CSE2315 Slides week 4

Matthijs Spaan   Stefan Hugtenburg

Algorithmics group

Delft University of Technology

13 March 2020

This lecture

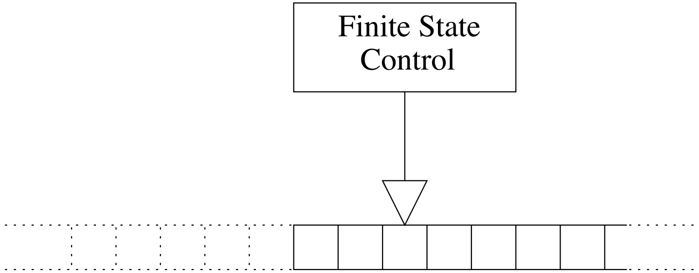
Introduction Turing machines

- Configuration of a TM
- Turing recognizability and decidability
- Descriptions of TMs

Variants of TMs

- Multi-tape TM
- Equivalence TMs
- Nondeterministic TMs
- Comparison computing power DTMs en NDTMs
- König’s Lemma
- Example NDTMs
- Differences between DTMs en NDTMs

Schematic Turing machine



Other literature:
tape also unbounded on the left

Sipser: tape bounded on the left

Formal definition Turing machine (Def. 3.3)

(See also video Introduction to Turing Machines and Definition of TMs and Related Language Classes)

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where:

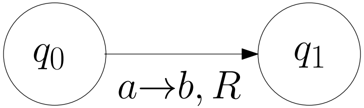
- Q is a finite set of states;
- Σ, the input alphabet, is a finite set not containing the 'blank' symbol $\sqcup$;
- Γ, the tape alphabet, is a finite set, with $\Sigma \subseteq \Gamma$ and $\sqcup \in \Gamma$.
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function.
- $q_0, q_{accept}, q_{reject} \in Q$ are the start state, the accept state and the reject state, respectively, with $q_{accept} \neq q_{reject}$.

The transition function δ

$\delta(q, a) = (r, b, L)$ means: if the TM is in state q and reads an a, then it overwrites it with a b, transitions to state r and shifts the read/write head one position to the left (L).

δ is usually displayed in the form of a transition diagram (for examples, see Sipser pp. 172–3).

Turing machine diagram


$$\begin{tikzcd} q_0 \arrow[r, "a \rightarrow b,\, R"'] & q_1 \end{tikzcd}$$

Denotes $\delta(q_0, a) = (q_1, b, R)$.

Other possibility for labels on arrows: $a, b \rightarrow L$.

Example TM

Let $\Sigma = \{a, b, c\}$ and

$$L = \{w \in \Sigma^* \mid w \text{ starts with an } a \text{ and ends in a } c\}.$$

Create a TM that “decides” L (diagram + definition)

For another example, see Turing Machine Examples .

Configurations

A configuration of a TM is determined by:

- its state,
- its tape contents, and
- the position of its read/write head.

Configurations are written as: $u\,q\,v$.

This means the TM is in state q, the tape contents are uv and the head is on the leftmost symbol of v.

(Sipser pp. 168–9)

Yielding

Given a TM M with transition function δ.

We say a configuration $ua\,q_i\,bv$ yields the configuration $u\,q_j\,acv$ if $\delta(q_i, b) = (q_j, c, L)$.

Similarly, a configuration $ua\,q_i\,bv$ yields the configuration $uac\,q_j\,v$ if $\delta(q_i, b) = (q_j, c, R)$.

(Sipser p. 169)

Special configurations

A start configuration is a configuration in which the state is q_0.

An accepting (rejecting) configuration is a configuration in which the state is q_{accept} (q_{reject}). These are the halting configurations.

Question: Is it possible for a TM never to arrive in a halting state for some input?

(Sipser p. 169)

Accepting TM

A TM M accepts a word w iff there exists a sequence of configurations $C_1, C_2, \ldots, C_k$ such that:

- C_1 is a start configuration;
- each C_i yields C_{i+1} $(1 \leq i < k)$; and
- C_k is an accepting configuration.

