Solutions: Homework 1
Prepared by Chien-Chung Huang

1 Operations on languages

1.1) Suppose you split any string of the form 0^n1^n into x_2x_1. Then either x_2 looks like 0^k and x_1 looks like $0^p1^q(q = k + p)$ or x_2 looks like 0^k1^p and x_1 looks like $1^p(k = p + q)$. Thus, if L=$\{0^n1^n \mid n > 0\}$, CYCLE(L) = $\{0^p1^q0^k \mid q = p + k\} \cup \{1^q0^k1^p \mid k = p + q\}$.

1.2) The trick is to come up with a language L such that if a string is in L, then none of its proper prefixes are in L. Here is such an infinite language L: (Let Σ be $\{0, 1\}$)
L = {10, 110, 1110, 11110, ...}
More formally, L = $\{1^n0 \mid n > 0\}$.
It is obvious that MIN(L) = MAX(L) = L.

1.3) No. For a proof, assume that MIN(L)= Σ^*. Let a be any symbol in Σ (since an alphabet, as we defined in class, is non-empty, such a symbol must exist in Σ). Since MIN(L) = Σ^*, the string a, aa must be in MIN(L). Since every string that is in MIN(L) is, by definition, also in L, it follows that a, aa are also in L. Thus, a proper prefix of the string aa is in L. So aa is not in MIN(L). This contradicts that MIN(L) = Σ^*. Hence the proof.

1.4) $HALF(L)$ contains the first halves of all even-length strings in L.
Take any strings x of the form 0^m1^n. Is there a string y of the same length as x such that xy is in $L = \{0^p1^q0^r \mid q = p + r\}$? Yes, take y to be 1^m0^n; then xy is surely in L. We also claim that 0^m1^n is the only form x can take, since the form $1^s0^m1^n$ can not be the prefix string in L.

We conclude that, for $L = \{0^p1^q0^r \mid q = p + r\}$, $HALF(L) = \{0^m1^n \mid \text{any } m,\ n\}$.

1.5) If x is any string of the form $0^n\ 1^n$, then xx^R is $0^n1^n1^n0^n = 0^n1^{2n}0^n$ and such a string is in L. Any other forms of x such as $1^m0^n1^n$ or $0^n1^n0^m$ or 0^m1^n causes xx^R is not in L anymore. Thus, for $L = \{0^p1^q0^r \mid q = p + r\}$, $HALFPALINDROME(L) = \{0^n1^n \mid \text{any } n\}$.

2 Tables $\longleftrightarrow$ Diagrams

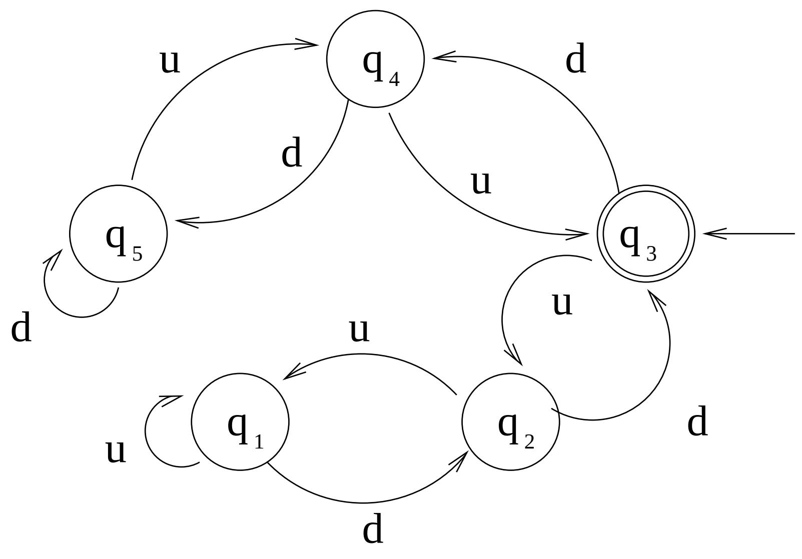
2.1) $M_1 = (Q, \Sigma, \delta, q_1, q_2)$.
$Q = \{q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, and

$\delta(q_1, a) = q_2, \delta(q_1, b) = q_1, \delta(q_2, a) = q_3,$
$\delta(q_2, b) = q_3, \delta(q_3, a) = q_2, \delta(q_3, a) = q_1$

$M_2 = (Q, \Sigma, \delta, q_1, \{q_1, q_4\})$.
$Q = \{q_1, q_2, q_3, q_4\}$, $\Sigma = \{a, b\}$, and

