Chapter 12 Solutions, Susanna Epp Discrete Math 5th Edition

https://github.com/spamegg1

January 4, 2024

Contents

1 Exercise Set 12.1

In 1 and 2, let $\Sigma = \{x, y\}$ be an alphabet.

1.1 Exercise 1

1.1.1 (a)

Let L_1 be the language consisting of all strings over Σ that are palindromes and have length ≤ 4. List the elements of L_1 between braces.

Proof. $L_1 = \{\lambda, x, y, xx, yy, xxx, xyx, yxy, yyy, xxxx, xyyx, yxxy, yyyy\}$ □

1.1.2 (b)

Let L_2 be the language consisting of all strings over Σ that begin with an x and have length ≤ 3. List the elements of L_2.

Proof. $L_2 = \{x, xx, xy, xxx, xxy, xyx, xyy\}$ □

1.2 Exercise 2

1.2.1 (a)

Let L_3 be the language consisting of all strings over Σ of length ≤ 3 in which all the x's appear to the left of all the y's. List the elements of L_3 between braces.

Proof. $L_3 = \{\lambda, x, y, xx, xy, yy, xxx, xxy, xyy, yyy\}$ □

1.2.2 (b)

List between braces the elements of Σ^4, the set of all strings of length 4 over Σ.

Proof. $L_4 = \{xxxx, xxxy, xxyx, xxyy, xyxx, xyxy, xyyx, xyyy, yxxx, yxxy, yxyx, yxyy, yyxx, yyxy, yyyx, yyyy\}$ □

1.2.3 (c)

Let $A = \Sigma^1 \cup \Sigma^2$ and $B = \Sigma^3 \cup \Sigma^4$. Describe A, B, and $A \cup B$ in words.

Proof. A is the set of strings over Σ of length 1 or 2. B is the set of strings over Σ of length 3 or 4. $A \cup B$ is the set of strings over Σ of length 1 or 2 or 3 or 4. □

1.3 Exercise 3

1.3.1 (a)

If the expression $ab + cd + \cdot$ in postfix notation is converted to infix notation, what is the result?

Proof. $(a + b) \cdot (c + d)$ □

1.3.2 (b)

Let $\Sigma = \{1, 2, *, /\}$ and let L be the set of all strings over Σ obtained by writing first a number (1 or 2), then a second number (1 or 2), which can be the same as the first one, and finally an operation (denoted * or /, where * indicates multiplication and / indicates division). Then L is a set of postfix, or reverse Polish, expressions. List all the elements of L between braces, and evaluate the resulting expressions.

Proof. $L = \{11*, 11/, 12*, 12/, 21*, 21/, 22*, 22/\}$ These evaluate to:
$1 \cdot 1 = 1, 1/1 = 1, 1 \cdot 2 = 2, 1/2 = \frac{1}{2}, 2 \cdot 1 = 2, 2/1 = 2, 2 \cdot 2 = 4, 2/2 = 1.$ □

In $4 - 6$, describe L_1L_2, $L_1 \cup L_2$, and $(L_1 \cup L_2)^*$ for the given languages L_1 and L_2.

1.4 Exercise 4

L_1 is the set of all strings of a's and b's that start with an a and contain only that one a; L_2 is the set of all strings of a's and b's that contain an even number of a's.

Proof. L_1L_2 is the set of all strings of a's and b's that start with an a and contain an odd number of a's. $L_1 \cup L_2$ is the set of all strings of a's and b's that contain an even number of a's or that start with an a and contain only that one a. (Note that because 0 is an even number, both λ and b are in $L_1 \cup L_2$.) $(L_1 \cup L_2)^*$ is the set of all strings of a's and b's. The reason is that a and b are both in $L_1 \cup L_2$, and thus every string in a and b is in $(L_1 \cup L_2)^*$. □

1.5 Exercise 5

L_1 is the set of all strings of a's, b's, and c's that contain no c's and have the same number of a's as b's; L_2 is the set of all strings of a's, b's, and c's that contain no a's or b's.

Proof. Note that $\lambda \in L_1$ since 0 is a number and λ contains the same number of, namely 0 of, a's and b's, and contains no c's. Similarly $\lambda \in L_2$ because λ contains no a's or b's.

L_1L_2 is the set of all strings of a's, b's and c's that contain the same number of a's and b's, where all the c's are to the right of all a's and b's. Since $\lambda \in L_1$ and $\lambda \in L_2$, $\lambda \in L_1L_2$ too, and moreover $L_1 \subseteq L_1L_2$ (due to $l_1\lambda$ for all $l_1 \in L_1$) and $L_2 \subseteq L_1L_2$ (due to λl_2 for all $l_2 \in L_2$).

$L_1 \cup L_2$ is the set of all strings of a's, b's and c's that either contain the same number of a's and b's and no c's, or contain no a's or b's.

$(L_1 \cup L_2)^*$ is the set of all strings of a's, b's and c's that contain the same number of a's and b's. □

1.6 Exercise 6

L_1 is the set of all strings of 0's and 1's that start with a 0; L_2 is the set of all strings of 0's and 1's that end with a 0.

Proof. L_1L_2 is the set of all strings of 0's and 1's that start with a 0 and end with a 0. (λ not included.)

$L_1 \cup L_2$ is the set of all strings of 0's and 1's that either start with a 0, or end with a 0 (or both). (λ not included.)

$(L_1 \cup L_2)^*$ is the set of all strings of 0's and 1's that either start with a 0, or end with a 0, including λ. $\square$

In $7-9$, add parentheses to emphasize the order of precedence in the given expressions.

1.7 Exercise 7

$(a|b^*b)(a^*|ab)$

Proof. $(a|((b^*)b))((a^*)|(ab))$ $\square$

1.8 Exercise 8

$0^*1|0(0^*1)^*$

Proof. $((0^*)1)|(0(((0^*)1)^*))$ $\square$

1.9 Exercise 9

$(x|yz^*)^*(yx|(yz)^*z)$

Proof. $((x|(y(z^*)))^*)((yx)|(((yz)^*)z))$ $\square$

In $10-12$, use the rules about order of precedence to eliminate the parentheses in the given regular expression.

1.10 Exercise 10

$((a(b^*))|(c(b^*)))((ac)|(bc))$

Proof. $(ab^*|cb^*)(ac|bc)$ $\square$

1.11 Exercise 11

$(1(1^*))|((1(0^*))|((1^*)1))$

Proof. $11^*|(10^*|1^*1)$ $\square$

1.12 Exercise 12

$(xy)(((x^*)y)^*)|(((yx)|y)(y^*))$

Proof. $xy(x^*y)^*|(yx|y)y^*$ $\square$

In $13-15$, use set notation to derive the language defined by the given regular expression. Assume $\Sigma = \{a, b, c\}$.

1.13 Exercise 13

$\lambda|ab$

Proof. $L(\lambda|ab) = L(\lambda)\cup L(ab) = \{\lambda\}\cup L(a)L(b) = \{\lambda\}\cup\{xy \,|\, x \in L(a) \text{ and } y \in L(b)\} = \{\lambda\} \cup \{xy \,|\, x \in \{a\} \text{ and } y \in \{b\}\} = \{\lambda\} \cup \{ab\} = \{\lambda, ab\}$ □

1.14 Exercise 14

$\varnothing|\lambda$

Proof. $L(\varnothing|\lambda) = L(\varnothing) \cup L(\lambda) = \varnothing \cup \{\lambda\} = \{\lambda\}$ □

1.15 Exercise 15

$(a|b)c$

Proof. $L((a|b)c) = L(a|b)L(c) = (L(a) \cup L(b))L(c) = (\{a\} \cup \{b\})\{c\} = \{a, b\}\{c\} = \{xy \,|\, x \in \{a, b\} \text{ and } y \in \{c\}\} = \{ac, bc\}$ □

In 16–18, write five strings that belong to the language defined by the given regular expression.

