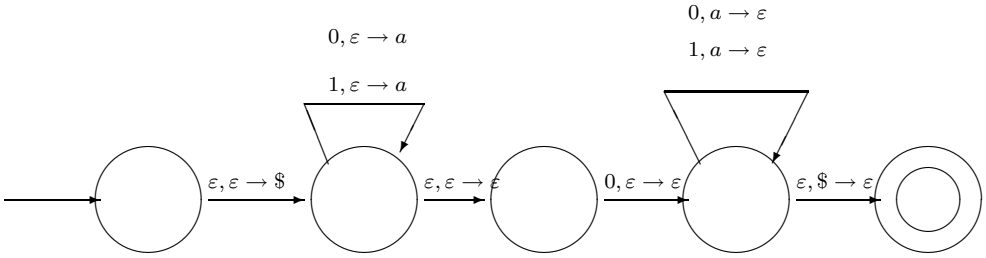
Solutions: Homework 4

Prepared by Chien-Chung Huang

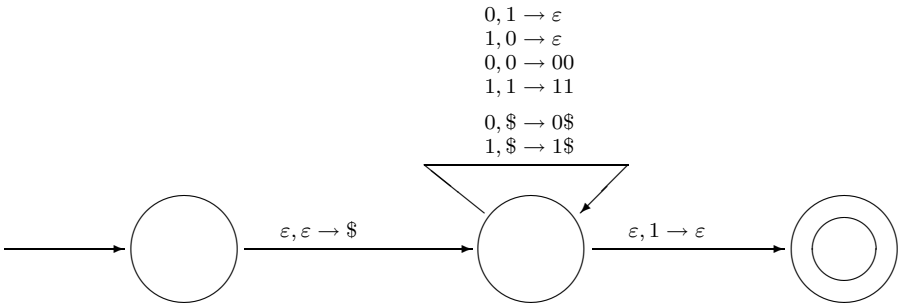
1

1.1 Answer:

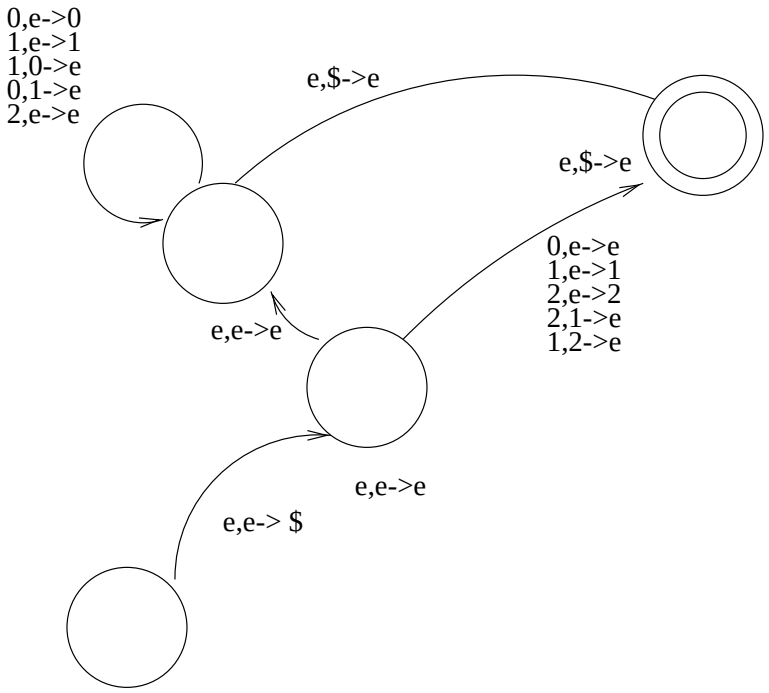
Answer: First, the PDA pushes a "rock" on the stack. From then on, at each point, the PDA splits itself into two incarnations: one inarnation believes that the next input symbol is the middle symbol and the second incarnation believes that the middle symbol is still some distance away in the input. The latter incarnation simply pushes the input symbol on its stack. The former throws out the input symbol, and from then on pops one stack symbol for every input symbol that it reads; when it sees the rock on the stack, it enters the accept state and stops consuming further input.



1.2 Answer: The PDA begins by pushing a rock on the stack. If the top symbol of the stack is either the rock or the same as the input symbol, then the PDA pushes the input symbol on the stack. If the input symbol and the stack symbol are different, the PDA consumes the input symbol and pops the top stack symbol. By these rules, at any point, the stack contains only 0's or only 1's, whichever occured more in the input consumed so far. So, the PDA enters accept state whenever the top stack symbol is 1.



1.3 The PDA begins by first pushing a rock on the stack. It then splits itself into two incarnations, each of which respectivly handles the possiblity of equal number of 1's and 0's, 1's and 2's. The PDA is shown below (notice that e is used to replace ε).



2

2.1 Answer:

Reflexive: $\forall w \in \Sigma^*(xw \in A \iff xw \in A) \longrightarrow x \equiv_A x$

Symmetric: $x \equiv_A y \implies \forall w \in \Sigma^*(xw \in A \iff yw \in A) \implies \forall w \in \Sigma^*(yw \in A \iff xw \in A) \implies y \equiv_A x$

Transitive

$x \equiv_A y \implies \forall w \in \Sigma^*(xw \in A \iff yw \in A)$

$y \equiv_A z \implies \forall w \in \Sigma^*(yw \in A \iff zw \in A)$

$\implies \forall w \in \Sigma^*(xw \in A \iff yw \in A \iff zw \in A) \implies x \equiv_A z$

Because $\equiv$ fulfills the three above requirements, it is an equivalence relation.