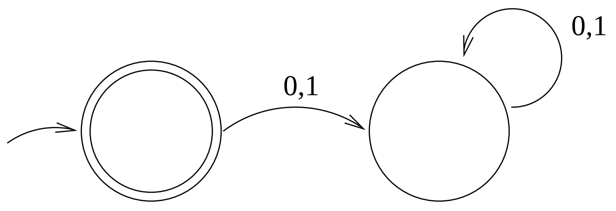
$\delta(q_1, a) = q_1, \delta(q_1, b) = q_2, \delta(q_2, a) = q_3,$
$\delta(q_2, b) = q_4, \delta(q_3, a) = q_2, \delta(q_3, b) = q_1,$
$\delta(q_4, a) = q_3, \delta(q_4, b) = q_4.$

2.2)

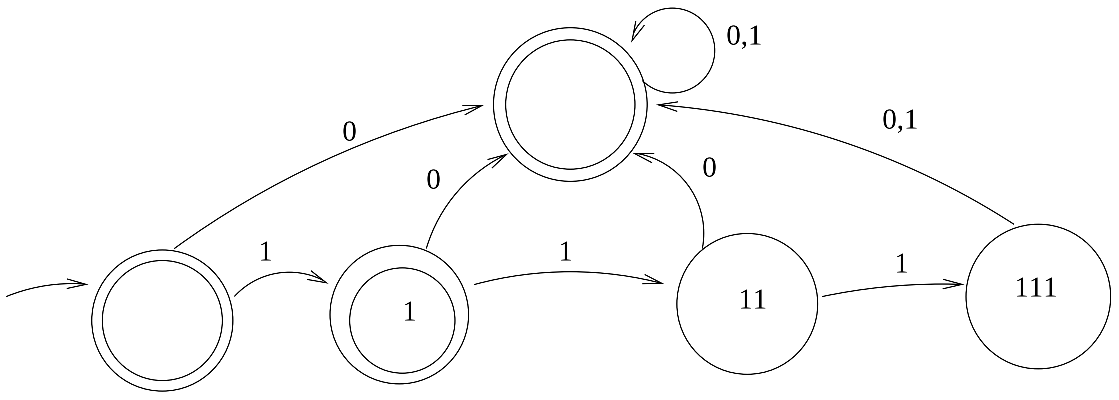


3 Designing DFAs

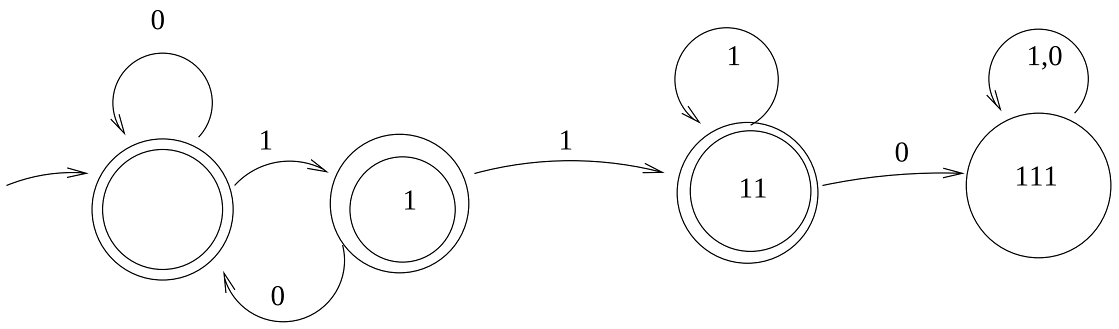
3.1) If the start state is nonfinal and the DFA remains in the start state always, then it clearly does not accept any string. Formally, here is DFA M $= (Q, \Sigma, \delta, q_0, F)$:
$Q = \{q_0\}$
$\Sigma = \{0,\ 1\}$
$F =\ empty\ set$
$\delta(q_0, a) = q_0\ for\ any\ a\ in\ \Sigma.$



3.2) As we read the first two or three symbols of the input, we will remember what they are if it seems like the string might be one of 11 or 111. The moment we notice that the string is surely not one of 11, 111 we go to a final state. Formally, here is the DFA M=$(Q, \Sigma, \delta, q_start, F)$:
$Q = \{q_{start},\ [1]\ [11],\ [111],\ q_{happy}\}$
$\Sigma = \{0,\ 1\}$
$F = \{q_{start},\ q_{happy}\}$
$\delta(q_{start},\ 0) = q_{happy}$
$\delta(q_{start},\ 1) = [1]$
$\delta([1],\ 0) = q_{happy}$
$\delta([1],\ 1) = [11]$
$\delta([11],\ 0) = q_{happy}$
$\delta([11],\ 1) = [111]$
$\delta(q_{happy},\ a) = q_{happy}\ for\ any\ a\ in\ \Sigma.$