1.16 Exercise 16

$0^*1(0^*1^*)^*$

Proof. Here is a sample of five strings out of infinitely many: 0101, 1, 01, 10000, and 011100. □

1.17 Exercise 17

$b^*|b^*ab^*$

Proof. $b, a, bab, babb, bbab$ □

1.18 Exercise 18

$x^*(yxxy|x)^*$

Proof. $yxxy, yxxyyxxy, xyxxy, xx, xxyxxyxx$ □

In $19 - 21$**, use words to describe the language defined by the given regular expression.**

1.19 Exercise 19

$b^*ab^*ab^*a$

Proof. The language consists of all strings of a's and b's that contain exactly three a's and end in an a. □

1.20 Exercise 20

$1(0|1)^*00$

Proof. All strings that begin with a 1 and end in 00. □

1.21 Exercise 21

$(x|y)y(x|y)^*$

Proof. All strings that start with either xy or yy, then followed by any string made up of x's and y's. □

In $22 - 24$, indicate whether the given strings belong to the language defined by the given regular expression. Briefly justify your answers.

1.22 Exercise 22

Expression: $(b|l)a(a|b) * a(b|l)$, strings: $aaaba, baabb$

Proof. $aaaba$ is in the language but $baabb$ is not because if a string in the language contains a b to the right of the left-most a, then it must contain another a to the right of all the b's. □

1.23 Exercise 23

Expression: $(x^*y|zy^*)^*$, strings: $zyyxz, zyyzy$

Proof. $zyyxz$ is not in the language because, due to the rule x^*y being the only rule that includes an x, the last x in a string must be followed by a y.

$zyyzy$ is in the language because, zyy can be obtained from zy^*, then zy can also be obtained from zy^*, and they can be concatenated due to the outer *. □

1.24 Exercise 24

Expression: $(01^*2)^*$, strings: 120, 01202

Proof. 120 is not in the language because, every nonempty string must contain a 0 at the start.

01202 is in the language because, 012 can be obtained from 01*2, then 02 can also be obtained from 01*2, then they can be concatenated due to the outer *. ☐

In 25 – 27, find a regular expression that defines the given language.

1.25 Exercise 25

The language consisting of all strings of 0's and 1's with an odd number of 1's. (Such a string is said to have odd parity.)

Proof. One solution is $0^*10^*(0^*10^*10^*)^*$. ☐

1.26 Exercise 26

The language consisting of all strings of a's and b's in which the third character from the end is a b.

Proof. One solution is $(a|b)^*b(aa|ab|ba|bb)$. ☐

1.27 Exercise 27

The language consisting of strings of x's and y's in which the elements in every pair of x's are separated by at least one y.

Proof. We can think of the string as follows: start with 0 or more y's, followed by one x and one y (because two x's cannot be next to each other) and 0 or more y's, which can be repeated as many times, then finally followed by either λ or one x.

So one solution is $y^*(xyy^*)^*(\lambda|x)$. ☐

Let r, s, and t be regular expressions over $\Sigma = \{a, b\}$. In 28 – 30, determine whether the two regular expressions define the same language. If they do, describe the language. If they do not, give an example of a string that is in one of the languages but not the other.

1.28 Exercise 28

$(r|s)t$ and $rt|st$

Proof. $L((r|s)t) = L(r|s)L(t) = (L(r) \cup L(s))L(t)$
$= \{xy|(x \in L(r) \cup L(s)) \text{ and } y \in L(t)\} = \{xy|(x \in L(r) \text{ or } x \in L(s)) \text{ and } y \in L(t)\}$
$= \{xy|(x \in L(r) \text{ and } y \in L(t)) \text{ or } (x \in L(s) \text{ and } y \in L(t))\}$
$= \{xy|xy \in L(rt) \text{ or } xy \in L(st)\} = L(rt) \cup L(st) = L(rt|st)$

The language can be described as: $\{xy \,|\, x \in L(r) \cup L(s) \text{ and } y \in L(t)\}$. ☐

1.29 Exercise 29

$(rs)^*$ and r^*s^*

Proof. The string rr is in the second language but not in the first language. $\square$

1.30 Exercise 30

$(rs)^*$ and $((rs)^*)^*$

Proof. $(rs)^* = \{\lambda, rs, rsrs, rsrsrs, rsrsrsrs, \ldots\}$ and
$((rs)^*)^* = \{\lambda, rs, rsrs, rsrsrs, rsrsrsrs, \ldots\}^* = \{\lambda, rs, rsrs, rsrsrs, rsrsrsrs, \ldots\}$.

The two expressions define the same language. It can be described as: the set of strings that are 0 or more occurrences of rs concatenated. $\square$

In 31 – 39, write a regular expression to define the given set of strings. Use the shorthand notations given in the section whenever convenient. In most cases, your expression will describe other strings in addition to the given ones, but try to make your answer fit the given strings as closely as possible within reasonable space limitations.

1.31 Exercise 31

All words that are written in lowercase letters and start with the letters *pre* but do not consist of *pre* all by itself.

Proof. $pre[a-z]^+$ $\square$

1.32 Exercise 32

All words that are written in uppercase letters, and contain the letters BIO (as a unit) or $INFO$ (as a unit).

Proof. $[A-Z]^*(BIO|INFO)[A-Z]^*$ $\square$

1.33 Exercise 33

All words that are written in lowercase letters, end in ly, and contain at least five letters.

Proof. $[a-z]^+[a-z]^+[a-z]^+ly$ $\square$

1.34 Exercise 34

All words that are written in lowercase letters and contain at least one of the vowels a, e, i, o, or u.

Proof. $[a-z]^*(a|e|i|o|u)[a-z]^*$ $\square$

1.35 Exercise 35

All words that are written in lowercase letters and contain exactly one of the vowels a, e, i, o, or u.

Proof. $[\hat{}aeiou]^*(a|e|i|o|u)[\hat{}aeiou]^*$. Here $\hat{}aeiou$ means all the letters except those five: $bcdfghjklmnpqrstvwxyz$. □

1.36 Exercise 36

All words that are written in uppercase letters and do not start with one of the vowels A, E, I, O, or U but contain exactly two of these vowels next to each other.

Proof. $[\hat{}AEIOU][A-Z]^*[AEIOU]\{2\}[A-Z]^*$ □

1.37 Exercise 37

All United States social security numbers (which consist of three digits, a hyphen, two digits, another hyphen, and finally four more digits), where the final four digits start with a 3 and end with a 6.

Proof. $[0-9]\{3\}-[0-9]\{2\}-3[0-9]\{2\}6$ □

1.38 Exercise 38

All telephone numbers that have three digits, then a hyphen, then three more digits, then a hyphen, and then four digits, where the first three digits are either 800 or 888 and the last four digits start and end with a 2.

Proof. $(800|888)-[0-9]\{3\}-2[0-9]\{2\}2$ □

1.39 Exercise 39

All signed or unsigned numbers with or without a decimal point. A signed number has one of the prefixes + or −, and an unsigned number does not have a prefix. Represent the decimal point as
. to distinguish it from the single dot symbol for an arbitrary character.

Proof. $([+-]|\lambda)[0-9]^*(\backslash.|\lambda)[0-9]^*$ □

1.40 Exercise 40

Write a regular expression to perform a complete check to determine whether a given string represents a valid date from 1980 to 2079 written in one of the formats of Example 12.1.11. (During this period, leap years occur every four years starting in 1980.)

Proof. Leap years from 1980 to 2079 are 1980, 1984, 1988, 1992, 1996, 2000, 2004, and so forth. Note that the fourth digit is 0, 4, or 8 for the years whose third digit is even and that the fourth digit is 2 or 6 for the years whose third digit is odd. □

1.41 Exercise 41

Write a regular expression to define the set of strings of 0's and 1's with an even number of 0's and even number of 1's.

Proof. $(00|11)^*\left((01|10)(00|11)^*(01|10)(00|11)^*\right)^*$ □

2 Exercise Set 12.2

2.1 Exercise 1

Find the state of the vending machine in Example 12.2.1 after each of the following sequences of coins have been input.

