Nondetermism and Universal Turing Machines

CS254

Chris Pollett

Sep 13, 2006.

Outline

- Finish up RAMs
- Nondeterminism,
- Universal Turing Machines
- Undecidability

Simulating RAMs on TMs

Theorem If L is decided by a RAM in time f(n) then it is in TIME($O(f(n)^3)$).

Proof: Let P be a RAM program. We will simulate it by a seven tape machine. The first tape will be used to hold the input string and it will never be overwritten. The second tape will be used to represent the content of all the registers. This will be represented by a sequence of semicolon separated pairs i, v. Here i says the register (which may be 0) and v says its value. When a register is updated we copy the pair to the end of our sequence, update the value, then X over the old value. An example sequence might be: 0, 101; XXX 1, 10; _

The runtime for the result comes because this tape can be shown (see book) to grow as $O(f(n)^2)$.

The states of M are split into m groups where m in the number of instructions in P. Each group implements one instruction. Tape 3 is used to store the current program counter. This is initially 1. At the start of the simulation tape 2 is initialize to the input configuration of a register machine based on the contents of the input tape. Thereafter, at the start of simulating an instruction. The program counter is read and the start state of the group of states of M for that instruction is entered. (see next slide)

Proof Continued

An instruction is then processed, tape 2 is updated, and the program counter on tape 3 is updated, then the next step can be simulated and so on. Most instructions are reasonably straightforward to carry out: To process an instruction that uses indirect addressing of the form (j), tape 4 is used to store the value k of the register j so that we can then go access register k on tape 2. For operations like Add and Sub, tapes 5 and 6 are used to store the operands and tape 7 is used to compute the result. If the RAM halts, the contents of register 0 (the accumulator) are looked up on tape 2, and the TM accepts if the value is positive.

Nondeterministic Turing Machines

- Nondeterministic Turing Machines (NTMs) are defined by modifying the transition function so that now it is a map:

  δ: Q x Γ --> P(Q x $\Gamma \cup$(L, R)).

  Here P(A) is the power set of A -- the set of all subsets of A.

  For example, δ(q,a) ={(q',L), (q,b})} would mean in state q reading an a we can either move left and enter state q' or we can stay in state q and write a b.

- A **computation** is a sequence of configurations (s, $\underline{\#}$x) :- (q_1,$w_1\underline{a}v_1$) :- … :- (q_n,$w_n\underline{b}v_n$) such that each follows the previous according to one of the possible transitions given by the transition function. Due to the nondeterminism there might be many legal computations each starting from the same start configuration.
- The language of such a machine is the set of strings x such that there is a computation on which it halts with a "yes" on x.
- A language L is in NTIME(f(n)), if there is a NTM, N, for L which for every string x of length n and for every computation path of N on x, the machine halts with either a "yes" or a "no" in fewer than f(n) steps.

Simulation of Nondeterministic Turing Machines

Theorem Any language decidable by an NTM is decidable by a 3-tape deterministic TM. Further, NTIME(f(n)) is contained in $\cup_{c>1}$TIME($c^{f(n)}$).

Proof: Let N be a NTM that recognizes some language. We will make a Turing Machine D to simulate N. The idea is to have D try all possible branches of N's computation. If D ever finds an accept state of N on any of the branches then D accepts. So that we don't get stuck on infinite branches we will do our simulation in an iterated deepening, breadth first manner. D will have three tapes: the input tape, a simulation tape, and an address tape. D operates as follows:

1. Initially the input tape has the input and the other two tapes are blank.
2. D copies the input tape to the simulation tape.
3. D then simulate N according to the nondeterministic choices on the address tape.
4. Let c be the finite maximum number of choices in any given state reading a given symbol for a next state.
5. On the address tape we are going to write strings over the alphabet 0, 1, 2…, c-1, starting with the empty string and proceeding in lexicographical order
6. If the address tape is blank, D checks to see if N immediately accepts. Otherwise, D writes a 0 on the address tape and simulates N one step using the first possible nondeterministic choice. If this doesn't accept. Then D writes a 1, erases the simulation tape, and simulate N according to the second nondeterministic choice. Once we have tried all single step computations, we then cycle over computations of length 2, etc.

Notice the runtime of this algorithm to simulate f(n) steps is

$O(1 + c + c^2 + \ldots + c^{f(n)}) = O(c^{f(n)+1}) = O(c^{f(n)})$ so will be in TIME ($c^{f(n)}$) by linear speed-up.

Universal Turing Machines

- It is natural to write Turing Machine programs as strings, say <M>.
- One might hope there is a decision procedure for the Halting problem:

  H={ <M,w> | Here <M,w> is a coding as a string for pair M, w where M is a TM and w is a string and M halts on w}
- This language is recursive enumerable. Consider:

  U=“ On input <M,w>, where M is a TM and w is a string:

  1. Simulate M on input w.
  2. If M ever enters ia halt state, halt the same way; otherwise keep running”
- The above Turing Machine is called a **Universal Turing Machine (a UTM)** because it can be used to simulate any other Turing machine.
- However, as U on a given input does not necessarily halt, it is not a decision procedure for H.
- It turns out it is impossible to get a decision procedure for H.