15-251: Great Theoretical Ideas In Computer Science

Recitation 2 Solutions

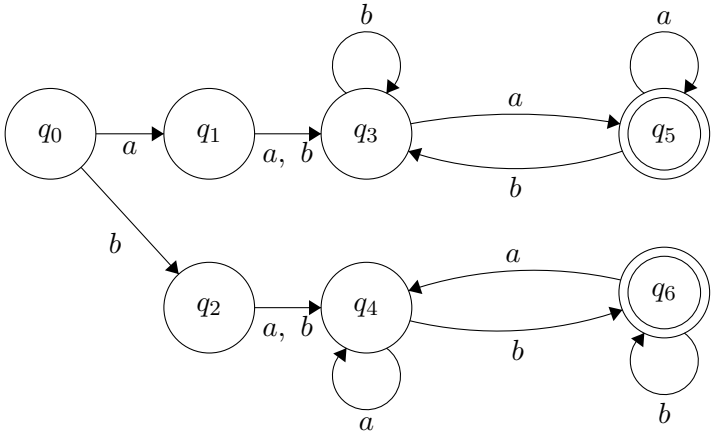
Announcements

- Be sure to start early on homeworks! It will be a smoother experience for all involved (less crowded office hours are better for you and us!)
- Please practice writing up your solutions beforehand. It will help with:
  - Structuring your proof on the page in an organized way.
  - Writing neatly (minimizing erasing/crossing out with practice).
  - Ensuring your solution is correct (often bugs pop up while writing).
- If you aren't sure about a piece of feedback or why you got a particular score on a problem, please ask us! The TAs initialed which problems they graded, so go to the correct TA if you are at all unsure. We're here to help!
- If you have any questions about the course overall, not restricted to homework, please ask! Again, we want to help you however we can!
- Reminder: Make sure that you're familiar with the collaboration policies on the website. The homework writing sessions and open collaboration problems are very different from many classes, so just make sure you're clear on the specifics.

Nature of Language

Which of the following languages are regular? Prove your answer.

(a) $L = \{xwx^R : x, w \in \Sigma^*, |x|, |w| > 0\}$, where $\Sigma = \{a, b\}$ and x^R means the reversal of x.



We note that this is just the set of strings that begin or end with the same letter. Our DFA splits this into the case where our string starts with a or b and then needs at least one more character to ensure w is not empty and checks if it ends with the same character.

(b) $L = \{x : x = x^R, x \in \Sigma^*\}$, where $\Sigma = \{a, b\}$.

This language is not regular. Suppose there is a DFA that accepts it on N states. Consider the set of strings $\{a^n : n \in \mathbb{N}, n \leq N\}$ then by the pigeonhole principle we know that a^i and a^j end on the same state where $i < j$. Then by appending ba^i we have a^iba^i is a palindrome but a^jba^i is not, hence we have reached a contradiction so there is no such DFA.

Closure of Regularity

(a) Prove that if L_1 and L_2 are regular, then $L_1 \cap L_2$ is regular.

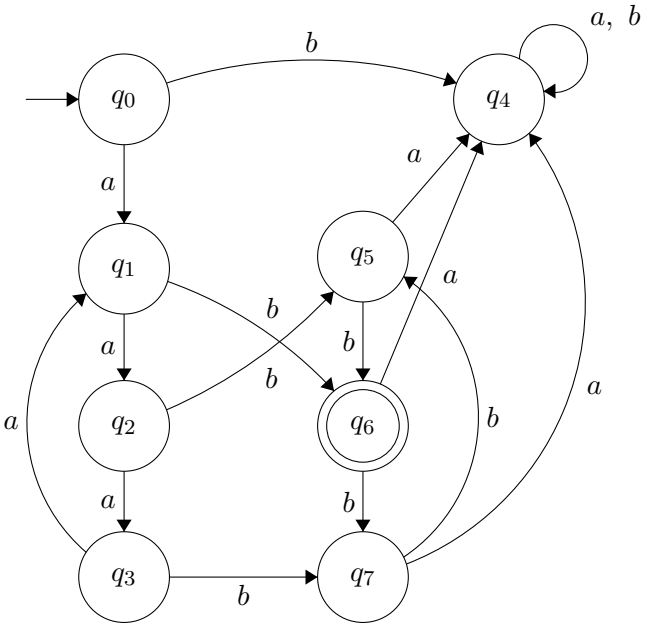
Use the construction for union from lecture, where we took the cross product of the state sets of L_1 and L_2, and copy the same transitions.
Instead of setting a state to be accept if either of the elements of the pair accepted, instead we make a state (a, b) accept if and only if both a and b were accept states.
This works as we require a string to be accepted by both L_1 and L_2, by definition of intersection.