3.3) The main point is that we keep a count on how many 1's have appeared in the substring.
M=$(Q, \Sigma, \delta, q_start, F)$
$Q\{q_{start}, [1], [11], [110]\}.\Sigma = \{0,1\}, F = \{q_{start}, [1], [11]\}.$
$\delta(q_{start}, 0) = q_{start},$
$\delta(q_{start}, 1) = [1],$
$\delta([1], 0) = q_{start},$
$\delta(1, 1) = [11],$
$\delta([11], 0) = [110],$
$\delta([11], 1) = [11],$
$\delta([110], 0) = [110],$
$\delta([110], 1) = [110]$

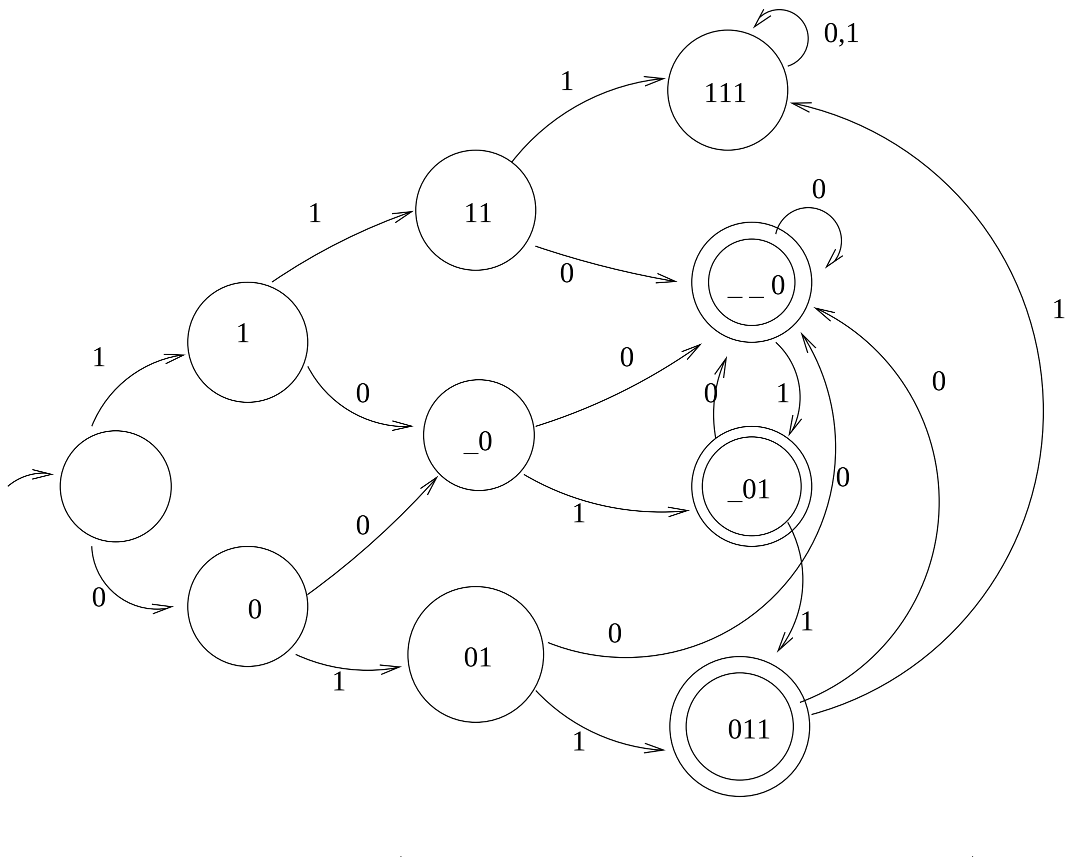


3.4) Informally, our DFA should keep track of the last three symbols seen. Furthermore, the moment it notices that less than one of the last three symbols seen are $0'$s, it should recognize right away that the string is no good, regardless of how the rest of the string is. Thus, once it is in a state [i, j, k] where three of i, j, k are $1'$s, it should never leave that "bad" state. With this intuition, let's define the DFA formally.

Let M=$(Q, \Sigma, \delta, q_0, F)$ be a DFA defined as follows:

$Q = \{[k_1k_2...k_n] \mid 0 \leq n \leq 3, each\ k_i\ is\ 0\ or\ 1\ \}$.
$\Sigma = \{0,\ 1\}$
$q = [\,]$
$F = \{[k_1k_2k_3] \mid$ one or more of k_1, k_2, k_3 are 0's $\}$
δ is defined as follows:

For $n < 3,\ \delta([k_1k_2...k_n], a) = [k_1k_2...k_na]$
If one or more of k_1, k_2, k_3 are 0's, $\delta([k_1k_2k_3], a) = [k_2k_3a]$
else $\delta([k_1k_2k_3], a) = [k_1k_2k_3]$
(Notice how the last rule makes sure that once the DFA reaches a bad state it will remain there forever.)
This DFA recognizes the given language.