The language of M is the set of words accepted by M, written as $L(M)$.

(Sipser pp. 169–170)

Turing-recognizable (Def. 3.5)

A language L is Turing-recognizable (also known as recursively enumerable) if there exists a TM M such that $L(M) = L$.

Such a TM M is called a recognizer of L.

Note: It is possible for M never to arrive in a halting state for some input, so that it ends up looping.

Turing-decidable (Def. 3.6)

A language L is Turing-decidable, or simply decidable (also known as recursive), if there exists a TM M such that $L(M) = L$ and such that M arrives in a halting state for every input.

Such a TM M is called a decider of L.

Descriptions of TMs

- Formal description: all details in terms of δ or a diagram.
- Implementation description: describes head movements and how data is placed on the tape(s)*.
- *High-level* description: the main idea without implementation details.

* We will discuss variants of TMs with multiple tapes as well.

(Sipser p. 185)

Exercise

Let $\Sigma = \{a, b\}$ and

$$L = \{w \in \Sigma^* \mid n_a(w) = 2 \cdot n_b(w)\},$$

where $n_a(w)$ and $n_b(w)$ denote the numbers of as and bs in w, respectively.

Give an implementation description of a TM that decides L.

See also video Turing Machine Programming Techniques for some techniques you can use when programming TMs.

Variants of TMs

- Multi-tape TMs: TMs with multiple tapes
- Nondeterministic TMs (NDTMs)
- other variants: see exercises.

Multitape TMs

A TM can have more than one tape, in which case we call it a multitape TM.

In this case, the signature of the transition function is $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$.

$$\delta(q_i, a_1, a_2, \ldots, a_k) = (q_j, b_1, b_2, \ldots, b_k, L, R, \ldots, S)$$

then relates to k simultaneous actions performed on the k tapes, transitioning from state q_i to q_j.

(Sipser p. 176; see also video Multitape Turing Machines)

Exercise

Give an implementation description of a TM that decides the following language P over alphabet $\Sigma = \{a, b\}$:

$$P = \{w \in \Sigma^* \mid w = w^R\}.$$

How would you do this using a single tape TM?

Equivalence TMs

Two TMs M_1 and M_2 are equivalent if they recognize the same language, i.e., $L(M_1) = L(M_2)$.

(Sipser p. 54)

Exercise

Suppose we have equivalent TMs M_1 and M_2.
Does it follow that M_1 and M_2 decide the same language?

Equivalence 1-TMs and k-TMs (Th. 3.13)

A k-tape TM can be simulated by a standard single tape TM (in no more than quadratic time relative to the simulated TM).

k-TMs and recognizability (Cor. 3.15)

A language L is Turing-recognizable iff there exists a k-TM M such that $L(M) = L$.

Similarly: A language L is Turing-decidable iff there exists a k-TM M such that $L(M) = L$ and such that M arrives in a halting configuration for every input.

Power set $\mathscr{P}(V)$

$\mathscr{P}$ is the power set-operator.

If V is a set, then:

$$\mathscr{P}(V) = \{X \mid X \subseteq V\}.$$

In other words, $\mathscr{P}(V)$ is the set consisting of all subsets of V.

Nondeterministic TMs

A nondeterministic TM is defined the same way as an ordinary TM, except that the transition function differs:

$$\delta : Q \times \Gamma \to \mathscr{P}(Q \times \Gamma \times \{L, R\}).$$

(Sipser p. 178; see also video Nondeterminism in Turing Machines)

Example

If

$$\delta(q_1, a) = \{(q_3, b, R), (q_4, c, L), (q_3, a, L)\},$$

then the relevant TM (which is in state q_1 and is reading an a) can perform three possible actions:

1. transition to state q_3, change a into b and shift the head right, or
2. transition to state q_4, change a into c and shift the head left, or
3. transition to state q_3, leave the a as it is and shift the head left.

How does such an NDTM “compute”?

This means the example NDTM has three possible subsequent configurations when the current one is $u\,q_1\,av$.

