Theory of Computation, CSCI 438 spring 2022
Nondeterministic Turing Machines, reading pages 178-180, March 30th

1. Create a non-deterministic TM for L= $(ab)^* \cup (ab)^n c^n$ where $n \geq 0$.

High level plan

Nondeterministically, recognize the $(ab)^*$ or the $(ab)^n c^n$.

To recognize $(ab)^*$, alternate between seeing an ‘a’ and a ‘b’. Reject if end with an ‘a’ and accept if end with a ‘b’.

To recognize $(ab)^n c^n$, alternate between seeing an ‘a’ and a ‘b’. Reject if end with an ‘a’. If see a ‘c’ after a ‘b’, count that there are the same number of ‘c’s and ‘a’s and accept if there are. Reject otherwise.

Detail plan

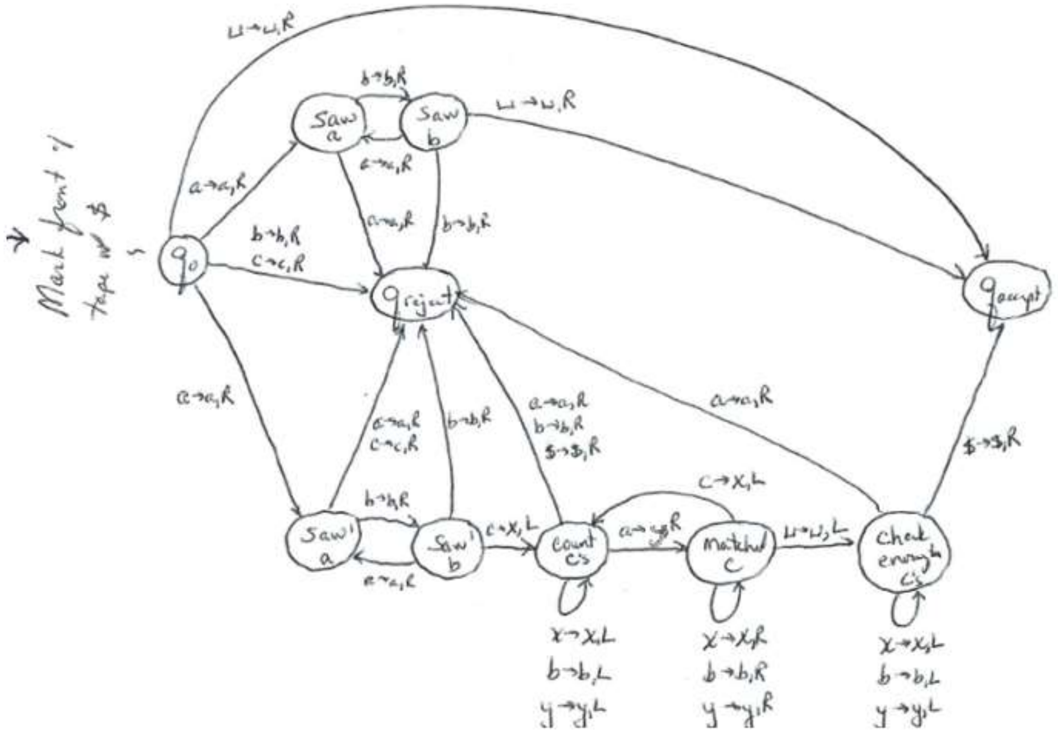
Mark the front of tape with $.

When see an ‘a’, nondeterministically choose whether to recognize the $(ab)^*$ or the $(ab)^n c^n$.

To recognize $(ab)^*$, alternate between seeing an ‘a’ and a ‘b’. Reject an ‘a’ follows as ‘a’, or if a ‘b’ follows a ‘b’. If reach a blank with no problems, accept.

To recognize $(ab)^n c^n$, alternate between seeing an ‘a’ and a ‘b’ as above. If reach a ‘c’, replace it with an ‘x’ and travel left to replace a matching ‘a’ with a ‘y’. Continue doing this until all of the ‘c’s are matched with ‘a’s. Reject if have a ‘c’ that isn’t matched with an ‘a’, or if all ‘a’s are not matched at the end.