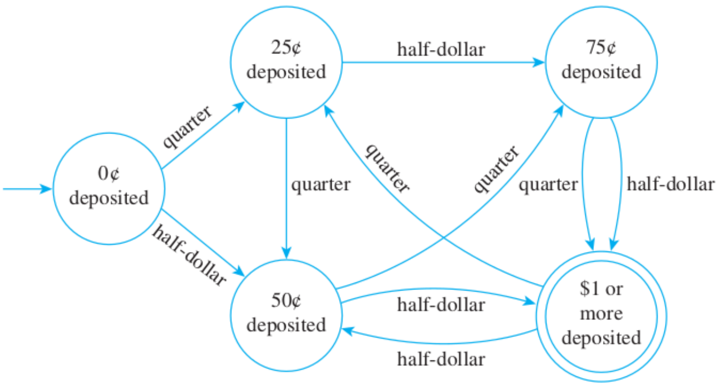


FIGURE 12.2.1 A Simple Vending Machine

2.1.1 (a)

Quarter, half-dollar, quarter

Proof. $1 or more deposited □

2.1.2 (b)

Quarter, half-dollar, half-dollar

Proof. $1 or more deposited □

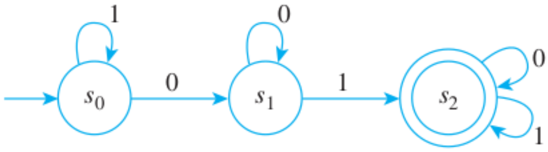
2.1.3 (c)

Half-dollar, quarter, quarter, quarter, half-dollar

Proof. 75¢ or more deposited □

In $2-7$, a finite-state automaton is given by a transition diagram. For each automaton:
a. Find its states.
b. Find its input symbols.
c. Find its initial state.
d. Find its accepting states.
e. Write its annotated next-state table.

2.2 Exercise 2

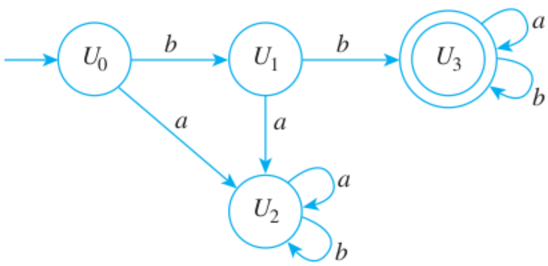


Proof. (a) s_0, s_1, s_2 (b) 0, 1 (c) s_0 (d) s_2 (e)

	State	Input 0	Input 1
$\rightarrow$	s_0	s_1	s_0
	s_1	s_1	s_2
◎	s_2	s_2	s_2

□

2.3 Exercise 3

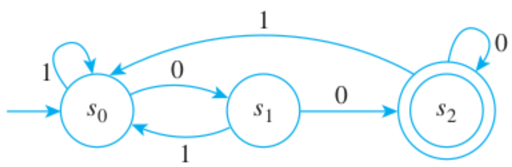


Proof. (a) U_0, U_1, U_2, U_3 (b) a, b (c) U_0 (d) U_3 (e)

		a	b
$\rightarrow$	U_0	U_2	U_1
	U_1	U_2	U_3
	U_2	U_2	U_2
$\circ$	U_3	U_3	U_3

□

2.4 Exercise 4

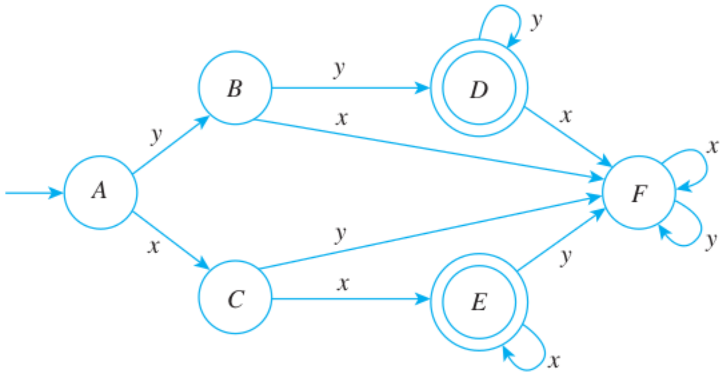


Proof. (a) s_0, s_1, s_2 (b) $0, 1$ (c) s_0 (d) s_2 (e)

		a	b
$\rightarrow$	s_0	s_1	s_0
	s_1	s_2	s_0
$\circ$	s_2	s_2	s_0

□

2.5 Exercise 5

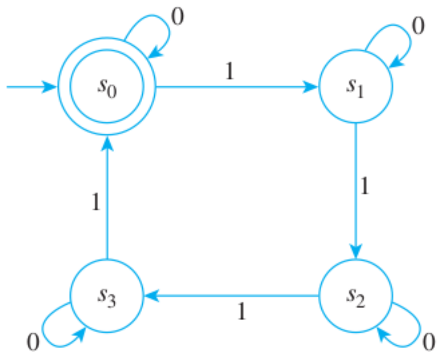


Proof. (a) A, B, C, D, E, F (b) x, y (c) A (d) D, E (e)

		Input	
	State	x	y
→	A	C	B
	B	F	D
	C	E	F
◎	D	F	D
◎	E	E	F
	F	F	F

□

2.6 Exercise 6



Proof. (a) s_0, s_1, s_2, s_3 (b) $0, 1$ (c) s_0 (d) s_0 (e)

		0	1
$\to\circ$	s_0	s_0	s_1
	s_1	s_1	s_2
	s_2	s_2	s_3
	s_3	s_3	s_0

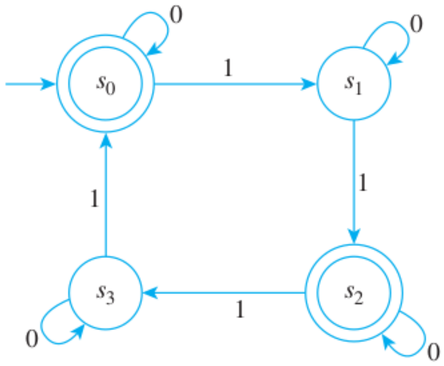
□

2.7 Exercise 7

Proof. (a) s_0, s_1, s_2, s_3 (b) 0, 1 (c) s_0 (d) s_0, s_2 (e)

			Input	
		State	0	1
→	◎	s_0	s_0	s_1
		s_1	s_1	s_2
	◎	s_2	s_2	s_3
		s_3	s_3	s_0

□



In 8 and 9, a finite-state automaton is given by an annotated next-state table. For each automaton:

a. Find its states.

b. Find its input symbols.

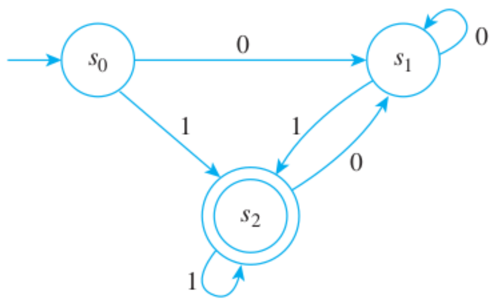
c. Find its initial state.

d. Find its accepting states.

e. Draw its transition diagram.

2.8 Exercise 8

			Input	
			0	1
State	→	s_0	s_1	s_2
		s_1	s_1	s_2
	⊚	s_2	s_1	s_2

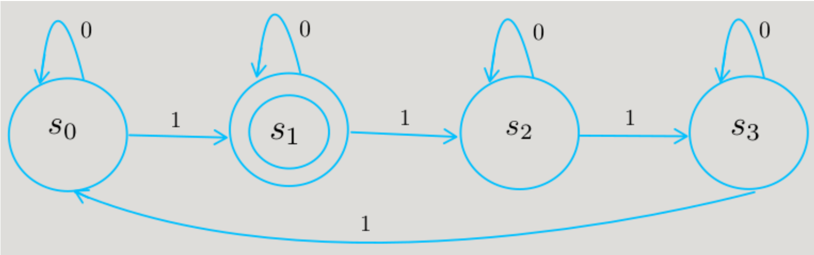


Proof. (a) s_0, s_1, s_2 (b) 0, 1 (c) s_0 (d) s_2 (e) □

2.9 Exercise 9

			Input	
			0	1
	→	s_0	s_0	s_1
	⊚	s_1	s_1	s_2
State		s_2	s_2	s_3
		s_3	s_3	s_0

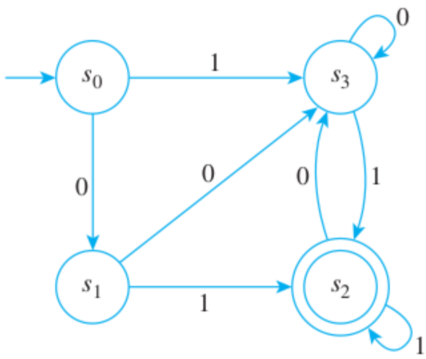
Proof. (a) s_0, s_1, s_2, s_3 (b) 0, 1 (c) s_0 (d) s_1

(e) 

□

2.10 Exercise 10

A finite-state automaton A, given by the transition diagram below, has next-state function N and eventual- state function N^*.



2.10.1 (a)

Find $N(s_1, 1)$ and $N(s_0, 1)$.

