Logic & Formal Languages

Final, first quarter

1.

Non-recursive start symbol and elimination of lambda rules:

NULL = {A, C}
S' → S | .\
S → AB | B | BCS | BS
A → aA | a | C
B → bbB | b
C → cC | c

Elimination of chain rules:

CHAIN(S) = {S, B} CHAIN(A) = {A, C} CHAIN(B) = {B} CHAIN(C) = {C}

S' → S
S → AB | bbB| b | BCS | BS
A → aA | a | cC | c
B → bbB | b
C → cC | c

Elimination of useless symbols:

TERM = {S, A, B, C}
REACH = {S, A, B, C}
no useless symbols

Replacement of terminals:

S' → S
S → AB | B'B'B| b | BCS | BS
A → A'A | a | C'C | c
B → B'B'B | b
C → C'C | c
A' → a
B' → b
C' → c

(Final grammar) Replacement of rules longer than two:

S' → S
S → AB | B'R_1 | b | BR_2 | BS
R_1 → B'B
R_2 → CS
A → A'A | a | C'C | c
B → B'R_1 | b
C → C'C | c
A' → a
B' → b
C' → c

2.

Algorithm 4.4.2

Iteration	**TERM**	**PREV**
0	{B, C, D}	
1	{B, C, D, A, S}	{B, C, D}

all symbols terminating

Algorithm 4.4.4

Iteration	**REACH**	**PREV**	**NEW**
0	{S}	0	
1	{S, A, B, C}	{S}	{S}
2	{S, A, B, C, D}	{S, A, B, C}	{A, B, C}

all symbols reachable

Removal of chain rules:

CHAIN(S) = {S, A, C, D}	CHAIN(A) = {A, C, D}	CHAIN(B) = {B}	CHAIN(C) = {C}	CHAIN(D) = {D}

S → cC | c | dD | d | CB
A → cC | c | dD | d
B → bB | b
C → cC | c
D → dD | d

TERM = {S, A, B, C, D}
REACH = {S, C, D, B}

A is not in the set of reachable non-terminals. Therefore, A is useless.

3.

abbb

	1	**2**	**3**	**4**
1	{A}	{S, X}	{T}	0
2		{B}	0	0
3			{B}	0
4				{B}

aabbb

	1	**2**	**3**	**4**	**5**
1	{A}	0	0	{S, X}	{T}
2		{A}	{S, X}	{T}	0
3			{B}	0	0
4				{B}	0
5					{B}

4.

$S \rightarrow A \mid C$
$A \rightarrow BR_1 \mid aR_1 \mid B \mid a$
$B \rightarrow CbR_2 \mid Cb$
$R_1 \rightarrow aBR_1 \mid aCR_1 \mid aB \mid aC$
$R_2 \rightarrow bR_2 \mid b$

5.

Original grammar:
$S \rightarrow AB$
$A \rightarrow BB \mid CC$
$B \rightarrow AD \mid CA$
$C \rightarrow a$
$D \rightarrow b$

No direct left recursion
Variable replacement for ordering:
$S \rightarrow AB$
$A \rightarrow BB \mid CC$
$B \rightarrow BBD \mid CCD \mid CA$
$C \rightarrow a$
$D \rightarrow b$

Removal of direct left recursion:
$S \rightarrow AB$
$A \rightarrow BB \mid CC$
$B \rightarrow CCDR_1 \mid CAR_1 \mid CCD \mid CA$
$C \rightarrow a$
$D \rightarrow b$
$R_1 \rightarrow BDR_1 \mid BD$

Replacement of leading variables in B:

$S \rightarrow AB$
$A \rightarrow BB \mid CC$
$B \rightarrow aCDR_1 \mid aAR_1 \mid aCD \mid aA$
$C \rightarrow a$
$D \rightarrow b$
$R_1 \rightarrow BDR_1 \mid BD$

Replacement of leading variables in A:

$S \rightarrow AB$
$A \rightarrow aCDR_1B \mid aAR_1B \mid aCDB \mid aAB \mid aC$
$B \rightarrow aCDR_1 \mid aAR_1 \mid aCD \mid aA$
$C \rightarrow a$
$D \rightarrow b$
$R_1 \rightarrow BDR_1 \mid BD$

