Xi'an Jiaotong-Liverpool University
西交利物浦大學

Int 201: Decision Computation and Language
Tutorial 9 Solution

Dr. Chunchuan Lyu

November 29, 2023

Question 1. Show step by step processing of input string 01#01 on the Turing Machine M in Figure1 , using configuration descriptions. i.e., start with $q_1$01#01. Note that $x \to R$ means, read x and move right without writing. Also, rejection state is not explicitly stated, but if there is no valid transition, the machine rejects.

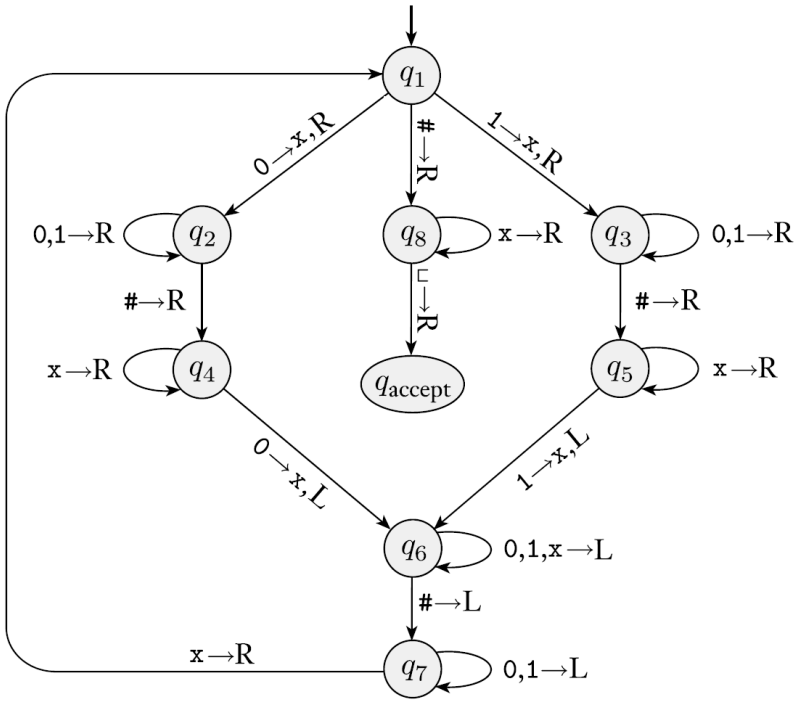

Figure 1: Turing Machine M

Solution 1. 1. $q_1$01#01

2. $xq_2 1\#01$
3. $x1q_2\#01$
4. $x1\#q_4 01$
5. $x1q_6\#x1$
6. $xq_7 1\#x1$
7. $q_7 x1\#x1$
8. $xq_1 1\#x1$
9. $xxq_3\#x1$
10. $xx\#q_5 x1$
11. $xx\#xq_5 1$
12. $xx\#q_6 xx$
13. $xxq_6\#xx$
14. $xq_7 x\#xx$
15. $xxq_1\#xx$
16. $xx\#q_8 xx$
17. $xx\#xq_8 x$
18. $xx\#xxq_8$
19. $xx\#xxq_{accept}$

Question 2. Write down the set description of the language being accepted by the Turing Machine M in Figure1. Justify your answer.

Solution 2. $L = \{w\#w | w \in \{0,1\}^*\}$, the machine cross over 0s or 1s pair before and after the # sign, and accept the input if all strings before # matches strings after # and all inputs are exhausted.

Question 3. Is the Turing Machine M in Figure1 a decider (i.e., all inputs halt on M)? Justify your answer.

Solution 3. Yes, this TM is a decider, as the zigzag procedure always terminates for finite length input string. In fact, the machine rejects whenever there is a mismatch, therefore for any length $2n+1$ (if length not odd, it rejects sooner than the time spend on accepting $2n+3$), the longest running step happen when the string gets accepted. Total time for accepting $2n+1$ length-ed string is the sum of all the zigzag plus final checking all inputs being exhausted. So, it is $O(n^2)$.