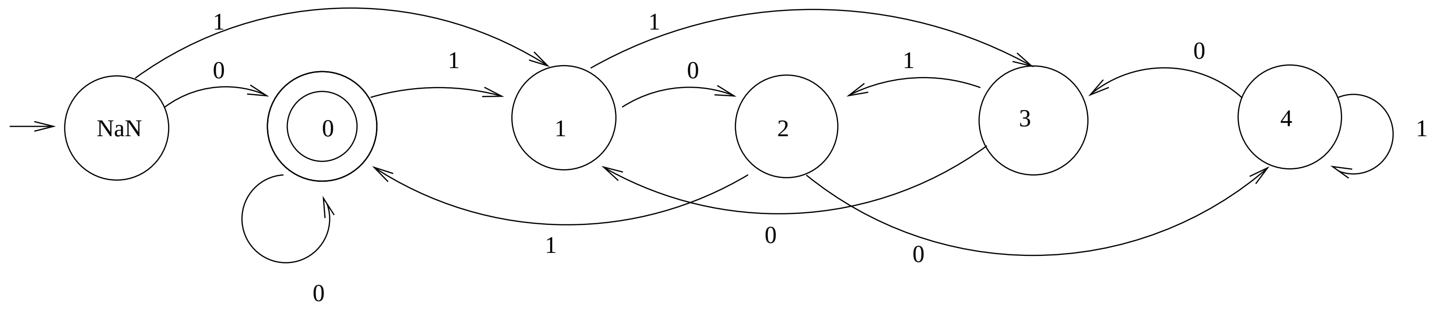


3.5) As we read the binary string (from most significant bit to least significant), let us only keep track of the remainder of the number that we have so far read (when the number is divided by 5). Thus, if we have so far read the string 110 (so its value is 6), we will remember 1. Since we only keep track of the remainder and there are only five possible remainders (0, 1, 2, 3, 4,)when a number is divided by 5, our DFA will have only five states, named 0, 1, 2, 3, 4. The trick is to come up with the correct transition function.

To get this, we will make the following observation.
Let rem(w) denote the remainder when the binary string w is divided by 5. With simple arithmetic, we can show that, if w is a string and a is a symbol, then rem(wa) is the remainder left by 2*rem(w)+a. For example, suppose w is 110 and a is 1. Then, rem(w) = 1, rem(wa) = rem(1101) =3, and you see that rem(wa) is the remainder left by 2*rem(w)+a = 3!

From the above observation, it is easy to figure out the transitions, From state r, on reading a symbol a, we must go to state s, where s is the remainder left by $2r + a$.

Formally, here is the DFA M=$(Q, \Sigma, \delta, q_0, F)$:
$Q = \{0,\ 1,\ 2,\ 3,\ 4\ \}$
$\Sigma = \{0,\ 1\}$
$q_0 = 0$
$F = \{0\}$
δ(q,a)= p where p is the remainder obtained when 2q+a is divides by 5.



3.6) Informally, our DFA should keep track of the last 100 symbols seen. Furthermore, the moment it notices that less than 10 out of the last 100 symbols seen are $0^{'}$s, it should recognize right away that the string is no good, regardless of how the rest of the string is. Thus, once it is in a state $[k_1k_2...k_{100}]$ where less than 10 k_i are 0s, it should never leave that "bad" state. With this intuition, let's define the DFA formally.

Let M=$(Q, \Sigma, \delta, q_0, F)$ be a DFA defined as follows:

$Q = \{[k_1k_2...k_n] \mid 0 \leq n \leq 100, each\ k_i\ is\ 0\ or\ 1\ \}$.
$\Sigma = \{0,\ 1\}$
$q = [\,]$
$F = \{[k_1k_2...k_{100}] \mid$ at least 10 out of $k_1, k_2...k_{100}$ are 0's $\}$
δ is defined as follows:

For $n < 100$, $\delta([k_1k_2...k_n], a) = [k_1k_2...k_na]$
If at least 10 of k_1, k_2, k_3 are 0's, $\delta([k_1k_2...k_{100}], A) = [K_2k_3...k_{100}a]$
else $\delta([k_1k_2...k_{100}], a) = [k_1k_2...k_{100}]$