Replacement of leading variables in S:

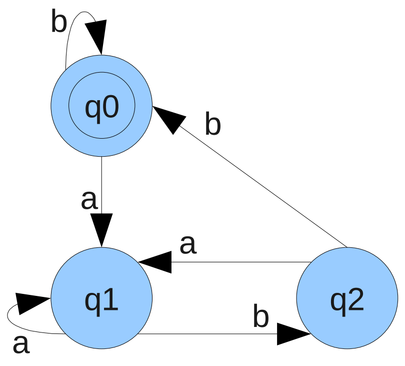
$S \rightarrow aCDR_1BB \mid aAR_1BB \mid aCDBB \mid aABB \mid aCB$
$A \rightarrow aCDR_1B \mid aAR_1B \mid aCDB \mid aAB \mid aC$
$B \rightarrow aCDR_1 \mid aAR_1 \mid aCD \mid aA$
$C \rightarrow a$
$D \rightarrow b$
$R_1 \rightarrow BDR_1 \mid BD$

(Final grammar) Replacement of leading variables in R_1:

$S \rightarrow aCDR_1BB \mid aAR_1BB \mid aCDBB \mid aABB \mid aCB$
$A \rightarrow aCDR_1B \mid aAR_1B \mid aCDB \mid aAB \mid aC$
$B \rightarrow aCDR_1 \mid aAR_1 \mid aCD \mid aA$
$C \rightarrow a$
$D \rightarrow b$
$R_1 \rightarrow aCDR_1DR_1 \mid aAR_1DR_1 \mid aCDDR_1 \mid aADR_1 \mid aCDR_1D \mid aAR_1D \mid aCDD \mid aAD$

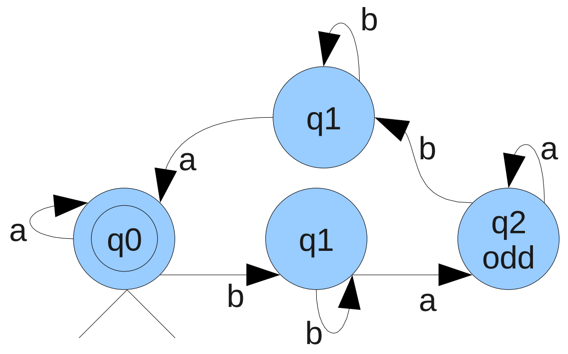
6.

(a)



(b) [q0, babaab] |- [q0, abaab] |- [q1, baab] |- [q2, aab] |- [q1, ab] |- [q1, b] |- [q2, lambda] *rejects*
(c) b*(b*a((ba) U a)*bb)*
(d) b*(b*a((ba) U a)*bb)* U b*a((ba) U a)* U b*a(((ba) U a)*bbb*a)*

7.

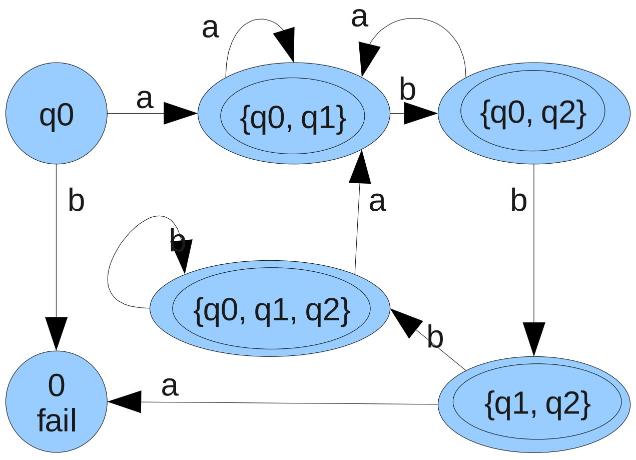


8.