2.2 Answer:

$C_1 = \{a, b\}$

$C_2 = \{aa, ba\}$

$C_3 = \{aaa, baa\}$

$C_4 = \{\varepsilon\}$

$C_5 = \Sigma - (C_1 \cup C_2 \cup C_3 \cup C_4)$

Note: C_5 is the infinite class the question alludes to. Any string not found in the first four classes are related to one another in the following way,

$$\nexists w \in \Sigma^*(xw \in A \cap yw \in A),$$

i.e., $x \equiv_A y$ holds vacuously.

2.3 Answer:

$C_1 = \{a^*\}$

$C_2 = \{a^*bb^*\}$

$C_3 = \{a^*b^*cc^*\}$

$C_4 = \Sigma - (C_1 \cup C_2 \cup C_3)$

2.4 Answer:

$C_1 = \{(ab \cup ba)^*\}$

$C_2 = \{(ab \cup ba)^*a\}$

$C_3 = \{(ab \cup ba)^*b\}$

$C_4 = \Sigma - (C_1 \cup C_2 \cup C_3)$

2.5 Answer:

$C_0 = \{0^n1^n : n \geq 0\}$

$C_1 = \{0^n1^{n-1} : n \geq 1\}$

$C_2 = \{0^n1^{n-2} : n \geq 2\}$

...

$C_k = \{0^n1^{n-k} : n \geq k\}$

...

There are infinite classes. Also, there is another class $C_\# = \{0,1\}^* - \bigcup_i C_i$

2.6 Answer:

Given $x, y \in [x]_A$, we know that

$$\begin{aligned} & x \equiv_A y \\ \Longrightarrow \quad & \forall au \in \Sigma^*(xau \in A \Longleftrightarrow yau \in A) \\ \Longrightarrow \quad & \forall u \in \Sigma^*((xa)u \in A \Longleftrightarrow (ya)u \in A) \\ \Longrightarrow \quad & xa \equiv_A ya\,. \end{aligned}$$

2.7 Answer:

Informal descritpion: To prove a language is regular, one possible way is to create a DFA to describe it. A very intuitive idea here is to represent these equivalence classes as the states in the DFA. However, a very obvious question pops up immediately: How can we be sure that all the strings represented by a single state exhibit the same behavior? Fortunately, 2.6 gives us a strong hint: the class $[xa]_A$ (the new state) is completely determined by the class $[x]_A$ (the old state) and the alphabet symbol a (the input of the transition function). In other words, we can define the transition function based on the alphabet symbol and the old state.

Proof: Define $M = \{Q, \Sigma, \delta, q_0, F\}$,

$Q = \{q_{[x_1]_A}, q_{[x_2]_A}, ...q_{[x_n]_A}\}$,

$\delta(q_{[x_i]_A}, a) = q_{[x_j]_A}$, if $[x_j]_A = [x_ia]_A$,

$q_0 = q_{[\varepsilon]_A}$,

$F = \{q_{[x_i]_A} : x_i \in A\}$.

For the definition of F to make sense, we have to make sure that whether or not $x_i \in A$ depends only on the class $[x_i]_A$ and not the particular representative x_i of the class we pick. But this is easy: if $[x]_A = [y]_A$, then $x \equiv_A y$, which means $x\varepsilon \in A \Longleftrightarrow y\varepsilon \in A$, i.e., $x \in A \Longleftrightarrow y \in A$.

2.8 Answer:

(This answer written by Amit Chakrabarti)

Informal description: If A is a regular language over the alphabet Σ, we know we can build a DFA M to describe it. We will map strings in Σ^* to states of M in a natural way and show that

if two strings both map to the same state, then they are equivalent (under "$\equiv_A$").

In other words, if two strings are *not* equivalent, they will map to distinct states of M. If A did in fact have infinitely many left equivalence classes, then we could have found infinitely many strings (one from each class) such that no two were equivalent and these strings would have to map into an infinite number of different states. However, being a DFA, M only has a finite number of states.

Proof:

Given a regular language A, we can first build $M(A) = \{Q, \Sigma, \delta, q_0, F\}$. Recall the function $\hat{\delta} : Q \times \Sigma^* \to Q$ from class (and the Oct 11 lecture notes): $\hat{\delta}(q, x)$ tells us the state $M(A)$ ends up in when it is started in state q and fed input x. For each string $x \in \Sigma^*$, define $q(x) = \hat{\delta}(q_0, x)$. This is the state that we're going to map x to.

We claim that if $q(x) = q(y)$, then $x \equiv_A y$. As explained in the informal description, this would complete the proof.

Suppose $q(x) = q(y)$. We must show that $\forall w\,(xw \in A \iff yw \in A)$. Suppose w is a string such that $xw \in A$. Then, the machine $M(A)$ must accept xw, so $q(xw) \in F$. But $q(xw) = \hat{\delta}(q(x), w) = \hat{\delta}(q(y), w) = q(yw)$; thus $q(yw) \in F$ as well. So $M(A)$ accepts yw, i.e., $yw \in A$. We have shown that $xw \in A \implies yw \in A$. Clearly, a very similar proof shows that $xw \in A \impliedby yw \in A$, and we are done.

2.9 Answer:

Proof by contradiction.

Assume L_4 is regular, From 2.8 we know that A has only finitely many distinct left equivalence classes. However, from 2.5 we know that A in fact has infinite left equivalence classes. Thus we derive a contradiction.

2.10 Answer:

Consider the following collection of strings $A = \{01, 001, 0001, 00001, ...\}$. Not any two of strings in this set belong to the same left equivalence class. Thus, A has infinitely many classes. Since A only contains a portion of the left equivalence classes, the total classes must be infinite. As in 2.9, we know that the languages with infinitely many classes cannot be regular.