(b) Using (a), show that $L = \{w : w$ has the same number of 0s and 1s$\}$ over the alphabet $\Sigma = \{0, 1\}$ is irregular.

Suppose for the sake of contradiction that L were regular.
Now, consider $L' = \{0^n1^m : n, m \in \mathbb{N}\}$.
This is trivially regular (have a DFA with 3 states, one while looping on 0, one while looping on 1, which is the accept state, and one failure state if we go back to 0).
However, note that $L \cap L' = \{0^n1^n : n \in \mathbb{N}\}$, which we have established to be irregular.
As regularity is closed under intersection, we have a contradiction, so L must be irregular.

DFAs Can Count

What language does the following DFA decide?

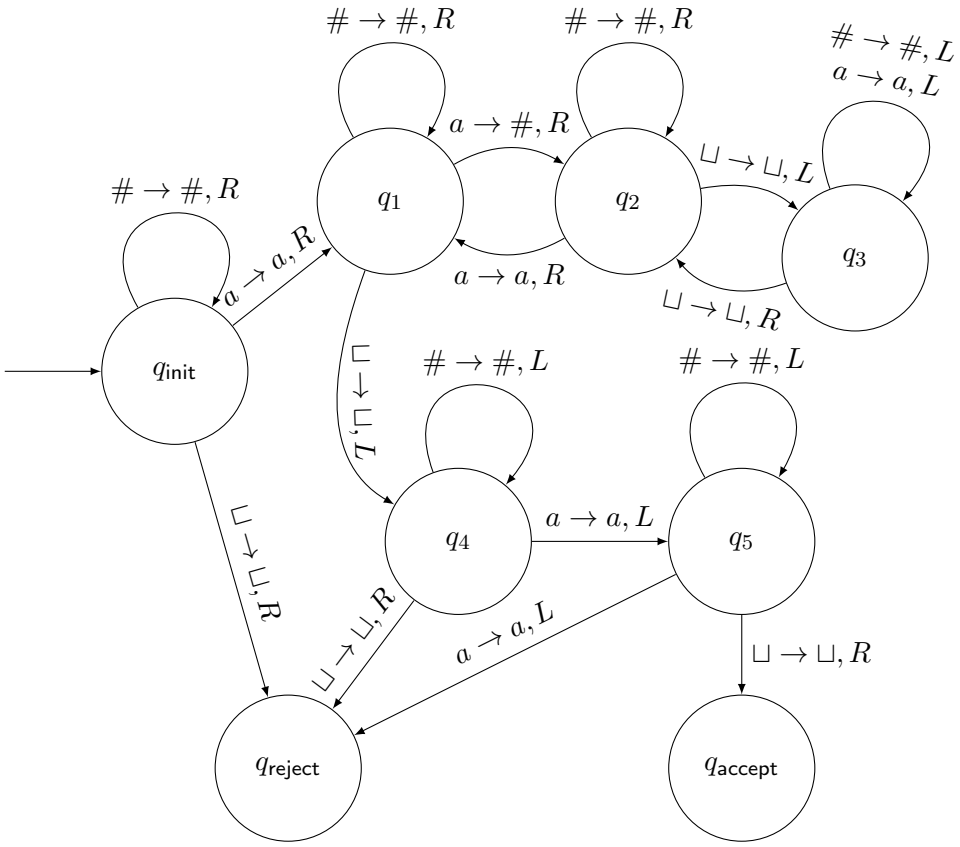


$L = \{a^n b^m : n, m > 0, n \equiv m \pmod 3\}$, where $\Sigma = \{a, b\}$.
Relabel $q_1 = a_1$, $q_2 = a_2$, $q_3 = a_0$, $q_5 = b_1$, $q_6 = b_2$, $q_7 = b_3$.
From the given DFA, clearly the empty string or any string starting with b is rejected. States a_0, a_1, a_2 keep track of the number of a's at the beginning of the string (mod 3). Once a b is encountered, we transition to either of the states b_1, b_2, b_3 depending on how many a's we encountered before.
Any string of any other form ends up in the failure state q_4.