Proof. $N(s_1, 1) = s_2, N(s_0, 1) = s_3$ □

2.10.2 (b)

Find $N(s_2, 0)$ and $N(s_1, 0)$.

Proof. $N(s_2, 0) = s_3$ and $N(s_1, 0) = s_3$ □

2.10.3 (c)

Find $N^*(s_0, 10011)$ and $N^*(s_1, 01001)$.

Proof. $N^*(s_0, 10011) = s_2, N^*(s_1, 01001) = s_2$ □

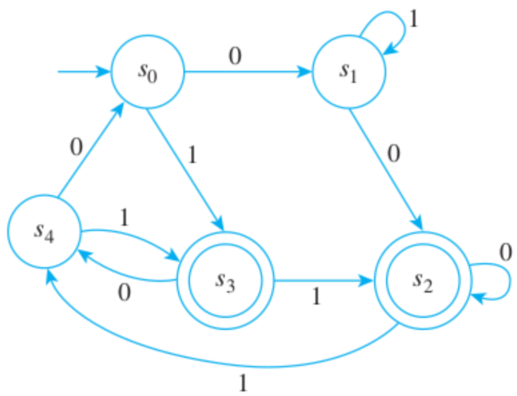
2.10.4 (d)

Find $N^*(s_2, 11010)$ and $N^*(s_0, 01000)$.

Proof. $N^*(s_2, 11010) = s_3$ and $N^*(s_0, 01000) = s_3$. □

2.11 Exercise 11

A finite-state automaton A, given by the transition diagram below, has next-state function N and eventual- state function N^*.



2.11.1 (a)

Find $N(s_3, 0)$ and $N(s_2, 1)$.

Proof. $N(s_3, 0) = s_4, N(s_2, 1) = s_4$ □

2.11.2 (b)

Find $N(s_0, 0)$ and $N(s_4, 1)$.

Proof. $N(s_0, 0) = s_1$ and $N(s_4, 1) = s_3$. □

2.11.3 (c)

Find $N^*(s_0, 010011)$ and $N^*(s_3, 01101)$.

Proof. $N^*(s_0, 010011) = s_3, N^*(s_3, 01101) = s_4$ □

2.11.4 (d)

Find $N^*(s_0, 1111)$ and $N^*(s_2, 00111)$.

Proof. $N^*(s_0, 1111) = s_3$ and $N^*(s_2, 00111) = s_2$. □

Note that multiple correct answers exist for part (d) of exercises 12 and 13, part (b) of exercises $14 - 19$, and for exercises $20 - 48$.

2.12 Exercise 12

Consider again the finite-state automaton of exercise 2.

2.12.1 (a)

To what state does the automaton go when the symbols of the following strings are input to it in sequence, starting from the initial state?

(i) 1110001 (ii) 0001000 (iii) 11110000

Proof. (i) s_2 (ii) s_2 (iii) s_1 □

2.12.2 (b)

Which of the strings in part (a) send the automaton to an accepting state?

Proof. those in (i) and (ii) but not (iii) □

2.12.3 (c)

What is the language accepted by the automaton?

Proof. The language accepted by this automaton is the set of all strings of 0's and 1's that contain at least one 0 followed (not necessarily immediately) by at least one 1. □

2.12.4 (d)

Find a regular expression that defines the language.

Proof. $1^*00^*1(0|1)^*$ □

2.13 Exercise 13

Consider again the finite-state automaton of exercise 3.

2.13.1 (a)

To what state does the automaton go when the symbols of the following strings are input to it in sequence, starting from the initial state?

(i) *bb* (ii) *aabbbaba* (iii) *babbbbbabaa* (iv) *bbaaaabaa*

Proof. (i) U_3 (ii) U_2 (iii) U_2 (iv) U_3 □

2.13.2 (b)

Which of the strings in part (a) send the automaton to an accepting state?

Proof. those in (i) and (iv) but not (ii) or (iii) □

2.13.3 (c)

What is the language accepted by the automaton?

Proof. All strings of a's and b's starting with at least two b's. □

2.13.4 (d)

Find a regular expression that defines the language.

Proof. $bb(a|b)^*$ □

In each of 14 – 19, (a) find the language accepted by the automaton in the referenced exercise, and (b) find a regular expression that defines the same language.

2.14 Exercise 14

Exercise 4

2.14.1 (a)

Proof. The language accepted by this automaton is the set of all strings of 0's and 1's that end in 00. □

2.14.2 (b)

Proof. $(0|1)^*00$ □

2.15 Exercise 15

Exercise 5

2.15.1 (a)

Proof. The language accepted by this automaton is the set of all strings of x's and y's of length at least two that consist either entirely of x's or entirely of y's. □

2.15.2 (b)

Proof. $xxx^*|yyy^*$ □

2.16 Exercise 16

Exercise 6

2.16.1 (a)

Proof. The language accepted by this automaton is the set of all strings of 0's and 1's where the number of 1's is a multiple of 4. (This includes when the number of 1's is zero, i.e. strings consisting only of 0's.) □

2.16.2 (b)

Proof. $0^*|(0^*10^*10^*10^*10^*)^*$ □

2.17 Exercise 17

Exercise 7

2.17.1 (a)

Proof. The language accepted by this automaton is the set of all strings of 0's and 1's with the following property: If n is the number of 1's in the string, then $n \mod 4 = 0$ or $n \mod 4 = 2$. This is equivalent to saying that n is even. □

2.17.2 (b)

Proof. $0^*|(0^*10^*10^*)^*$ □

2.18 Exercise 18

Exercise 8

2.18.1 (a)

Proof. The language accepted by this automaton is the set of all strings of 0's and 1's that end in 1. □

2.18.2 (b)

Proof. $(0|1)^*1$ □

2.19 Exercise 19

Exercise 9

2.19.1 (a)

Proof. The language accepted by this automaton is set of all strings of 0's and 1's, with the property that if n is the number of 1's in the string, then $n \mod 4 = 1$. □

2.19.2 (b)

Proof. (0*10*10*10*1)*1(0*10*10*10*1)* □

In each of 20 – 28, (a) design an automaton with the given input alphabet that accepts the given set of strings, and (b) find a regular expression that defines the language accepted by the automaton.

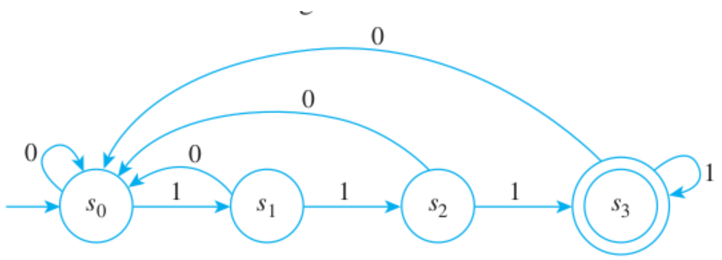
2.20 Exercise 20

Input alphabet = $\{0, 1\}$; Accepts the set of all strings for which the final three input symbols are 1.

2.20.1 (a)

Proof. Call the automaton being constructed A. Acceptance of a string by A depends on the values of three consecutive inputs. Thus A requires at least four states:
s_0: initial state
s_1: state indicating that the last input character was a 1
s_2: state indicating that the last two input characters were 1's
s_3: state indicating that the last three input characters were 1's, the acceptance state

If a_0 is input to A when it is in state s_0, no progress is made toward achieving a string of three consecutive 1's. Hence A should remain in state s_0. If a_1 is input to A when it is in state s_0, it goes to state s_1, which indicates that the last input character of the string is a 1. From state s_1, A goes to state s_2 if a 1 is input. This indicates that the last two characters of the string are 1's. But if a_0 is input, A should return to s_0 because the wait for a string of three consecutive 1's must start over again. When A is in state s_2 and a 1 is input, then a string of three consecutive 1's is achieved, so A should go to state s_3. If a 0 is input when A is in state s_2, then progress toward accumulating a sequence of three consecutive 1's is lost, so A should return to s_0. When A is in a state s_3 and a 1 is input, then the final three symbols of the input string are 1's, and so A should stay in state s_3. If a 0 is input when A is in state s_3, then A should return to state s_0 to await the input of more 1's. Thus the transition diagram is as follows:



□

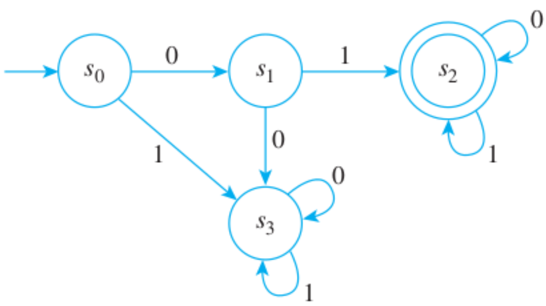
2.20.2 (b)

Proof. $(0|1) * 111$ □

2.21 Exercise 21

Input alphabet $= \{0, 1\}$; Accepts the set of all strings that start with 01.