Question 4. Given a context-free grammar CYK algorithm can find a valid parse tree for a given input in finite time or claiming there is no valid tree. Are all context-free languages are Turing-decidable ?

Solution 4. Yes, as running CYK algorithm takes finite time. A TM can simulate the running of CYK, and TM accepts the string iff CYK produce a parse. Therefore, all CFL are Turing-decidable. Alternatively, this is a corollary from lecture 10, where we built a more powerful device that handles all grammars and inputs at once.

Question 5.

$N =$ On input w

1. Check if $w \in a^*b^*c^*$, reject if not.
2. Count the number of a's,b's, and c's
3. Accept if all counts are equal; reject if not

Write down the set description of the language being accepted by the Turing Machine N of the above high-level descriptions.

Solution 5. $L = \{a^k b^k c^k | k \geq 0\}$

Question 6. A queue automaton is like a push-down automaton except that the stack is replaced by a queue. A queue is a tape allowing symbols to be written only on the left-hand end and read only at the right-hand end. Each write operation (we'll call it a push) adds a symbol to the left-hand end of the queue and each read operation (we'll call it a pull) reads and removes a symbol at the right-hand end. As with a PDA, the input is placed on a separate read-only input tape, and the head on the input tape can move only from left to right. The input tape contains a cell with a blank

symbol following the input, so that the end of the input can be detected. A queue automaton accepts its input by entering a special accept state at any time. **Show that a language can be recognized by a deterministic queue automaton iff the language is Turing-recognizable.**[1] In other words, explain in high-level description how to simulate a TM with a queue automaton and vice versa.

Solution 6. To simulate a TM with a queue automaton, the idea is to use the queue to store the tape of Turing machine. Then all token processing becomes ϵ, the real condition depends on the queue elements and the state.

Initial step In the beginning we push a H to the queue to mark the head, then we read all strings with the queue automaton, and push then to the queue one by one, and push an end symbol $\sqcup$ to the queue. $q_{start}x_1x_2x_3x_4x_5$ will have queue initialized as $Hx_1x_2x_3x_4x_5\sqcup$, where the left side is the first in.

Read To simulate the TM read a symbol, we pop and push all the symbols, until we find the H. Then, we pop another symbol, that will be the symbol we are reading. For this queue configuration, $x_1x_2x_3x_4x_5 \sqcup H$, the reading happens when we have $x_3x_4x_5 \sqcup x_1$ in the queue and read out x_2.

Transition to the right Then, to simulate a TM step with $x_2 \to x_2^{new}, R$. We push x_2^{new} into the queue, then we put H into the queue. We have the resulting queue as $x_3x_4x_5 \sqcup x_1x_2^{new}H$. Meanwhile, we change our state as the Turing machine's.

Transition to the left Then, to simulate a TM step with $x_2 \to x_2^{new}, L$. We change the order of push, and we have the resulting queue as $x_3x_4x_5 \sqcup x_1Hx_2^{new}$. Meanwhile, we change our state as the Turing machine's.

Reading $\sqcup$ When we are at such situation $x_1x_2x_3x_4x_5H\sqcup$, our reading will result in a new symbol. For left move, $\sqcup \to x_{new}, L$, we have the resulting queue as $x_1x_2x_3x_4x_5Hx_{new}\sqcup$ by pushing H first, then pushing x_{new} and $\sqcup$. For $\sqcup \to x_{new}, R$, we push x_{new}, then H and finally $\sqcup$ resulting in $x_1x_2x_3x_4x_5x_{new}H\sqcup$.

Accept Similar to the Turing machine, the queue automaton might not stop, but it accepts once it enters an accept state, and rejects once no valid transition is avaible or enters an rejection state.

[1] As a corollary, a two-stack PDA can simulate a queue automaton, and therefore a two-stack PDA is equivalent to a Turing machine.

Now, to simulate a queue automaton with TM. We do this with a two tape TM, where the first tape reads off the input, and the second tape is used to simulate a queue. The TM accepts when all string in the first tape has being processed and the state is in acceptance for queue automaton.