(a)

delta	**a**	**b**
q0	{q0, q1}	0
q1	0	{q0, q2}
q2	0	{q1, q2}

(b) [q0, aabb] |- [q0, abb] |- [q1, bb] |- [q2, b] |- [q2, lambda] accepts
[q0, aabb] |- [q0, abb] |- [q1, bb] |- [q2, b] |- [q1, lambda] accepts
(c) Yes
(d) a+((ba+) U (bb+))*b*
(e)



(f) (a*((ab) U (abb*bb))*) U (a*a((ba*a) U (bb*b))*)
(this was a very algorithmic translation, sorry if it's not so readable)

9.

(a) {{append([], Y, Y)}, {append([X|Y], Z, [X|W]), ~append(Y, Z, W)}}
This translation is an imitation of example 13.1 on p. 145 of the logic book.

(b) append([a,b], [c,d], Z).
We did not go over this in class and there is no example of it in the book, so here's my attempt.
<{~append([a,b], [c,d], Z).}, {append([a|b], [c,d], [a|W]), ~append([b], [c,d], W}>
<{~append([b], [c,d], W}, {append([b|], [c,d], [b|W]), ~append([], [c,d], W}>
<{~append([], [c,d], W)}, {append([], [c,d], [c,d]}>
<□>
Here's the tree version like on p. 75:

```
~append([a,b], [c,d], Z)
          |
~append([b], [c,d], [W])
          |
~append([], [c,d], [W])
          |
          □
```

10.

(a), (b), and (c)

```
grandparentOf(X, Y) :- parentOf(X, Z), parentOf(Z, Y).
greatgrandparentOf(X, Y) :- parentOf(X, P), parentOf(P, Q),
                            parentOf(Q, Y).

sibling(X, Y) :- X\=Y, parentOf(Z, X), parentOf(Z, Y).

cousin(X, Y) :- parentOf(P, X), parentOf(Q, Y), sibling(P, Q).

ancestorOf(X, Y) :- parentOf(X, Y).
ancestorOf(X, Y) :- parentOf(P, Y), ancestorOf(X, P).

% The complete genealogy of the royal house of Troy from the
Iliad
parentOf(zeus, dardanus).
parentOf(dardanus, erichthonius).
parentOf(erichthonius, tros).
parentOf(tros, ganymede).
parentOf(tros, assaracus).
parentOf(tros, ilus).
parentOf(assaracus, capys).
parentOf(capys, anchises).
parentOf(anchises, aeneas).
parentOf(ilus, laomedon).
parentOf(laomedon, priam).
parentOf(priam, hector).
parentOf(laomedon, clytius).
parentOf(laomedon, hicetaon).
parentOf(laomdedon, tithonus).
```

```
     parentOf(clytius, caletor).
     parentOf(hicetaon, melannipus).
     parentOf(lampus, dolops).
     parentOf(laomedon, lampus).

Test results:
?- parentOf(zeus, dardanus).
true.
?- grandparentOf(zeus, erichthonius).
true.
?- grandparentOf(zeus, tros).
false.
?- greatgrandparentOf(zeus, tros).
true.
?- greatgrandparentOf(tros, zeus).
false.
?- sibling(assaracus, ilus).
true .
?- sibling(ilus, assaracus).
true .
?- cousin(capys, laomedon).
true .
?- ancestorOf(capys, X).
X = anchises ;
X = aeneas ;
false.
```

11.

I wish the book had more examples on how to do this.
Even though we have only two constant symbols, does that mean our entire domain has only two elements or instead the domain is N, Q, or Z as in all the chapter's examples?
Otherwise, why would we bother naming the constant symbol 0 in the first place since it's not in either of the formulas?
I have stared at the chapter for at least an hour and can't seem to ascertain an answer.
Also, if we're only given two constant symbols, are we allowed to have a function that produces a third thing?

Domain is Q
$0^A = 2$
$1^A = ½$
Let < be the predicate less than
Let + be multiplication (we will call it f instead to avoid confusion)
Thus:

$f(x, 1) < x$	this will always be true (I.e. $x*1/2 < x$)
$f(x, x) < x$	not always true since $f(0, 0) > 0$ (I.e. $2*2>2$)

12.