2.21.1 (a)



Proof. □

2.21.2 (b)

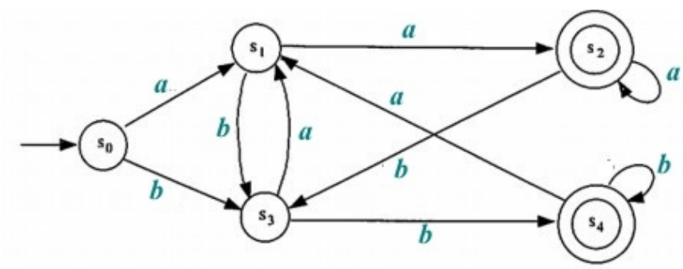
Proof. $01(0|1)^*$ □

2.22 Exercise 22

Input alphabet $= \{a, b\}$; Accepts the set of all strings of length at least 2 for which the final two input symbols are the same.

2.22.1 (a)

Proof. Use five states: s_0 (the initial state), s_1 (the state indicating that the previous input symbol was an a), s_2 (the state indicating that the previous input symbol was a b), s_3 (the state indicating that the previous two input symbols were a's), and s_4 (the state indicating that the previous two input symbols were b's).



□

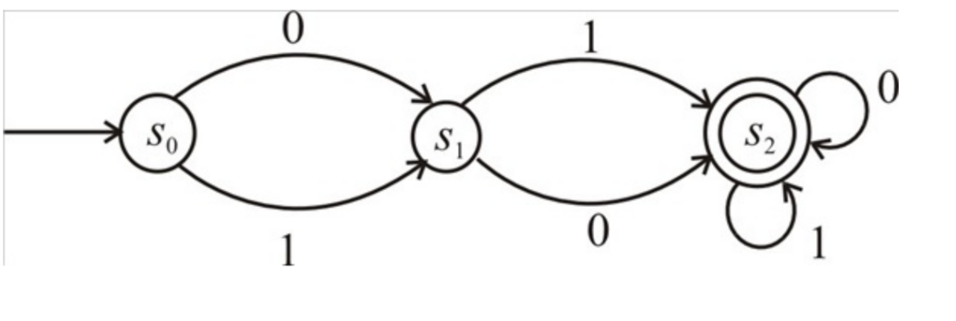
2.22.2 (b)

Proof. $(a|b)^*(aa|bb)$ □

2.23 Exercise 23

Input alphabet $= \{0, 1\}$; Accepts the set of all strings that start with 01 or 10.

2.23.1 (a)



Proof. □

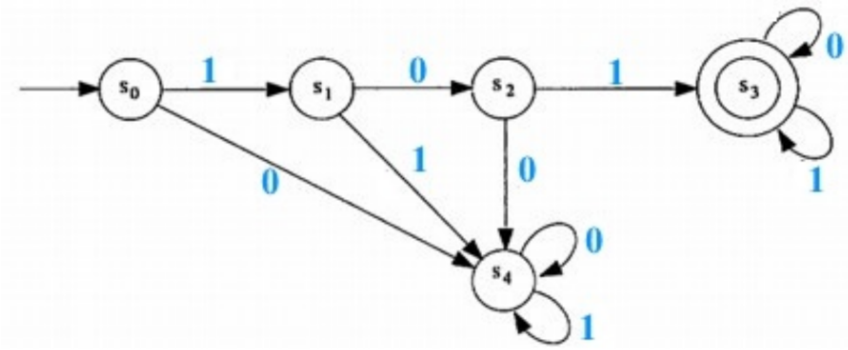
2.23.2 (b)

Proof. $(01|10)(0|1)^*$ □

2.24 Exercise 24

Input alphabet $= \{0, 1\}$; Accepts the set of all strings that start with 101.

2.24.1 (a)



Proof. □

2.24.2 (b)

Proof. $101(0|1)^*$ □

2.25 Exercise 25

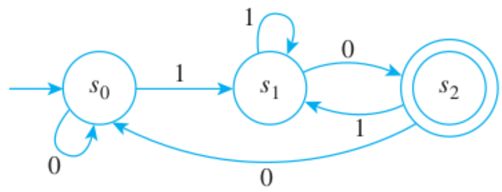
Input alphabet $= \{0, 1\}$; Accepts the set of all strings that end in 10.

2.25.1 (a)

Proof. □

2.25.2 (b)

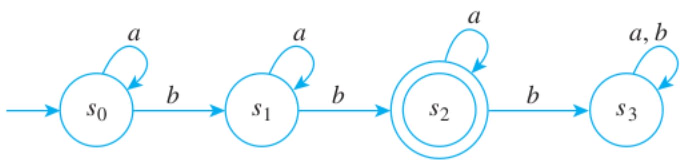
Proof. $(0|1)^*10$ □



2.26 Exercise 26

Input alphabet $= \{a, b\}$; Accepts the set of all strings that contain exactly two b's.

2.26.1 (a)



Proof. □

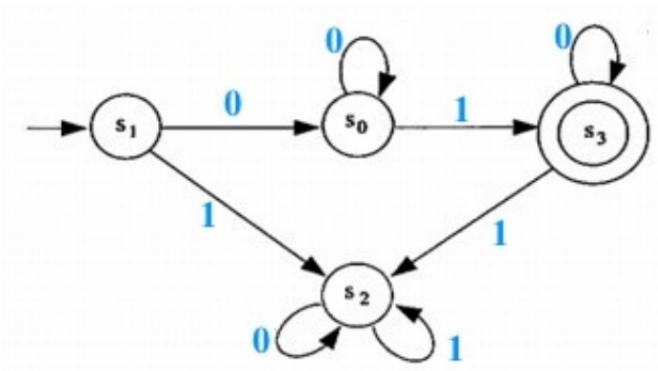
2.26.2 (b)

Proof. $a^*ba^*ba^*$ □

2.27 Exercise 27

Input alphabet $= \{0, 1\}$; Accepts the set of all strings that start with 0 and contain exactly one 1.

2.27.1 (a)



Proof. □

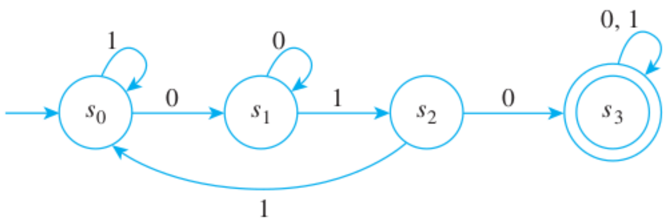
2.27.2 (b)

Proof. 00^*10^* □

2.28 Exercise 28

Input alphabet $= \{0, 1\}$; Accepts the set of all strings that contain the pattern 010.

2.28.1 (a)



Proof. □

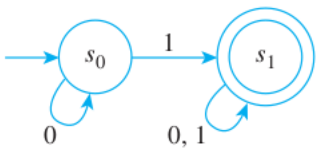
2.28.2 (b)

Proof. $(0|1)^*101(0|1)^*$ □

In 29 – 47, design a finite-state automaton to accept the language defined by the regular expression in the referenced exercise from Section 12.1.

