CS112: Theory of Computation (LFA)

Lecture11: Turing Machines

Dumitru Bogdan

Faculty of Computer Science
University of Bucharest

May 6, 2021

Table of contents

Section 1

Admin stuff

Examn related

- Exam date: 18 June at 8 AM
- Exam type: 24h
- Exam problems: one theory related, one practical
- Reexam date: TBA

Section 2

Previously on CS112

PDA

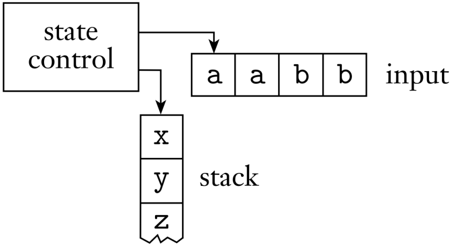

Figure: Schematic of a pushdown automaton

- A **pushdown automaton** (PDA) can write symbols on the stack and read them back later
- Writing a symbol "pushes down" all the other symbols on the stack.
- At any time the symbol on the top of the stack can be read and removed. The remaining symbols then move back up
- Writing a symbol on the stack is often referred to as pushing the symbol, and removing a symbol is referred to as popping it

PDA

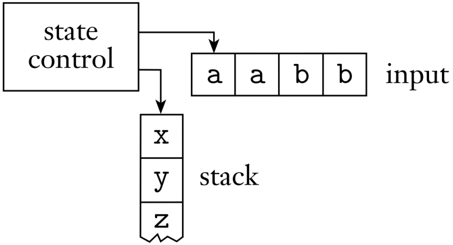

Figure: Schematic of a pushdown automaton

- Note that all access to the stack, for both reading and writing, may be done only at the top. In other words a stack is a "last in, first out" storage device
- If certain information is written on the stack and additional information is written afterward, the earlier information becomes inaccessible until the later information is removed
- A stack is valuable because it can hold an unlimited amount of information

PDA

Definition

A pushdown automaton is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q, Σ, Γ and F are all finite sets, and

1. Q is the set of states
2. Σ is the input alphabet
3. Γ is the stack alphabet
4. $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ is the transition function
5. $q_0 \in Q$ is the start state
6. $F \in Q$ is the set of accept states

PDA

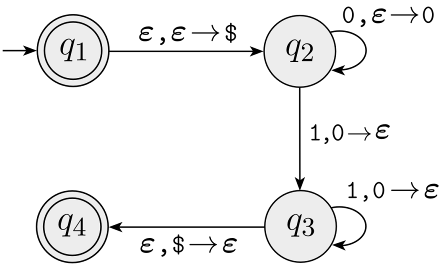

Figure: State diagram for the PDA M_1

- We can use a state diagram to describe a PDA
- Such diagrams are similar to the state diagrams used to describe finite automata, modified to show how the PDA uses its stack when going from state to state
- We write $a, b \to c$ to signify that when the machine is reading an a from the input, it may replace the symbol b on the top of the stack with c

Section 3

Context setup

Context setup

Corresponding to Sipser 3.1

Context setup

- We study a much more powerful model, first proposed by Alan Turing in 1936, called the **Turing machine**
- Similar to a finite automaton but with an unlimited and unrestricted memory
- A Turing machine is a much more accurate model of a general purpose computer
- **A Turing machine can do everything that a real computer can do**
- **However, even a Turing machine cannot solve certain problems**
- Because, in a very real sense, these problems are beyond the theoretical limits of computation

Context setup

- Turing is today one of the most celebrated figures of computer science
- The father of computer science and the father of the modern computer
- I think it is fair to say that the concept of Turing machines has revolutionised the world
- Re-watch *The Imitation Game*

