Computer Science 181, Homework 2

Michael Wu
UID: 404751542

April 16th, 2018

Postponed Problem 2

d)

$$\{0, 00, 01, 000, 011, 1, 10, 11, 100, 111\}$$

I have chosen to include the empty string ϵ in L_b, which is why 0 and 1 are in this concatenation.

e)

$$\{\}$$

Postponed Problem 6

This is the set S of all strings s over $\Sigma = \{0, 1\}$ which have at least two 1's and ends in 1.

Problem 0

a) I would interpret this as recursing from the end of the string. The notation implies that the function gets the previous state from recursively applying the transition function δ^* to the prefix string consisting of everything except the last character in the string. Then it applies δ to the last character and the previous state to get the final state. In our old definition, we imply that the function recurses from the beginning of the string. First it applies the transition function δ to the first element of the string and the starting state to get the next state. Then it calls δ^* on the remainder of the string and the next state, which begins the recursion. This recursion ends when no more characters are remaining in the string, and returns the final state.

b) Consider an arbitrary string of length n

$$x = x_0 x_1 \dots x_n$$

such that

$$\forall i \in \{0, 1, \dots, n\} \;\; x_i \in \Sigma$$

Then using our new definition of $\delta^*(q, x)$ we have

$$\delta^*(q, x) = \delta(\delta^*(q, x_0 x_1 \dots x_{n-1}), x_n)$$

which has a base case of $\delta^*(q, x_0) = \delta(q, x_0)$. Thus through induction we have

$$\delta^*(q, x) = \delta(\delta(\dots \delta(\delta(q, x_0), x_1) \dots, x_{n-1}), x_n)$$

Using our old definition of $\delta^*(q, x)$ we have

$$\delta^*(q, x) = \delta^*(\delta(q, x_0), x_1 \dots x_n)$$

which we expand by continually adding an inner δ and moving the first character from the second element of δ^* inside a parenthesis. Thus induction also gives us

$$\delta^*(q, x) = \delta(\delta(\dots \delta(\delta(q, x_0), x_1) \dots, x_{n-1}), x_n)$$

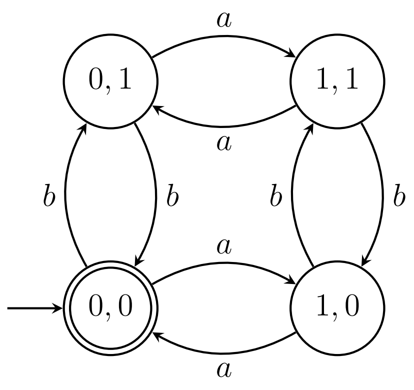
This is the exact same result given by the new definition of δ^*, so the definitions are equivalent.

Problem 1

a)

(x, y)	q	$\delta((x, y), q)$
$(0, 0)$	a	$(1, 0)$
$(0, 0)$	b	$(0, 1)$
$(0, 1)$	a	$(1, 1)$
$(0, 1)$	b	$(0, 0)$
$(1, 0)$	a	$(0, 0)$
$(1, 0)$	b	$(1, 1)$
$(1, 1)$	a	$(0, 1)$
$(1, 1)$	b	$(1, 0)$

b)



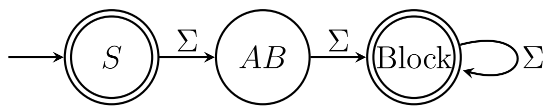
c) This is the set S of all strings s over $\Sigma = \{a, b\}$ which has an even number of a's and an even number of b's.

d) I would interpret the pair (x, y) using the following rules.

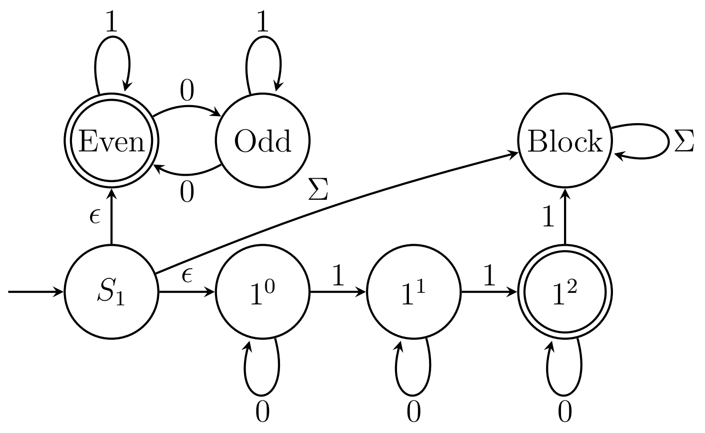
$$\begin{aligned}
\text{even number of } a\text{'s if } x = 0 \\
\text{odd number of } a\text{'s if } x = 1 \\
\text{even number of } b\text{'s if } y = 0 \\
\text{odd number of } b\text{'s if } y = 1
\end{aligned}$$

This allows us to interpret the corresponding names of the states in Q.

Problem 2



Problem 3



If the NFA can be not fully specified, this can be achieved with 6 states. Otherwise only using 6 states is not possible because both conditions require checking from the beginning of the string, and so no characters must be consumed before choosing a condition to validate. So there must be one starting state with two empty transitions, as shown above. The check for an even number of 0's requires two states, and the check for exactly two 1's requires three states. With the addition of a block state, there must be at least 7 states.

Problem 4

Two strings in the language are *bb* and *bab*. Two strings not in the language are ϵ and b.

Problem 5

The set of finite state languages is closed under homomorphisms on a language's alphabet Σ. Therefore, L_{five} is a finite state language if and only if every homomorphism on $\Sigma = \{a, b, c\}$ results in a finite state language when applied to L_{five}. But the homomorphism

$$h(a) = a \qquad h(b) = b \qquad h(c) = \epsilon$$

yields $h(L_{\text{five}}) = L_{\text{NFS}}$, which is not a finite state language. Therefore L_{five} cannot be a finite state language.