Introduction to Automata Theory, Languages, and Computation

Solutions for Chapter 2

Revised 9/6/01.

Solutions for Section 2.2

Exercise 2.2.1(a)

States correspond to the eight combinations of switch positions, and also must indicate whether the previous roll came out at *D*, i.e., whether the previous input was accepted. Let 0 represent a position to the left (as in the diagram) and 1 a position to the right. Each state can be represented by a sequence of three 0's or 1's, representing the directions of the three switches, in order from left to right. We follow these three bits by either *a* indicating it is an accepting state or *r*, indicating rejection. Of the 16 possible states, it turns out that only 13 are accessible from the initial state, 000r. Here is the transition table:

	A	B
->000r	100r	011r
*000a	100r	011r
*001a	101r	000a
010r	110r	001a
*010a	110r	001a
011r	111r	010a
100r	010r	111r
*100a	010r	111r
101r	011r	100a
*101a	011r	100a
110r	000a	101a
*110a	000a	101a

111r	001a	110a

Exercise 2.2.2

The statement to be proved is $\delta\text{-}hat(q,xy) = \delta\text{-}hat(\delta\text{-}hat(q,x),y)$, and we proceed by induction on the length of y.

Basis: If $y = \varepsilon$, then the statement is $\delta\text{-}hat(q,x) = \delta\text{-}hat(\delta\text{-}hat(q,x),\varepsilon)$. This statement follows from the basis in the definition of $\delta\text{-}hat$. Note that in applying this definition, we must treat $\delta\text{-}hat(q,x)$ as if it were just a state, say p. Then, the statement to be proved is $p = \delta\text{-}hat(p,\varepsilon)$, which is easy to recognize as the basis in the definition of $\delta\text{-}hat$.

Induction: Assume the statement for strings shorter than y, and break $y = za$, where a is the last symbol of y. The steps converting $\delta\text{-}hat(\delta\text{-}hat(q,x),y)$ to $\delta\text{-}hat(q,xy)$ are summarized in the following table:

Expression	Reason
$\delta\text{-}hat(\delta\text{-}hat(q,x),y)$	Start
$\delta\text{-}hat(\delta\text{-}hat(q,x),za)$	$y=za$ by assumption
$\delta(\delta\text{-}hat(\delta\text{-}hat(q,x),z),a)$	Definition of $\delta\text{-}hat$, treating $\delta\text{-}hat(q,x)$ as a state
$\delta(\delta\text{-}hat(q,xz),a)$	Inductive hypothesis
$\delta\text{-}hat(q,xza)$	Definition of $\delta\text{-}hat$
$\delta\text{-}hat(q,xy)$	$y=za$

Exercise 2.2.4(a)

The intuitive meanings of states A, B, and C are that the string seen so far ends in 0, 1, or at least 2 zeros.

	0	**1**
->A	B	A
B	C	A
*C	C	A

Exercise 2.2.6(a)

The trick is to realize that reading another bit either multiplies the number seen so far by 2 (if it is a 0), or multiplies by 2 and then adds 1 (if it is a 1). We don't need to remember the entire number seen --- just its remainder when divided by 5. That is, if we have any number of the form $5a+b$, where b is the remainder, between 0 and 4, then $2(5a+b) = 10a+2b$. Since $10a$ is surely divisible by 5, the remainder of $10a+2b$ is the same as the remainder of 2b when divided by 5. Since b, is 0, 1, 2, 3, or 4, we can tabulate the answers easily. The same idea holds if we want to consider what happens to $5a+b$ if we multiply by 2 and add 1.

The table below shows this automaton. State qi means that the input seen so far has remainder i when divided by 5.

	0	**1**
->*q0	q0	q1
q1	q2	q3
q2	q4	q0
q3	q1	q2
q4	q3	q4

There is a small matter, however, that this automaton accepts strings with leading 0's. Since the problem calls for accepting only those strings that begin with 1, we need an additional state *s*, the start state, and an additional ``dead state'' *d*. If, in state *s*, we see a 1 first, we act like *q0*; i.e., we go to state *q1*. However, if the first input is 0, we should never accept, so we go to state *d*, which we never leave. The complete automaton is:

	0	**1**
->s	d	q1
*q0	q0	q1
q1	q2	q3
q2	q4	q0
q3	q1	q2
q4	q3	q4
d	d	d

Exercise 2.2.9

Part (a) is an easy induction on the length of *w*, starting at length 1.

Basis: $|w| = 1$. Then δ*-hat*$(q_0,w) = \delta$*-hat*(q_f,w), because *w* is a single symbol, and δ*-hat* agrees with δ on single symbols.

Induction: Let $w = za$, so the inductive hypothesis applies to *z*. Then δ*-hat*$(q_0,w) = \delta$*-hat*$(q_0,za) = \delta(\delta$*-hat*$(q_0,z),a) = \delta(\delta$*-hat*$(q_f,z),a)$ [by the inductive hypothesis] $= \delta$*-hat*$(q_f,za) = \delta$*-hat*(q_f,w).

For part (b), we know that δ*-hat*$(q_0,x) = q_f$. Since $x\varepsilon$, *we know by part (a) that* δ*-hat*$(q_f,x) = q_f$. *It is then a simple induction on k to show that* δ*-hat*$(q_0,x^k) = q_f$.

Basis: For k=1 the statement is given.

