Problem 1.

Prove or disprove: every finite language is recognized by some FA.

Solution

Proof. First note that L is finite, define $K = |L|$, the number of strings in L, K finite. As we have a finite number of strings, prescribe some arbitrary order to them, and assign each string some unique index i, where $1 \leq i \leq K$. Using this information, we can define an NFA which I claim will recognize the language. Start with some initial state s_0, if the empty string is in the language, s_0 is final, else it is not. Next, for each string, define a sequence of new states, starting with s_0 with each edge representing the next character in the string, the state will then represent the current progress into that string for that particular index. The terminating state of this sequence is final. By doing this, we form a tree starting at s_0 (which has K edges leaving it) with branches leaving s_0 each representing one of the (finite) strings in the language. All leaves with no children are final states, s_0 may or may not be, and no other states are final. Finally, if there are any characters in the alphabet which do not have edges from s_0 add them all pointing to a new arbitrary non-final trap state.

Claim: For every finite language, this algorithm produces an NFA representing it.
Let L be a finite language.
Let $w \in L$, then w represents a branch of the NFA leading to a final state $\Rightarrow w$ is represented by the NFA.
Let $x \notin L$, if x is empty, then it will never leave s_0, as $x \notin L$ then s_0 is not final, therefore x is not represented by the NFA. If x is not empty, it will travel down a path of our NFA, either the path leading to the trap state, or a path along a string. By construction of the NFA, x must land on a non-final state, else it would be necessarily equivalent to a string in L (contradiction).

Therefore, as the NFA produced by the algorithm recognizes all strings in L, and only strings in L, every finite language is recognized by the NFA generated by the algorithm $\Rightarrow$ every finite language is recognized by some FA.

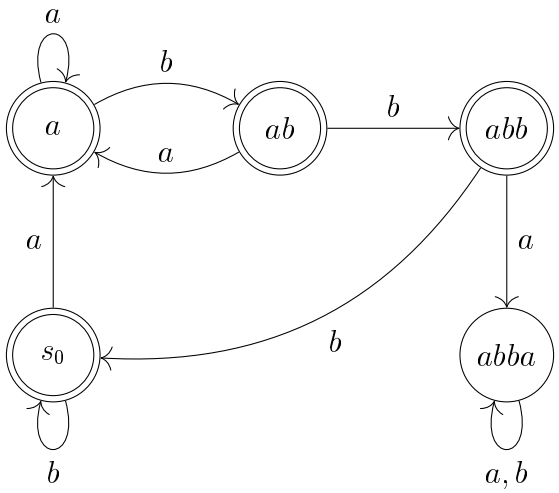
□

Problem 2.

Define a DFA, simplified to the best of your abilities, that recognizes the language $L = \{w \in \{a, b\}^* : w \text{ does not contain the substring } abba\}$.

Solution

The DFA for this is pretty straightforward, all states are final, until we find an occurrence of $abba$, in which case we hit a trap state. Each of the 4 other states represent how far away we are from discovering $abba$ (where $s0$ represents no progress).



Problem 3.

Conssider the n-bit binary representation of a natural number x:

$$\text{the binary representation of } x \text{ is } (x_{n-1}x_{n-2}\cdots x_1x_0) \iff x = \sum_{i=0}^{n-1} x_i 2^i$$

where each bit x_i is a binary digit, either zero or one.

$$\begin{aligned} L = &\{a_0b_0c_0\cdots a_{n-1}b_{n-1}c_{n-1} : n \in \mathbb{N} \wedge \forall i, 0 \leq i < n, a_i \in \{0,1\}, b_i \in \{0,1\}, c_i \in \{0,1\} \wedge \\ &(a_{n-1}\cdots a_0) + (b_{n-1}\cdots b_0)_2 = (c_{n-1}\cdots c_0)_2\} \end{aligned}$$

Define a DFA that accepts L.

Solution

The general idea of this DFA is simple, we have 3 sets of states, and one trap state. The DFA progresses through 3 columns, with the first column representing length mod 3 = 0. The second and third column represent partial sums of the first two bits. Depending on the sum and the next bit, we either enter the trap state, or return to the first column (where state r represents a remainder bit, which we cannot end on, as addition is incomplete).