What does that mean?

Three ways to look at it:

- The NDTM “chooses” (nondeterministically) a random next step.
- The NDTM “chooses” the correct next step every time.
- The NDTM takes all possible next steps in parallel.

Computation of a nondeterministic TM (NDTM)

A computation performed by an NDTM (given an input) can be represented as a tree of which the nodes form configurations and the branches transitions. The root of this tree represents the start configuration and for each node that does not represent a halting configuration, its child nodes are all possible configurations that can be yielded by the configuration represented by that node.

Acceptation by NDTMs

This is defined the same way as for ordinary TMs:

An NDTM M accepts the word w iff there exists a sequence of configurations $C_1, C_2, \dots, C_k$ such that:

- C_1 is a start configuration;
- every C_i yields C_{i+1} $(1 \leq i < k)$; and
- C_k an accepting configuration.

So a word is accepted by an NDTM if there is an ‘accepting path’ in the computation tree, i.e., a path with an accepting configuration at its end.

Paths in computation trees

Given an M and an input word w, there are three possibilities for paths in the corresponding computation tree:

- ending in an accepting state — $w \in L(M)$;
- ending in a rejecting state — such a path gives no information, so both $w \in L(M)$ and $w \notin L(M)$ are possible;
- not ending (infinite branch) — gives no information.

Moral:

- false negatives are permitted,
- false positives, on the other hand, are not!

Language of an NDTM

Let M be an NDTM. The language $L(M)$ of M is the set

$$L(M) = \{w \mid M \text{ accepts } w\}.$$

We say M is a nondeterministic recognizer of L.

Example NDTM

Let $\Sigma = \{a, b, c\}$. Give a high-level description of an NDTM recognizing the following language:

$$L = \{w \in \Sigma^* \mid \text{there exists an } x \in \Sigma \text{ such that } n_x(w) \geq 7\}.$$

Note: $n_x(w)$ stands for the number of occurrences of x in w, where x is an element of Σ.

Solution

A high-level description for a TM M recognizing L is given by:

$M =$ "On input $w \in \Sigma^*$:

1. Pick a letter a, b or c.
2. Sweep across the tape from left to right and determine whether the chosen letter occurs at least seven times in w.
   If so: accept. If not: reject."

Equivalence of TMs and NDTMs (Th. 3.16)

For every nondeterministic TM, an equivalent deterministic TM exists.

NDTMs and recognizability (Cor. 3.18)

A language L is Turing-recognizable iff there exists an NDTM M such that $L(M) = L$.

Decision by an NDTM

An NDTM M is a nondeterministic decider for a language L iff $L(M) = L$ and for all input words w, all possible paths in the computation tree for M and w end in a halting configuration.

(Sipser p. 154)

NDTMs and decidability (Cor. 3.19)

A language L is Turing-decidable there exists a nondeterministic decider for L.

Theorem needed: König's Lemma

A tree with a finite branching factor and an infinite number of nodes contains an infinitely long path.

Contrapositive: A tree with a finite branching factor that contains no infinite paths, has a finite number of nodes.

Exercise

Let M be a deterministic TM recognizing L. Which of the following statements are correct (if any)?

- If $w \in L$, then M halts on input w in the accept state.
- If $w \notin L$, M does not halt on w.
- If $w \in L$, then M can loop on w.
- If $w \notin L$, then M halts in the reject state.

Solution

The statements are:

- Correct, the TM M must accept the input w.
- Incorrect, M might halt on w.
- Incorrect, the TM must accept at all times in this case.
- Incorrect, the TM recognizes the language L but we do not know whether it also decides this language. M can therefore also loop on w.

Exercise

Let M be an NDTM recognizing L. Which of the following statements are correct (if any)?

- If $w \in L$, then M always halts on w in the accept state.
- If $w \notin L$, then M may halt on w in the accept state.
- If $w \in L$, then M may halt on w in the reject state.
- If $w \notin L$, then M halts in the reject state.
- If $w \in L$, then M may loop on w.
- If $w \notin L$, then M may loop on w.