2.29 Exercise 29

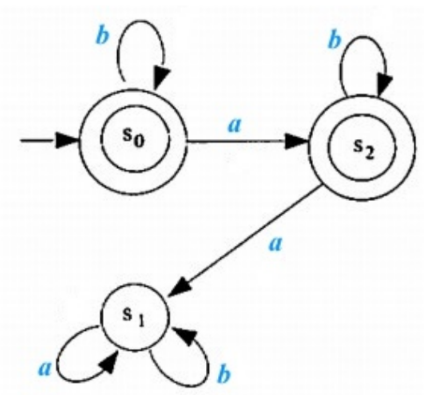
Exercise 16: $0^*1(0^*1^*)^*$



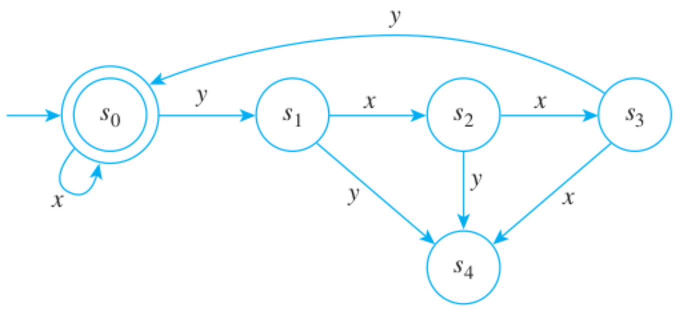
Proof. □

2.30 Exercise 30

Exercise 17: $b^*|b^*ab^*$



Proof. □



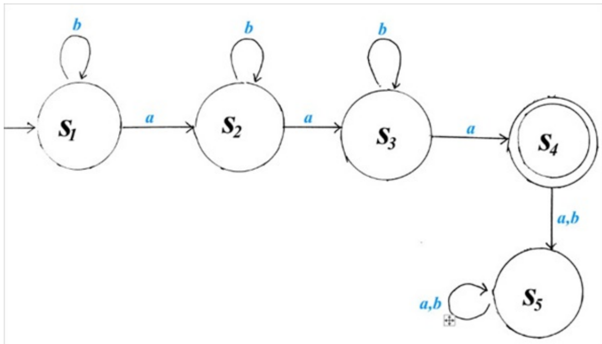
2.31 Exercise 31

Exercise 18: $x^*(yxxy|x)^*$

Proof. Errata says: add two arrow loops from s_4 to itself, one for x and one for y. □

2.32 Exercise 32

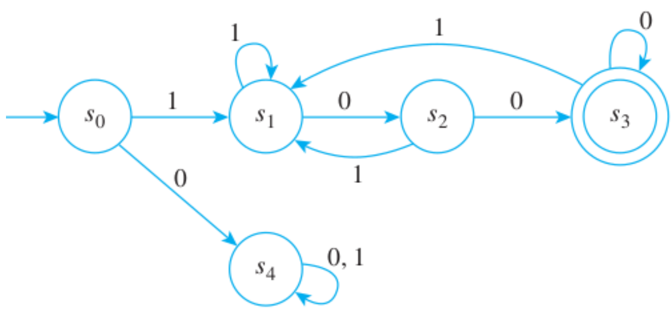
Exercise 19: $b^*ab^*ab^*a$



Proof. □

2.33 Exercise 33

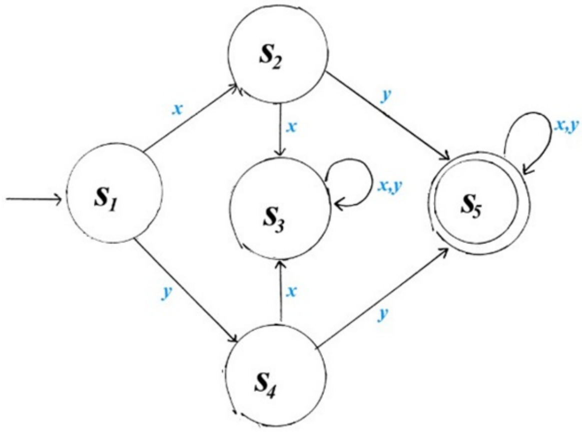
Exercise 20: $1(0|1)^*00$



Proof. □

2.34 Exercise 34

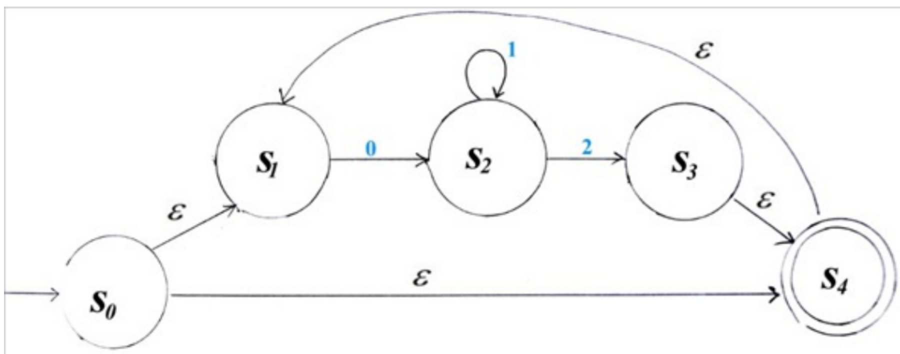
Exercise 21: $(x|y)y(x|y)^*$



Proof. □

2.35 Exercise 35

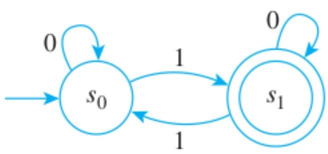
Exercise 24: $(01^*2)^*$



Proof. □

2.36 Exercise 36

Exercise 25: $0^*10^*(0^*10^*10^*)^*$.

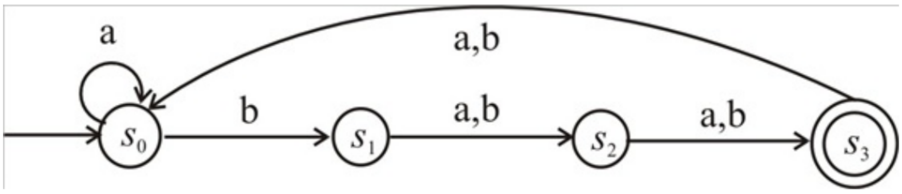


Proof. □

2.37 Exercise 37

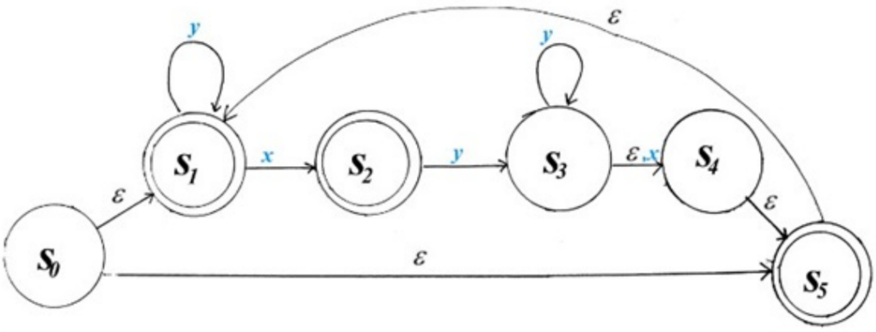
Exercise 26: $(a|b)^*b(aa|ab|ba|bb)$

Proof. □



2.38 Exercise 38

Exercise 27: $y^*(xyy^*)^*(\lambda|x)$.

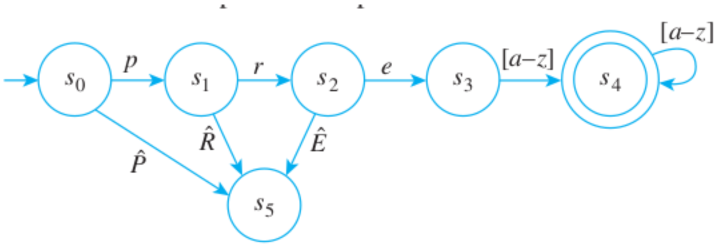


Proof. □

2.39 Exercise 39

Exercise 31: $pre[a-z]^+$

Proof. Let $\hat{P}$ denote a list of all letters of a lowercase alphabet except p, $\hat{R}$ denote a list of all the letters of a lowercase alphabet except r, and $\hat{E}$ denote a list of all the letters of a lowercase alphabet except e.