Section 4

Intuition

How it looks

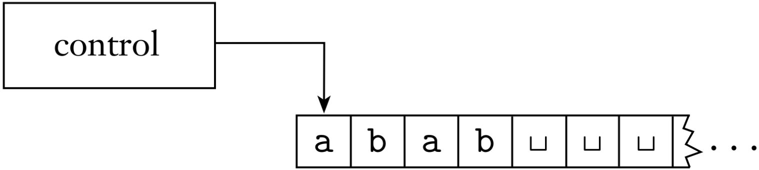

Figure: Schematic of a Turing machine

How it works

- The Turing machine (TM) model uses an infinite tape as its unlimited memory
- It has a tape head that can read and write symbols and move around on the tape
- Initially the tape contains only the input string and is blank everywhere else
- If the machine needs to store information, it may write this information on the tape
- To read the information that it has written, the machine can move its head back over it
- The machine continues computing until it decides to produce an output
- The outputs accept and reject are obtained by entering designated accepting and rejecting states
- If it doesn't enter an accepting or a rejecting state, it will go on forever, never halting

Differences between TM and FA

1. A Turing machine can both write on the tape and read from it
2. The read–write head can move both to the left and to the right
3. The tape is infinite
4. The special states for rejecting and accepting take effect immediately

Example

- We want to create M_1 for testing membership in the language $B = \{w\#w | w \in \{0,1\}^*\}$
- We want M_1 to accept if its input is a member of B and to reject otherwise
- Assume a very long input string, too long to be remembered, but you are allowed to move back and forth over the input and make marks on it
- The obvious strategy is to zig-zag to the corresponding places on the two sides of the $\#$ and determine whether they match
- It makes multiple passes over the input string with the read–write head
- On each pass it matches one of the characters on each side of the $\#$ symbol
- To keep track of which symbols have been checked already, M_1 crosses off each symbol as it is examined
- If it crosses off all the symbols, that means that everything matched successfully, and M_1 goes into an accept state
- If it discovers a mismatch, it enters a reject state

Example

$M_1 =$ "On input string w:

1. Zig-zag across the tape to corresponding positions on either side of the $\#$ symbol to check whether these positions contain the same symbol. If they do not, or if no $\#$ is found, reject. Cross off symbols as they are checked to keep track of which symbols correspond
2. When all symbols to the left of the $\#$ have been crossed off, check for any remaining symbols to the right of the $\#$. If any symbols remain, reject; otherwise, accept"

Example

The below figure contains several nonconsecutive snapshots of M_1's tape after it is started on input 011000#011000:

```
0 1 1 0 0 0 # 0 1 1 0 0 0 ⊔ ...
x 1 1 0 0 0 # 0 1 1 0 0 0 ⊔ ...
x 1 1 0 0 0 # x 1 1 0 0 0 ⊔ ...
x 1 1 0 0 0 # x 1 1 0 0 0 ⊔ ...
x x 1 0 0 0 # x 1 1 0 0 0 ⊔ ...
x x x x x x # x x x x x x ⊔ ...
                          accept
```

Figure: Snapshots of Turing machine M_1

Section 5

Formal definition

Delta function

- The heart of the definition of a TM is the transition function δ because it tells us how the machine gets from one step to the next
- For a TM δ takes the form $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
- When the TM is in a certain state q and the head is over a tape square containing a symbol a, and if $\delta(q, a) = (r, b, L)$, the machine writes the symbol b replacing the a, and goes to state r
- The third component is either L or R and indicates whether the head moves to the left or right after writing. In this case, the L indicates a move to the left.

Formal definition

Definition

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where Q, Σ, Γ are all finite sets and

1. Q is the set of states
2. Σ is the input alphabet not containing the *blank symbol* $\sqcup$
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function
5. $q_0 \in Q$ is the start state
6. $q_{accept} \in Q$ is the accept state
7. $q_{reject} \in Q$ is the reject state and $q_{accept} \neq q_{reject}$

TM computation

A TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ computes as follows:

- Initially, M receives its input $w = w_1 w_2 ... w_n \in \Sigma^*$ on the leftmost n squares of the tape, and the rest of the tape is blank (i.e., filled with blank symbols)
- The head starts on the leftmost square of the tape. Note that Σ does not contain the blank symbol, so the first blank appearing on the tape marks the end of the input
- Once M has started, the computation proceeds according to the rules described by the transition function
- If M ever tries to move its head to the left off the left-hand end of the tape, the head stays in the same place for that move, even though the transition function indicates L
- The computation continues until it enters either the accept or reject states, at which point it halts. If neither occurs, M goes on forever.