Solution

The statements are:

- Incorrect, since M may have made a wrong choice. There needs to be only one path in the computation tree resulting in the acceptance of w.
- Incorrect, since then $w \in L$, which would form a contradiction.
- Correct, this is possible (it would be a *false negative*).
- Incorrect, since M can loop on w.
- Correct (see also the first statement).
- Correct (this is possible just as with a deterministic TM).

NDTMs and recognizability

Let M be an NDTM recognizing L.

For words $w \in L$, there must exist at least one accepting path in the computation tree. Such a path is finite. Other paths may reject w or be infinitely long.

For words $w \notin L$, there cannot be an accepting path in the computation tree.

Exercise

Let M be an NDTM deciding L. Which of the following statements are correct (if any)?

- If $w \in L$, then M always halts on w in the accept state.
- If $w \notin L$, then M may halt on w in the accept state.
- If $w \in L$, then M may halt on w in the reject state.
- If $w \notin L$, then M halts in the reject state.
- If $w \in L$, then M may loop on w.
- If $w \notin L$, then M always halts on w in the reject state.

Solution

The statements are:

- Incorrect, since M may have made a wrong choice (nondeterminism).
- Incorrect, since otherwise we would get $w \in L$.
- Correct, there must exist at least one path through which M accepts input w. With other paths/choices, M may halt on w in the reject state.
- Correct, M is a decider and cannot loop.
- Incorrect, since M is a decider and cannot loop.
- Correct, since M is a decider and cannot loop.

Decidability and NDTMs

Let M be an NDTM deciding a language L.

For words $w \in L$, at least one accepting path must exist in the computation tree. All paths for w are finite, but not all of them have to end in the accept state.

For words $w \notin L$, all paths in the computation tree must end in the reject state.

Exercise

Let M be a nondeterministic recognizer for L. Suppose for a given word w, all computation paths in the tree for M and w end in the reject state. Which of the following statements are correct (if any)?

- $w \in L$;
- $w \notin L$;
- we cannot say anything about this.

Example NDTM

Let $\Sigma = \{1\}$. Give a high-level description of an NDTM deciding the language $L \subseteq \Sigma^*$, where

$$L = \{1^n \mid \text{there exists an } m \in \mathbb{N} \text{ with } m > 1 \text{ and } n \bmod m^2 = 0\}.$$

Solution

A high-level description of a TM M with 2 auxiliary tapes deciding L is given by:

$M =$ "On input $w \in \Sigma^*$:

1. Write a 1 on auxiliary tape 1.
2. Write a 1 on auxiliary tape 1. Let m be the number of symbols on auxiliary tape 1. If $m \leq n$: either repeat step 2 or go to step 3. Otherwise, go to step 3.
3. Write a number m^2 of 1s to auxiliary tape 2.
4. Determine whether the contents of auxiliary tape 2 fit an integer number of times into the input word.
   If so: <u>accept</u>. If not: <u>reject</u>."

Constructing NDTMs

Phase 1: Guess a solution.
Phase 2: Check the solution: if it suffices, go to q_{accept}, otherwise go to q_{reject}.

Exercise

Suppose we have the following language $L \subseteq \{a, b, c\}^*$:

$$L = \{uvw \mid |v| > 0 \text{ and } n_a(v) + n_b(v) = n_c(v)\}.$$

Give a high-level description for a nondeterministic recognizer for L that scans its input at most twice and that uses no more than two auxiliary tapes.
This Turing machine must be “truly” nondeterministic. That is to say, the machine must not be a deterministic Turing machine.

Solution

A high-level description for an NDTM N with two auxiliary tapes deciding L is given by:

$N =$ "On input $x = uvw \in \Sigma^*$:

1. Sweep from left to right across the input and guess where v starts.
2. Sweep from left to right across the input, copy as and bs to auxiliary tape 1 and cs to auxiliary tape 2. Guess where v ends and go to step 3.
3. Determine whether auxiliary tape 1 and 2 contain the same number of symbols.
   If so: accept. If not: reject."