□

2.40 Exercise 40

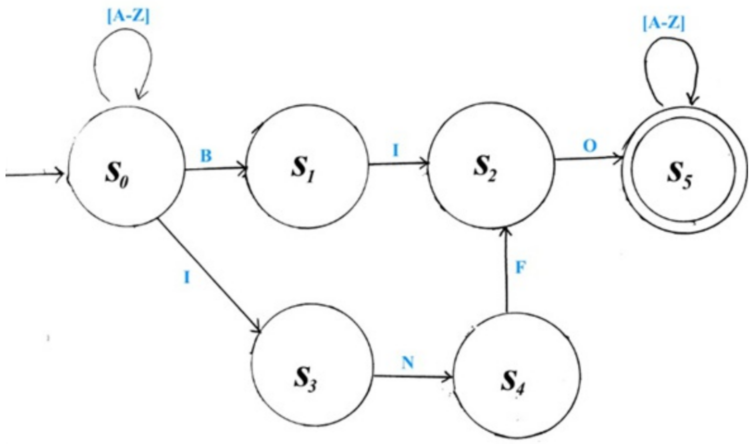
Exercise 32: $[A-Z]^*(BIO|INFO)[A-Z]^*$

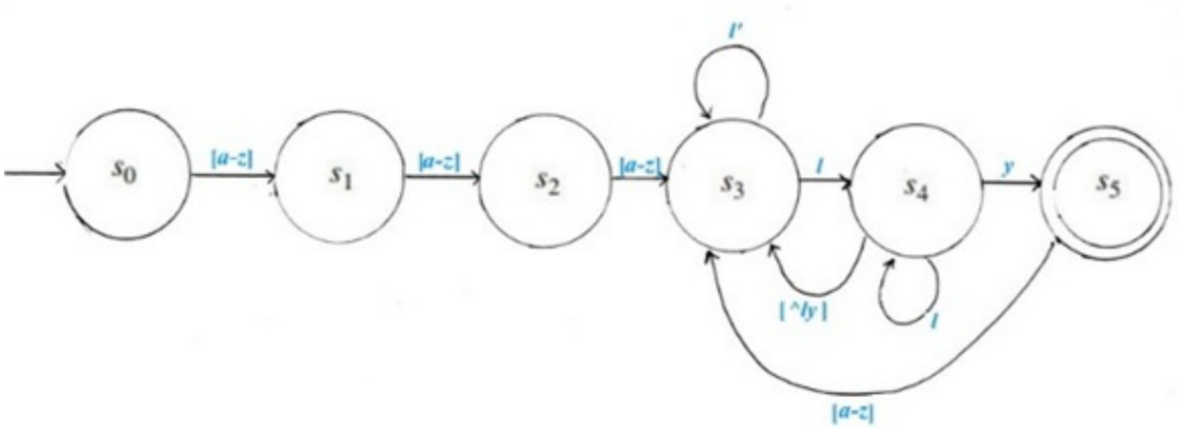
Proof. □

2.41 Exercise 41

Exercise 33: $[a-z]^+[a-z]^+[a-z]^+ly$

Proof. □

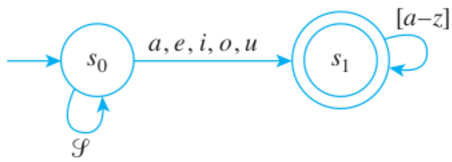




2.42 Exercise 42

Exercise 34: $[a-z]^*(a|e|i|o|u)[a-z]^*$

Proof. Let $\mathscr{S}$ denote a list of all the consonants in a lowercase alphabet.



□

2.43 Exercise 43

Exercise 35: $[\hat{\ }aeiou]^*(a|e|i|o|u)[\hat{\ }aeiou]^*$

Proof. *???* □

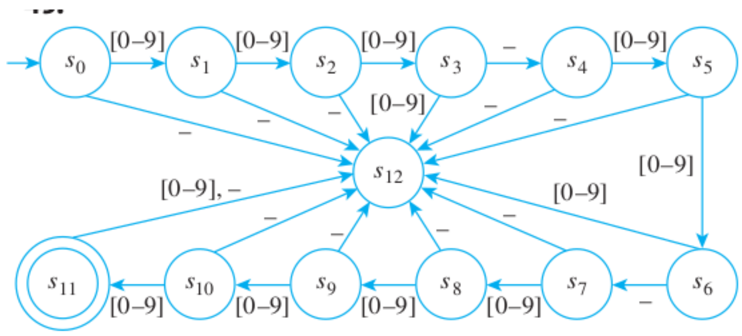
2.44 Exercise 44

Exercise 36: $[\hat{\ }AEIOU][A-Z]^*[AEIOU]\{2\}[A-Z]^*$

Proof. *???* □

2.45 Exercise 45

Exercise 37: $[0-9]\{3\}-[0-9]\{2\}-3[0-9]\{2\}6$



Proof. □

2.46 Exercise 46

Exercise 38: $(800|888)-[0-9]\{3\}-2[0-9]\{2\}2$

Proof. ??? □

2.47 Exercise 47

Exercise 39: $([+-]|\lambda)[0-9]^*(\backslash.|\lambda)[0-9]^*$

Proof. ??? □

2.48 Exercise 48

A simplified telephone switching system allows the following strings as legal telephone numbers. Design a finite-state automaton to recognize all the legal telephone numbers in (a), (b), and (c). Include an "error state" for invalid telephone numbers.

2.48.1 (a)

A string of seven digits in which neither of the first two digits is a 0 or 1 (a local call string).

Proof. A regular expression is $[2-9]2[0-9]5$. □

2.48.2 (b)

A 1 followed by a three-digit area code string (any digit except 0 or 1 followed by a 0 or 1 followed by any digit) followed by a seven-digit local call string.

Proof. A regular expression is $1[2-9](0|1)[0-9][2-9]2[0-9]5$. Here we are using the regular expression from part (a) for the local call string. □

2.48.3 (c)

A 0 alone or followed by a three-digit area code string plus a seven-digit local call string.

Proof. A regular expression is $0(([2-9](0|1)[0-9])?)[2-9]2[0-9]5$ (using the local call string and the area code string). $\square$

2.49 Exercise 49

Write a computer algorithm that simulates the action of the finite-state automaton of exercise 2 by mimicking the action of the transition diagram.

Proof. *???* $\square$

2.50 Exercise 50

Write a computer algorithm that simulates the action of the finite-state automaton of exercise 8 by repeated application of the next-state function.

Proof. *???* $\square$

2.51 Exercise 51

Let L be the language consisting of all strings of the form $a^m b^n$, where m and n are positive integers and $m \geq n$. Show that there is no finite-state automaton that accepts L.

Proof. This proof is virtually identical to that of Example 12.2.8. Just take p and q in that proof so that $p > q$. From the fact that A accepts $a^p b^p$, you can deduce that A accepts $a^q b^p$. Since $p > q$, this string is not in L. $\square$

2.52 Exercise 52

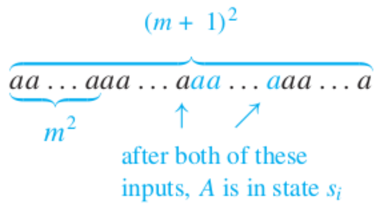
Let L be the language consisting of all strings of the form $a^m b^n$, where m and n are positive integers and $m \leq n$. Show that there is no finite-state automaton that accepts L.

Proof. *???* $\square$

2.53 Exercise 53

Let L be the language consisting of all strings of the form a^n, where $n = m^2$, for some positive integer m. Show that there is no finite-state automaton that accepts L.

Proof. Suppose the automaton A has N states. Choose an integer m such that $(m+1)^2 - m^2 > N$. Consider strings of a's of lengths between m^2 and $(m+1)^2$. Since there are more strings than states, at least two strings must send A to the same state s_i:



It follows (by removing the a's shown in color) that the automaton must accept a string of the form a^k, where $m^2 < k < (m+1)^2$. □

2.54 Exercise 54

2.54.1 (a)

Let A be a finite-state automaton with input alphabet Σ, and suppose $L(A)$ is the language accepted by A. The complement of $L(A)$ is the set of all strings over Σ that are not in $L(A)$. Show that the complement of a regular language is regular by proving the following: If $L(A)$ is the language accepted by a finite-state automaton A, then there is a finite-state automaton A' that accepts the complement of $L(A)$.

Proof. ??? □

2.54.2 (b)

Show that the intersection of any two regular languages is regular as follows: First prove that if $L(A_1)$ and $L(A_2)$ are languages accepted by automata A_1 and A_2, respectively, then there is an automaton A that accepts $(L(A_1))^c \cup (L(A_2))^c$. Then use one of De Morgan's laws for sets, the double complement law for sets, and the result of part (a) to prove that there is an automaton that accepts $L(A_1) \cap L(A_2)$.

