the algorithm for Z_M is as follows.

Suppose Z_M is given an input of length n which codes a maze. Z_M divides its work tape into blocks of length $\log_2 n$ each. Each room can be given a name which can be stored in $\log_2 n$ squares of tape. So each block can store the name of one room.

In the course of the algorithm, Z_M will have the names of two rooms r, r'' stored on its work tape, and will need to check if there is a path from r to r'' which passes through at most 2^m rooms. Furthermore Z_M will have to perform this task using only m blocks of storage. Z_M proceeds as follows. Z_M runs through (in a systematic way) all rooms r' of the maze and for each r', Z_M checks to see if there is a path from r to r' which passes through at most 2^{m-1} rooms and a path from r' to r'' which passes through at most 2^{m-1} rooms. Z_M needs one block of storage to record its guess at r', leaving it with m-1 blocks of storage. So Z_M has reduced the task of finding a path of length 2^m, using m blocks of storage, to finding paths of length 2^{m-1}, using $m-1$ blocks of storage. Z_M then reduces each path of length 2^{m-1} to two paths of length 2^{m-2}. Z_M continues to reduce the length of the paths it need check, until it need only check, for appropriate rooms r_1, r_2, whether there is a corridor leading directly from r_1 to r_2. This it can easily do using zero storage.

Now an input of length n can code a maze with at most n rooms. So if there is any threading of the maze, then there is one passing through at most $n = 2^{\log n}$ rooms. So Z_M can find this threading using $\log_2 n$ blocks of storage or a total of $(\log_2 n)^2$ squares of storage.

If we allow the machine to operate non-deterministically, then the $(\log_2 n)^2$ bound can be lowered to $\log_2 n$. That is:

Theorem 2. There is a non-deterministic Turing machine which accepts M_Σ within storage $\log_2 n$.

A straightforward crossing sequence argument shows that the $\log_2 n$ bound in Theorem 2 is the best possible, up to a constant factor.

Main Theorem

A non-deterministic Turing machine, with input w, gives rise in a natural way to a maze. The rooms of the maze are the instantaneous descriptions of the machine, and the corridors are given by the transition function of the machine. That is, there is a corridor from ID_1 to room ID_2 if, with input w, the machine can in one step change its configuration from ID_1 to ID_2. The initial instantaneous description of the machine is designated the start room. Those instantaneous descriptions which include an accepting state are designated goal rooms. The machine will accept the input w if and only if the corresponding maze is threadable. Using Theorem 1 and this correspondence between non-deterministic machines and mazes, we obtain the following:

Theorem 3. Suppose a set A is accepted by a non-deterministic Turing machine, Z_N, within storage $L(n) \geq \log_2 n$. Then A is accepted by a deterministic Turing machine, Z_D, within storage $(L(n))^2$.

Given an input w, Z_D mimics Z_M to see if the maze corresponding to Z_N and w is threadable. Z_M is the deterministic machine of Theorem 1 which recognizes threadable mazes.

Setting $L(n) = n$ in Theorem 3, we get:

Corollary. Every context sensitive language is accepted by some deterministic Turing machine within storage $L(n) = n^2$.

If the $(\log_2 n)^2$ tape bound given in Theorem 1 could be reduced, we could obtain a corresponding reduction of the $(L(n))^2$ and n^2 bounds in Theorem 3 and its corollary. In particular, if the set of threadable mazes, M_Σ, could be recognized within deterministic storage $\log_2 n$, then every context sensitive language could be recognized by a deterministic linear bounded automaton.

An unsolved problem in the theory of tape complexity is whether there is some set A of strings and some function $L(n) \geq \log_2 n$ such that A is accepted by some non-deterministic Turing machine within storage $L(n)$ but accepted by no deterministic Turing machine within storage $L(n)$. Although we cannot offer a solution to this problem, we can show that: if any such A and $L(n)$ exist, then $A = M_\Sigma$ and $L(n) = \log_2 n$ will do. This is proven using Theorem 2 and the remarks in the previous paragraph.

References

1. P. M. Lewis II, R. E. Stearns, and J. Hartmanis, "Memory Bounds for the Recognition of Context Free and Context Sensitive Languages", IEEE Conference Record on 1965 Symposium on Switching Circuit Theory and Logical Design.

2. S. A. Cook and W. J. Savitch, "Mazes and Turing Machines", Technical Report #29, December 1968, Computer Center, University of California, Berkeley.