TM computation

- As a TM computes, changes occur in the current state, the current tape contents, and the current head location
- A setting of these three items is called a *configuration* of the TM. Configurations often are represented in a special way
- For a state q and two strings u and v over the tape alphabet Γ, we write uqv for the configuration where the current state is q, the current tape contents is uv, and the current head location is the first symbol of v. The tape contains only blanks following the last symbol of v
- For example, $1011q_701111$ represents the configuration when the tape is 101101111, the current state is q_7, and the head is currently on the second 0

TM computation

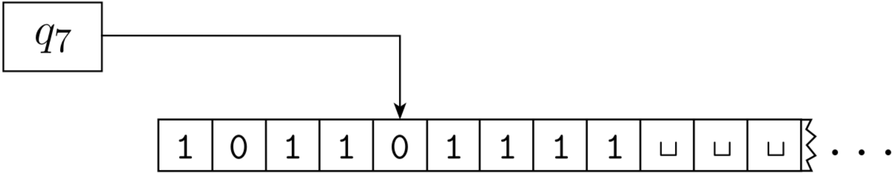

Figure: A TM with configuration $1011q_701111$

TM computation

Now we formalize our intuitive understanding of the way that a TM computes

- Say that configuration C_1 yields configuration C_2 if the TM can legally go from C_1 to C_2 in a single step
- Suppose that we have $a, b, c \in \Gamma$ and $u, v \in \Gamma^*$ and states q_i and q_j. In that case uaq_ibv and uq_jacv are two configurations
- We say that uaq_ibv yields uq_jacv if the transition function $\delta(q_i, b) = (q_j, c, L)$. That handles the case where the TM moves leftward
- For a rightward move, say that uaq_ibv yields uq_jacv if the transition function $\delta(q_i, b) = (q_j, c, R)$
- Special cases occur when the head is at one of the ends of the configuration

TM computation

- The start configuration of M on input w is the configuration $q_0 w$, which indicates that the machine is in the start state q_0 with its head at the leftmost position on the tape
- In an accepting configuration, the state of the configuration is q_{accept}
- In a rejecting configuration, the state of the configuration is q_{reject}
- Accepting and rejecting configurations are *halting configurations* and do not yield further configurations
- A TM M accepts input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where
    1. C_1 is the start configuration of M on input w
    2. each C_i yields C_{i+1}
    3. C_k is an accepting configuration

Recognizable and decidable

- The collection of strings that M accepts is the language of M or the language recognized by M, denoted $L(M)$
- When we start a TM on an input, three outcomes are possible. The machine may accept, reject, or loop. By loop we mean that the machine simply does not halt
- M can fail to accept an input by entering the q_{reject} state and rejecting, or by looping. Sometimes distinguishing a machine that is looping from one that is merely taking a long time is difficult. For this reason, we prefer Turing machines that halt on all inputs; such machines never loop.
- These machines are called *deciders* because they always make a decision to accept or reject. A decider that recognizes some language also is said to decide that language

Recognizable and decidable

Definition

Call a language Turing-recognizable if some Turing machine recognizes it

Definition

Call a language Turing-decidable or simply decidable if some Turing machine decides it

Section 6

Examples

Examples

- Next, we study examples of decidable languages
- Every decidable language is Turing-recognizable
- As we did for finite and PDA, we can formally describe a particular TM by specifying each of its seven parts
- However, going to that level of detail can be cumbersome for almost all TM
- So, we will give only higher level descriptions because they are precise enough for our purposes and are much easier to understand

Example I

We describe M_2 that decides $A = \{0^{2^n} | n \geq 0\}$ the language consisting of all strings of 0s whose length is a power of 2 $M_2 =$ "On input string w:

1. Sweep left to right across the tape, crossing off every other 0
2. If in stage 1 the tape contained a single 0, *accept*
3. If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, *reject*
4. Return the head to the left-hand end of the tape
5. Go to stage 1"

Example I

- Each iteration of stage 1 cuts the number of 0s in half
- As the machine sweeps across the tape in stage 1, it keeps track of whether the number of 0s seen is even or odd
- If that number is odd and greater than 1, the original number of 0s in the input could not have been a power of 2
- Therefore, the machine rejects in this instance. However, if the number of 0s seen is 1, the original number must have been a power of 2. So in this case, the machine accepts