Proof. ??? □

3 Exercise Set 12.3

3.1 Exercise 1

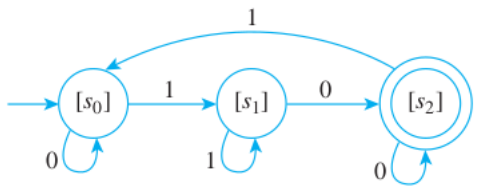
3.1.1 (a)

Proof. 0-equivalence classes: $\{s_0, s_1, s_3, s_4\}, \{s_2, s_5\}$
1-equivalence classes: $\{s_0, s_3\}, \{s_1, s_4\}, \{s_2, s_5\}$
2-equivalence classes: $\{s_0, s_3\}, \{s_1, s_4\}, \{s_2, s_5\}$

□

3.1.2 (b)

Proof. □



3.2 Exercise 2

3.2.1 ()

Proof. □

3.3 Exercise 3

3.3.1 ()

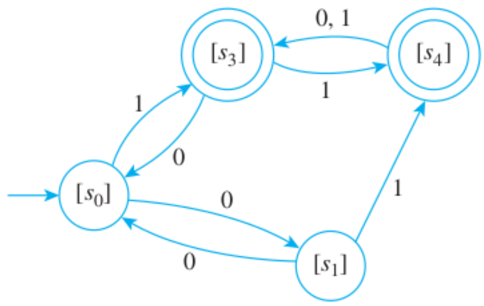
Proof. □

3.4 Exercise 4

3.4.1 (a)

Proof. 0-equivalence classes: $\{s_0, s_1, s_2\}, \{s_3, s_4, s_5\}$
1-equivalence classes: $\{s_0, s_1, s_2\}, \{s_3, s_5\}, \{s_4\}$
2-equivalence classes: $\{s_0, s_2\}, \{s_1\}, \{s_3, s_5\}, \{s_4\}$
3-equivalence classes: $\{s_0, s_2\}, \{s_1\}, \{s_3, s_5\}, \{s_4\}$ □

3.4.2 (b)



Proof. □

3.5 Exercise 5

3.5.1 ()

Proof. □

3.6 Exercise 6

3.6.1 (a)

Proof. The 3-equivalence classes are $\{s_0\}, \{s_1\}, \{s_2\}, \{s_3\}, \{s_4\}, \{s_5\}$, and $\{s_6\}$. □

3.6.2 (b)

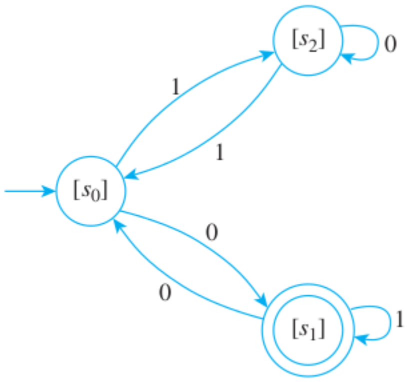
Proof. □

3.7 Exercise 7

3.7.1 ()

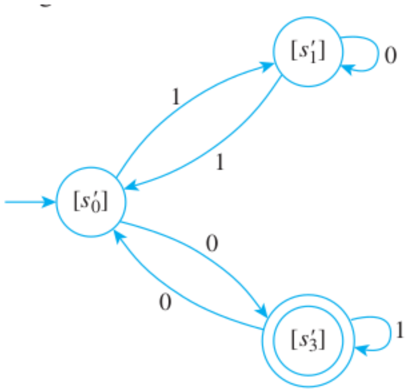
Proof. Yes. For A:
0-equivalence classes: $\{s_0, s_2\}, \{s_1, s_3\}$
1-equivalence classes: $\{s_0\}, \{s_2\}, \{s_1, s_3\}$
2-equivalence classes: $\{s_0\}, \{s_2\}, \{s_1, s_3\}$

Transition diagram for $\overline{A}$:



For A':
0-equivalence classes: $\{s'_0, s'_1, s'_2\}, \{s'_3\}$
1-equivalence classes: $\{s'_0, s'_2\}, \{s'_1\}, \{s'_3\}$
2-equivalence classes: $\{s'_0, s'_2\}, \{s'_1\}, \{s'_3\}$

Transition diagram for $\overline{A'}$:



Except for the labeling of the states, the transition diagrams for A and A' are identical. Hence A and A accept the same language, and so, by Theorem 12.3.3, A and A' also accept the same language. Thus A and A' are equivalent automata. □

3.8 Exercise 8

3.8.1 ()

Proof. □

3.9 Exercise 9

3.9.1 ()

Proof. For A:
0-equivalence classes: $\{s_1, s_2, s_4, s_5\}, \{s_0, s_3\}$
1-equivalence classes: $\{s_1, s_2\}, \{s_4, s_5\}, \{s_0, s_3\}$
2-equivalence classes: $\{s_1\}, \{s_2\}, \{s_4, s_5\}, \{s_0, s_3\}$
3-equivalence classes: $\{s_1\}, \{s_2\}, \{s_4, s_5\}, \{s_0, s_3\}$

Therefore, the states of $\overline{A}$ are the 3-equivalence classes of A.

For A':
0-equivalence classes: $\{s'_2, s'_3, s'_4, s'_5\}, \{s'_0, s'_1\}$
1-equivalence classes: $\{s'_2, s'_3, s'_4, s'_5\}, \{s'_0, s'_1\}$

Therefore, the states of $\overline{A'}$ are the 1-equivalence classes of A'.

According to the text, two automata are equivalent if, and only if, their quotient automata are isomorphic, provided inaccessible states have first been removed. Now A and A' have no inaccessible states, and A has four states, whereas A' has only two states. Therefore, A and A' are not equivalent.

This result can also be obtained by noting, for example, that the string 11 is accepted by A' but not by A. □

3.10 Exercise 10

3.10.1 ()

Proof. □

3.11 Exercise 11

3.11.1 ()

Proof. Partial answer: Suppose A is a finite-state automaton with set of states S and relation R^* of $*$-equivalence of states. *[To show that R^* is an equivalence relation, we must show that R is reflexive, symmetric, and transitive.]*

Proof that R^* is symmetric: *[We must show that for all states s and t, if sR^*t then tR^*s.]*

Suppose that s and t are any states of A such that sR^*t. *[We must show that tR^*s.]* Since sR^*t, then for every input string w,

$N^*(s, w)$ is an accepting state $\iff$ $N^*(t, w)$ is an accepting state,

where N^* is the eventual-state function on A. It follows from the symmetry of the $\iff$ relation that for every input string w,

$N^*(t, w)$ is an accepting state $\iff$ $N^*(s, w)$ is an accepting state.

Hence tR^*s *[as was to be shown]*, and so R^* is symmetric. □

3.12 Exercise 12

3.12.1 ()

Proof. The proof is identical to the proof of property (12.3.1) given in the solution to exercise 11 provided every occurrence of "for each input string w" is replaced by "for each input string w of length less than or equal to k." □

3.13 Exercise 13

3.13.1 ()

Proof. By property (12.3.2), for each integer $k \geq 0$, k-equivalence is an equivalence relation. Now by Theorem 10.3.4, the distinct equivalence classes of an equivalence relation form a partition of the set on which the relation is defined. In this case, the relation is defined on the set of all states of the automaton. So the k-equivalence classes form a partition of the set of all states of the automaton. □

3.14 Exercise 14

3.14.1 ()

Proof. □

3.15 Exercise 15

3.15.1 ()

Proof. Hint 1: Suppose Ck is a particular but arbitrarily chosen k-equivalence class. You must show that there is a $(k-1)$-equivalence class C_{k-1} such that $C_k \subseteq C_{k-1}$.

If s is any element in C_k, then s is a state of the automaton. Now the $(k-1)$-equivalence classes partition the set of all states of the automaton into a union of mutually disjoint subsets, so $s \in C_{k-1}$ for some $(k-1)$-equivalence class C_{k-1}.

To show that $C_k \subseteq C_{k-1}$, you must show that for any state t, if $t \in C_k$, then $t \in C_{k-1}$. □

3.16 Exercise 16

3.16.1 ()

Proof. □

3.17 Exercise 17

3.17.1 ()

Proof. If $m < k$, then every input string of length less than or equal to m has length less than or equal to k. □

3.18 Exercise 18

3.18.1 ()

Proof. □

3.19 Exercise 19

3.19.1 ()

Proof. Suppose two states s and t are equivalent. We must show that for any input symbol m, the next-states $N(s, m)$ and $N(t, m)$ are equivalent. To do this, use the definition of equivalence and the fact that for any string w', input symbol m, and state s, $N^*(N(s, m), w') = N^*(s, mw')$. □