Machine



Mark front of tape w/ $
⊔→⊔,R
b→b,R
Saw a
Saw b
⊔→⊔,R
a→a,R
a→a,R
a→a,R
b→b,R
b→b,R
c→c,R
q0
q reject
q accept
a→a,R
a→a,R
c→c,R
b→b,R
a→a,R
b→b,R
$→$,R
a→a,R
$→$,R
c→X,L
b→b,R
Saw' a
Saw' b
a→a,R
c→X,L
Count c's
a→y,R
Matched c
⊔→⊔,L
Check enough c's
X→X,L
b→b,L
y→y,L
X→X,R
b→b,R
y→y,R
X→X,L
b→b,L
y→y,L

2. What is the relation between Turing recognizable languages and languages recognizable by a nondeterministic Turing machine? Prove your answer.

Turing recognizable languages and the set of languages recognizable by non-deterministic Turing machines are the same.

In other words, the following Theorem holds:
Theorem: A language is Turing-recognizable iff some nondeterministic Turing machine recognizes it.
(Theorem 3.16, page 178)

Proof:
⇒ (only if) Given a language that is Turing-recognizable, then it is recognized by some nondeterministic TM. In other words, $\mathcal{L}(TM) \subseteq \mathcal{L}(TM_{ND})$.

Say that a language is Turing-recognizable. Since it is Turing-recognizable, there is a Turing machine M that recognizes it. This language is also recognized by a nondeterministic Turing machine N, which is very similar to M, except that N is a nondeterministic Turing machine, which doesn't use nondeterminism.

⇐ (if) Given a language that is recognized by a nondeterministic TM, then it is Turing-recognizable. In other words, $\mathcal{L}(TM_{ND}) \subseteq \mathcal{L}(TM)$.

Say that a language that is recognized by a nondeterministic Turing machine, N. Construct a deterministic Turing machine, D, that recognizes the same language.

High Level Plan

View the deterministic execution of a particular string by a particular nondeterministic Turing machine as the exploration of a search tree with states as vertices, symbols on the branches, and branches representing moves in the nondeterministic Turing machine. When a state and symbol in the nondeterministic machine have multiple choices, say x choices, corresponding vertex in the tree will have x children, one for each choice. When a state and symbol in the nondeterministic machine has only 1 choice, the corresponding vertex will have only 1 child. Accept and reject vertices do not need to have children.

The deterministic TM can simulate the nondeterministic TM by traversing this search looking for an 'accept' vertex. The deterministic TM will find an 'accept' vertex exactly when the nondeterministic TM will accept. When the deterministic TM finds a 'reject' vertex, it will not stop, rather it will continue to search because, a different branch may lead to an accept.

Thus, if at any point the deterministic machine enters an accept state, it accepts. If it enters a reject state, it explores the next branch. Thus, the deterministic TM does a breadth-first search of all possible executions. If one of these enters an accept state, accept. Otherwise, an exhaustive search may occur and the deterministic TM may reject, or the deterministic machine may search forever.

More Detailed Plan

Use a 3-tape machine to accomplish the breadth-first search.

1. Input tape: holds the original input and is never changed.
2. Simulation tape: For each simulation, a fresh copy of the input is copied onto this tape. A deterministic Turing machine is simulated on the tape, choosing the transitions indicated on the progress tape.
3. Progress tape: This tape controls the breadth-first search. It holds what moves to try next. The progress tape tells what transition to make each move, making a particular simulation deterministic.

Once all of the choices have been pursued at the given level, explore the nodes at the next level.

For each state/symbol combination, number the transitions 1, 2, … n. If there is only one choice for a state/symbol pair, it will be numbered 1.

Initially the progress tape holds 1. The simulation is run, using the 1st rule of the start symbol and the tape symbol under the read/write head. If an accept state is entered, the string is accepted. If an accept state is not entered, the progress tape is updated and a new simulation is started, beginning back at the start state, and with fresh input on the simulation tape.

Updating the progress tape: If there is another move to make for the state/symbol combination, that position on the tape is updated. If there is not another move, the position on the tape is set to 1, and the previous position on the tape is updated. If there is not another move at that position, that position is set to 1 and the previous position on the tape is updated. If we get back to the first position, and there are no more moves for that position, add a new 1 to the end of the progress tape. This indicates that the search will go another transition further, i.e. another level deeper on the tree.

With considerable thought, it can be seen that this new deterministic TM will have a finite number of states, and will accept exactly those strings as were in the language of the nondeterministic machine.