By length I assume you mean the actual length of the string, counting characters.

Basis

- A term of length 1 is either a variable or constant symbol. [2.2i, 2.2ii]
- The tree with 1 node as both root and leaf is a variable or constant symbol. [3.1.i.1]

Inductive hypothesis

A term of n applications of the recursive step [2.2iii] is >= to the depth of a formation tree for that term.

Induction

- The n+1st term is a function symbol f with t1...tn as parameters. It adds 1 function symbol, 2 parentheses, and n-1 commas. [2.2iii]. n >= 1 [2.2iii]
- Each of the parameters have length greater than their respective formation trees by the inductive hypothesis. Let the sum of all the lengths of t1..tn be called x and the greatest depth of the parameters be called y. thus, x >= y.
- The length x of the term becomes x+1+2+(n-1) = x+3+(n-1) in the n+1st step.
- The formation tree for that term has f as its root and n children for each of its parameters. 1 level has been added. [3.8i3]. Thus y becomes y+1
- x+3+(n-1) > y+1
- To make it clearer, let n be the minimum number of parameters: n=1. Thus, x+3 > y + 1

13.

Note: the formulas that add boolean operators seem to add spaces as well, but I did not include this, because it still seems easily provable without them.

Basis

- The atomic formula has one predicate symbol, two parentheses, n terms, and n-1 commas. Let the sum of the lengths of each of its terms be x. Thus its length is 3+x+(n-1)
- The depth of each of the formation trees for each of the terms is <= to the length of those terms [#12]
- The depth of the atomic formation auxiliary formation tree is 1. [3.5i] The depth of the full atomic formula formation tree is 1 + the depth of its deepest term, which we call y. [3.5ii]
- x >= y [#12]
- x+3+(n-1) > y + 1

Inductive hypothesis

All formulas created with n applications of the recursive step have length greater than their respective formula formation trees.

Induction

- The n+1st formula has either 3 [2.5ii] or 6 [2.5iii] characters added to it.
- Each subformula has length greater than the depth of their respective formation trees. [IH]
- The depth of the n+1st formula tree is 1 + the depth of its deepest subformula y. [3.8i]
- Let the depth of the deepest subformula be y
- Let the sum of the lengths of the subformulas be x
- x > y [IH]
- x+3 > y+1
- x+6 > y+1

14.

```
           ((Ax)((R(g(x,z,z)) → P(y)) & P(z)))
                              |
                ((R(g(x,z,z)) → P(y)) & P(z))
                         /         \
      (R(g(x,z,z)) → P(y))        P(z)
      /         \                    \
 R(g(x,z,z))    P(y)                   z
    /             \
 g(x,z,z)          y
 /  |  \
x   z   z
```

15.

<u>Basis</u>
The atomic formula begins with R where R is an n-ary predicate symbol [2.4]
<u>Inductive hypothesis</u>
Formulas created with n applications of the recursive steps [2.5ii, 2.5iii] do not being with ~, E, A, (E, (A [note: E and A are the quantifiers]
<u>Induction</u>
The subformulas X and Y do not begin with the above strings by the inductive hypothesis.
For a formula created from n+1 applications of the recursive step, there are seven cases:

1. (X & Y)
2. (X v Y)
3. (X → Y)
4. (X ↔ Y)
5. (~X) *or* (~Y)
6. ((Ev)X) *or* ((Ev)Y)
7. ((Av)X) *or* ((Ev)Y)

As is clearly seen, none of the above cases begin with any of the strings ~, E, A, (E, (A

16.

(a) is a malformed formula because the quantifier E has no variable with it. If it is Ex, then f(z,y) is not substitutable because x is not free. If it is Ey, then it is not substitutable due to 2.8, because y is in f(z,y) and it is not free. If it is Ez, then it is not substitutable because of 2.8

(b) Substitutable. Resulting formula: (((Ex)P(x) & R(x,y)) → P(g(f(z,y),a))
[note: this had an extra a paren]

(c) not substitutable because x has no free occurrences

(d) Substitutable. Resulting formula: ((Ey)(R(a,g(a,b)) & P(y)))