Tough Decisions

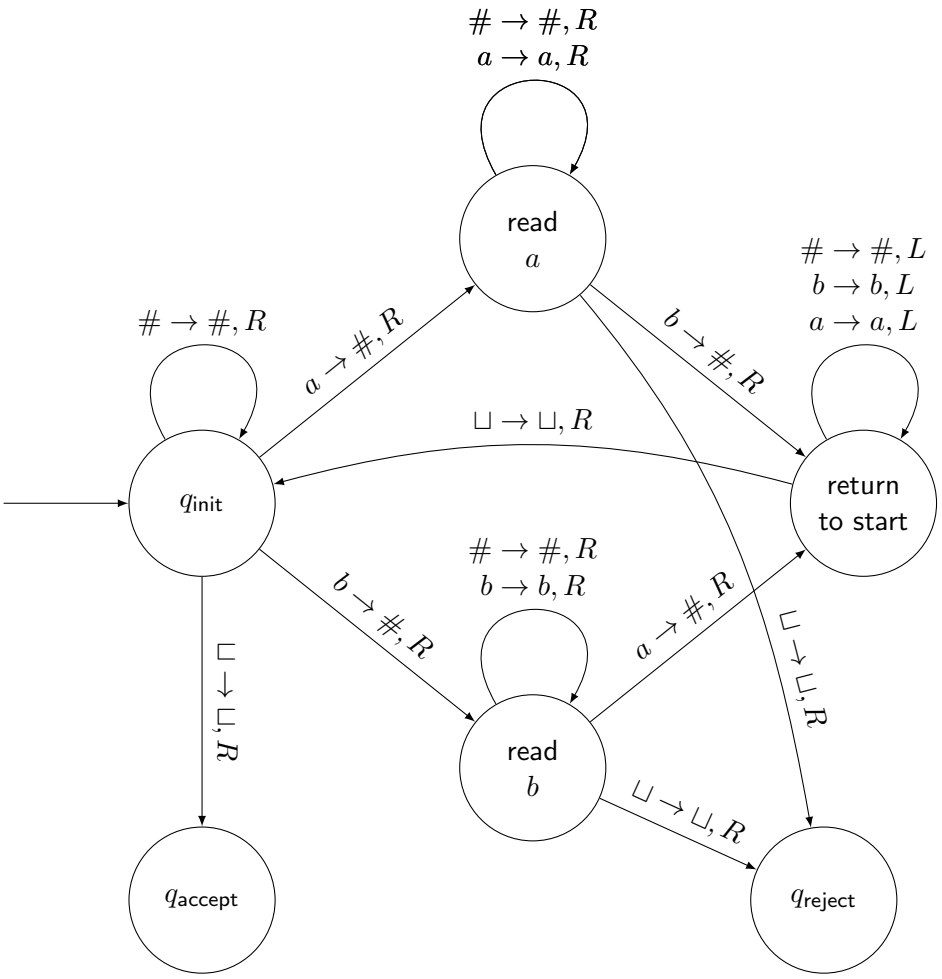
For each language below, draw a TM that decides the language. As usual, you can use any finite tape alphabet Γ containing Σ and $\sqcup$; just be sure to specify what it is. In addition, explain briefly in prose how the TM works, and the "meaning" of each state.

(a) $L = \{a^n : n \text{ is a nonnegative integer power of 2}\}$, where $\Sigma = \{a\}$.



The tape alphabet is $\Gamma = \{a, \sqcup, \#\}$.
This TM takes in a string S, marks every second a with $\#$ repeatedly over the tape until either one a remains (so the length of S is a nonnegative integer power of 2) or more than one a remains and a second a to remove cannot be found. In other words, we constantly divide the number of as by 2, until either we get 1 (accepted) or we get an non-1 odd number (rejected).
q_init is the initial state. If an empty string is read, it is rejected. Otherwise, we will read an a and go to state q_1.
q_1 indicates one a that is unmatched (yet) is read.
q_2 indicates that no as that are read are unmatched.
q_3 indicates that we have matched every pair of as and is in the process of going back to the beginning of the string.
q_4 indicates that there is only one unmatched a and every other a has been paired off.
q_5 indicates that there is at least one a remaining in the string. If we don't read any more as and reach the end of the string, then S is accepted. Otherwise, S is rejected.

(b) $L = \{x : x \text{ has the same number of } a\text{'s and } b\text{'s}\}$, where $\Sigma = \{a, b\}$.



The tape alphabet is $\{a, b, \#, \sqcup\}$.
In this TM, at each iteration within the non-blank part of the tape (the input tape cells), the first non $\#$ character is read and marked as $\#$. If there is no non $\#$ character, then we accept the string. If we find an a, then we go to the right and find a b to mark as $\#$. If we find a b, we find an a to the right of it to mark as $\#$. If we don't find the second character, then reject the string. Otherwise, once the second character is found and marked off, we return to the beginning of the string and repeat the process.

Powerful Vocabulary

For a language $L \subseteq \{0, 1\}^*$, define $\text{perm}(L)$ to be the set of all permutations of all the words in L. Show that if L is decidable, then $\text{perm}(L)$ is decidable.

We are not looking for a detailed TM description. We want to use the Church-Turing thesis and describe an algorithm to decide L.