Institut for Matematik og Datalogi
Syddansk Universitet

February 15, 2024
JBJ

DM553 – Spring 2024– Weekly Note 4

First set of exam problems
These are available from itslearning and the homepage of the couse from February 15.
Key points:

- A Turing machine (TM) is an automaton whose head can move both left and right. Furthermore, a TM may alter its tape which is one-way infinite.
- TM's are much more powerful than DFA's and PDA's. In fact, a TM is generally accepted to be as powerful as modern computers, as hypothesized by the Church-Turing thesis.
- TM's provide insight into the theoretical limits of computation, i.e. what computers can do and how fast they can do it.
- A TM M **recognizes** a language L iff every string in L leads M to its accept state. Strings not in L will either lead to the reject state or M will loop forever. $L(M)$ denotes the language recognized by M. Note that **every TM recognizes exactly one language**, namely the set of strings started upon which it halts in its accept state!
- A language L is called **recognizable** if there is some TM M that recognizes L. The class of recognizable languages is set of all the languages which are recognized by some TM.
- A TM M **decides** a language L iff every string in L leads M to its accept state and every string not in L leads M to its reject state. In particular, M halts on every string!
- A **decider** is a TM that always halts (and thus ends in either its accept state or in its reject state). **Every TM which is a decider decides exactly one language** namely the set of strings started upon which it halts in its accept state!
- A language L is called **decidable** if there is some TM M that decides L. The class of decidable languages is set of all the languages which are decided by some TM.
- If a language is decidable, then it is also recognizable but the opposite is not always true as we shall see soon!
- There are different ways to specify a TM: transition table, pseudo code, state diagram, and high level description.

- Terminology:

  Different books use different names for the same concept. In old exam problems you may find names different from the ones used in the book.

  - Turing computable (Turing beregneligt) = decidable (afgørligt) = recursive (rekursivt)
  - Partial Turing computable (partielt Turing beregneligt) = semidecidable (semi-afgørligt) = recursively enumerable (rekursivt enumerabelt) = Turing enumerable (Turing enumerabelt) = recognizable

- I described a couple of elementary TM's which can be used as subroutines These are

  - Shift right (S_R) which starts in the configuration q_0w and ends in the configuration $q_{accept} \rhd w$.
  - Shift left (S_L) which starts in the configuration $q_0\#w$ and ends in the configuration $q_{accept}w$. Here # is some symbol not in the alphabet of the language L that we work on.

  These are just a few examples of useful subroutines one can easily make. There are more in the exercises below. Note that given such subroutines we can now start building more complicated ones by concattenating these or using them as "states" in diagrams. By this we mean the following: Suppose M_1, M_2, M_3 are different TMs with the same input alphabet. Then by $M_1M_2M_3$ we mean the TM that starts in the stating state of M_1 and runs M_1. Then when M_1 would stop in its accepting state, we start M_2 and run it etc. We can also make arrows like this $M_1 \overset{a}{\to} M_2$ which is supposed to mean that we first run M_1 and then if it stops in its accepting state **and** the head is reading the symbol a, then we start M_2 with its head in the current position. Similarly we could write $M_1 \overset{\neq a}{\to} M_3$ to mean that when M_1 stops we start M_3 provided that the character current read is not 'a'. Finally $M_1 \to M_2$ means the TM that starts as M_1 and then, if M_1 finishes in its accepting state, starts M_2 from the tape position that M_1 stopped in. Now you should be able to continue this idea and draw diagrams of fairly complicated TMs starting from simple TMs as building blocks.

Lectures in Week 8:
These will be on Section 3.2 on various extensions of the standard Turing machine. I will probably also start on Section 3.3. The relevant videos are Videos 8, 9 and some of Video 10.

Exercises in Week 8

- 3.2 (b)-(d) on page 187 (same in 3rd edition)

$$\begin{tikzcd}
{} \arrow[r] & R \arrow[r, "a"] & \sqcup \arrow[l, bend right=60]
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
{} \arrow[r] & R_{\sqcup}L \arrow[r, "a \neq \sqcup"] \arrow[d, "a = \sqcup"] & \sqcup RaL \arrow[l, bend right=60] \\
& \rhd &
\end{tikzcd}$$

(b)

Figure 1: In (a) the TM moves right and the erases 'a's until a symbol different from 'a' is read. In (b) we show a diagram of S_R (the right shifting machine). Note that here we use the notation that the TM remembers a after reading it so that it can write it later (one position to the right). The Turing machine $R_{\sqcup}$ moves right until it finds a blank to the right of its starting position (so if it started on a blank, it will still move one step right). You will study how to find the building blocks (small TMs) in the exercises below.

- 3.7 and 3.8 on page 188 (same in 3rd edition)
- Describe a Turing machine for deciding the language $L = \{a^n b^{n!} | n \geq 1\}$. Here $n! = n \cdot (n-1) \cdot \ldots \cdot 1$.
- Exam 2002 problem 5.
- This exercise is about constructing some simple Turing machines. For each of the following you should explain how to realize TMs with excatly that property. Once you have constructed a TM you may use it as a building block for constructing other TMs. Note that we do not assume that these TMs are started on the leftmost cell of the tape!
  - The TM R that just moves its head one position to the right from where it is started and stops.
  - The TM L that just moves its head one position to the left from where it is started and stops. Note that here you need to say what L will do if is it started on the leftmost tape cell.
  - The TM R_a, where $a \in \Gamma$, for some alphabet Γ. R_a will move to the right until it finds the symbol 'a' and then it will stop. Note that if there is no 'a' to the right of the current head position, then R_a will never stop. Note that R_a always moves at least one step to the right!
  - The TM $R_{\neq a}$, where $a \in \Gamma$, for some alphabet Γ. $R_{\neq a}$ will move to the right until it finds a symbol that is different from 'a'. Again $R_{\neq a}$ will always move at least one step to the right.

- The left moving equivalents L_a and $L_{\neq a}$ of R_a and $R_{\neq a}$.
- The TM a, where $a \in \Gamma$, for some alphabet Γ. This TM simply writes the symbol 'a' and stops (gives over control to the next TM in some sequence that we are building when we use this as a subroutine).

Here are some examples of what you can do with these simple machines:

- Construct a TM M that decides the language $\{a^n b^n c^n | n \geq 0\}$. You should explain important steps of the TM and produce a diagram of M as composed by simple TMs ala the ones described above.
- Make a TM diagram for the TM C (Copy). Recall that Copy starts in the configuration $q_o w$ and ends in the configuration $q_{accept} ww$.
- Describe a Turing machine M which given a string $w = w_1 w_2 \ldots w_n$ returns the string w_{odd} where $w_{odd} = w_1 w_3 \ldots w_{n-2} w_n$ if n is odd and $w_{odd} = w_1 w_3 \ldots w_{n-3} w_{n-1}$ if n is even. That is, M starts in the configuration $q_0 w$ and ends in the configuration $q_{accept} w_{odd}$.