Example I

Formal description of $M_2 = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$
- $\Sigma = \{0\}$
- $\Gamma = \{0, x, \sqcup\}$
- The start, accept and reject states are q_1, q_{accept}, and q_{reject}, respectively.
- δ is described in the next figure

Example I

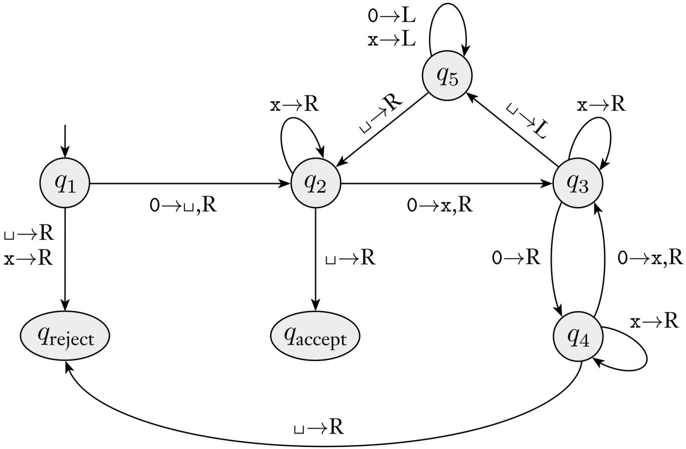

Figure: State diagram for Turing machine M_2

Example I

- The label $0 \rightarrow \sqcup, R$ appears on the transition from q_1 to q_2
- This label signifies that when in state q_1 with the head reading 0, the machine goes to state q_2, writes $\sqcup$ and moves the head to the right
- In terms of δ function we have: $\delta(q_1, 0) = (q_2, \sqcup, R)$
- For clarity we use the shorthand $0 \rightarrow R$ in the transition from q_3 to q_4, to mean that the machine moves to the right when reading 0 in state q_3 but doesn't alter the tape, so $\delta(q_3, 0) = (q_4, 0, R)$
- This machine begins by writing a blank symbol over the leftmost 0 on the tape so that it can find the left-hand end of the tape in stage 4

Example I

Below we have a sample run of this machine on input 0000. The starting configuration is $q_1 0000$. The sequence of configurations the machine enters appears as follows; read down the columns and left to right

$q_1 0000$	$\sqcup q_5 \mathtt{x}0\mathtt{x}\sqcup$	$\sqcup \mathtt{x} q_5 \mathtt{xx}\sqcup$
$\sqcup q_2 000$	$q_5 \sqcup \mathtt{x}0\mathtt{x}\sqcup$	$\sqcup q_5 \mathtt{xxx}\sqcup$
$\sqcup \mathtt{x} q_3 00$	$\sqcup q_2 \mathtt{x}0\mathtt{x}\sqcup$	$q_5 \sqcup \mathtt{xxx}\sqcup$
$\sqcup \mathtt{x}0 q_4 0$	$\sqcup \mathtt{x} q_2 0\mathtt{x}\sqcup$	$\sqcup q_2 \mathtt{xxx}\sqcup$
$\sqcup \mathtt{x}0\mathtt{x} q_3 \sqcup$	$\sqcup \mathtt{xx} q_3 \mathtt{x}\sqcup$	$\sqcup \mathtt{x} q_2 \mathtt{xx}\sqcup$
$\sqcup \mathtt{x}0 q_5 \mathtt{x}\sqcup$	$\sqcup \mathtt{xxx} q_3 \sqcup$	$\sqcup \mathtt{xx} q_2 \mathtt{x}\sqcup$
$\sqcup \mathtt{x} q_5 0\mathtt{x}\sqcup$	$\sqcup \mathtt{xx} q_5 \mathtt{x}\sqcup$	$\sqcup \mathtt{xxx} q_2 \sqcup$
		$\sqcup \mathtt{xxx}\sqcup q_{\mathrm{accept}}$

Section 7

Variants of Turing Machines

Variants of Turing Machines

See Sipser 3.2 :)