Induction: Assume the statement for k-1; i.e., δ*-hat*$(q_0$*,xSUP>k-1*$) = q_f$*. Using Exercise 2.2.2,* δ*-hat*$(q_0,x^k) = \delta$*-hat*$(\delta$*-hat*$(q_0,x^{k-1}),x) = \delta$*-hat*$(q_f,x)$ *[by the inductive hypothesis]* $= q_f$ *[by (a)].*

Exercise 2.2.10

The automaton tells whether the number of 1's seen is even (state A) or odd (state B), accepting in the latter case. It is an easy induction on |w| to show that dh(A,w) = A if and only if w has an even number of 1's.

Basis: |w| = 0. Then w, the empty string surely has an even number of 1's, namely zero 1's, and δ-hat(A,w) = A.

Induction: Assume the statement for strings shorter than w. Then w = za, where a is either 0 or 1.

Case 1: a = 0. If w has an even number of 1's, so does z. By the inductive hypothesis, δ-hat(A,z) = A. The transitions of the DFA tell us δ-hat(A,w) = A. If w has an odd number of 1's, then so does z. By the inductive hypothesis, δ-hat(A,z) = B, and the transitions of the DFA tell us δ-hat(A,w) = B. Thus, in this case, δ-hat(A,w) = A if and only if w has an even number of 1's.

Case 2: a = 1. If w has an even number of 1's, then z has an odd number of 1's. By the inductive hypothesis, δ-hat(A,z) = B. The transitions of the DFA tell us δ-hat(A,w) = A. If w has an odd number of 1's, then z has an even number of 1's. By the inductive hypothesis, δ-hat(A,z) = A, and the transitions of the DFA tell us δ-hat(A,w) = B. Thus, in this case as well, δ-hat(A,w) = A if and only if w has an even number of 1's.

[Return to Top](#)

Solutions for Section 2.3

Exercise 2.3.1

Here are the sets of NFA states represented by each of the DFA states A through H: A = {p}; B = {p,q}; C = {p,r}; D = {p,q,r}; E = {p,q,s}; F = {p,q,r,s}; G = {p,r,s}; H = {p,s}.

	0	**1**
->A	B	A
B	D	C
C	E	A
D	F	C
*E	F	G
*F	F	G
*G	E	H
*H	E	H

Exercise 2.3.4(a)

The idea is to use a state qi, for i = 0,1,...,9 to represent the idea that we have seen an input i and guessed that this is the repeated digit at the end. We also have state qs, the initial state, and qf, the final state. We stay in state qs all the time; it represents no guess having been made. The transition table:

	0	**1**	**...**	**9**

->qs	{qs,q0}	{qs,q1}	...	{qs,q9}
q0	{qf}	{q0}	...	{q0}
q1	{q1}	{qf}	...	{q1}
...	...	...	...	...
q9	{q9}	{q9}	...	{qf}
*qf	{}	{}	...	{}

[Return to Top](#)

Solutions for Section 2.4

Exercise 2.4.1(a)

We'll use q0 as the start state. q1, q2, and q3 will recognize abc; q4, q5, and q6 will recognize abd, and q7 through q10 will recognize aacd. The transition table is:

	a	b	c	d
->q0	{q0,q1,q4,q7}	{q0}	{q0}	{q0}
q1	{}	{q2}	{}	{}
q2	{}	{}	{q3}	{}
*q3	{}	{}	{}	{}
q4	{}	{q5}	{}	{}
q5	{}	{}	{}	{q6}
*q6	{}	{}	{}	{}
q7	{q8}	{}	{}	{}
q8	{}	{}	{q9}	{}
q9	{}	{}	{}	{q10}
*q10	{}	{}	{}	{}

Exercise 2.4.2(a)

The subset construction gives us the following states, each representing the subset of the NFA states indicated: A = {q0}; B = {q0,q1,q4,q7}; C = {q0,q1,q4,q7,q8}; D = {q0,q2,q5}; E = {q0,q9}; F = {q0,q3}; G = {q0,q6}; H = {q0,q10}. Note that F, G and H can be combined into one accepting state, or we can use these three state to signal the recognition of abc, abd, and aacd, respectively.

	a	b	c	d
->A	B	A	A	A
B	C	D	A	A
C	C	D	E	A

D	B	A	F	G
E	B	A	A	H
*F	B	A	A	A
*G	B	A	A	A
*H	B	A	A	A

[Return to Top](#)

Solutions for Section 2.5

Exercise 2.5.1

For part (a): the closure of p is just {p}; for q it is {p,q}, and for r it is {p,q,r}.

For (b), begin by noticing that a always leaves the state unchanged. Thus, we can think of the effect of strings of b's and c's only. To begin, notice that the only ways to get from p to r for the first time, using only b, c, and ε-transitions are bb, bc, and c. After getting to r, we can return to r reading either b or c. Thus, every string of length 3 or less, consisting of b's and c's only, is accepted, with the exception of the string b. However, we have to allow a's as well. When we try to insert a's in these strings, yet keeping the length to 3 or less, we find that every string of a's b's, and c's with at most one a is accepted. Also, the strings consisting of one c and up to 2 a's are accepted; other strings are rejected.

There are three DFA states accessible from the initial state, which is the ε closure of p, or {p}. Let A = {p}, B = {p,q}, and C = {p,q,r}. Then the transition table is:

	a	**b**	**c**
->A	A	B	C
B	B	C	C
*C	C	C	C

[Return to Top](#)