Discrete Structures: Sample Questions, Final Exam

Solutions

1. Construct the labeled tree of the algebraic expression

\[ \frac{(((x - y) * z) - 3) / (19 + (x * x))}{\cdot} \]

Solution:

![Labeled tree diagram]

2. Show the results of a PREORDER search for the following labeled positional binary tree.

![Binary tree diagram]

Solution: AXQXBZBYFAP
3. Consider the following example. Let \( G = \{V, S, \langle \text{integer} \rangle, \rightarrow\} \) for
\[
S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -\},
V = S \cup \{\langle \text{integer} \rangle, \langle \text{unsigned-integer} \rangle, \langle \text{digit} \rangle\},
\]
and let part of the production relation given in BNF notation be given by
\[
\langle \text{unsigned-integer} \rangle ::= \langle \text{digit} \rangle | \langle \text{digit} \rangle \langle \text{unsigned-integer} \rangle
\]
\[
\langle \text{digit} \rangle ::= 0|1|2|3|4|5|6|7|8|9
\]
Design the production relation for \( \langle \text{integer} \rangle \) so that an integer can be either an unsigned integer, or an unsigned integer preceded by a + or − (not both). So +324, 009, −8922 are all valid integers, but +−87, 00−2+, and + are not valid integers. Write the remaining part of the production relation in BNF notation.

**Solution:**
\[
\langle \text{integer} \rangle ::= \langle \text{unsigned-integer} \rangle \mid +\langle \text{unsigned-integer} \rangle \mid −\langle \text{unsigned-integer} \rangle
\]

4. Consider \( G = (V, I, v_0, \rightarrow) \), where \( V = \{v_0, w, a, b, c\} \), \( I = \{a, b, c\} \), and \( \rightarrow \) defined by
\[
v_0 \rightarrow aw, \quad w \rightarrow bw, \quad w \rightarrow bc.
\]
(a) Write the production relation in BNF notation.

**Solution:**
\[
\langle v_0 \rangle ::= a\langle w \rangle
\]
\[
\langle w \rangle ::= bb\langle w \rangle \mid bc
\]
(b) Draw the syntax diagrams of \( v_0 \) and \( w \) separately, and then draw a master syntax diagram for \( v_0 \). (Recall a master diagram is one that involves no nonterminal symbols.)

**Solution:**
(c) Show that the sentence $ab^5c$ is in the language $L(G)$. Draw a derivation tree for this sentence.

**Solution:** $ab^5c \in L(G)$ since

$$v_0 \Rightarrow aw \Rightarrow abbw \Rightarrow abbbbw \Rightarrow abbbbbc = ab^5c.$$  

(The first $\Rightarrow$ follows from $v_0 \Rightarrow aw$, the second two $\Rightarrow$’s follow from $w \Rightarrow bbw$, and the last $\Rightarrow$ follows from $w \Rightarrow bc$.) The derivation tree is below:

(d) Describe the language $L(G)$ in words, and find the regular expression over $I = \{a, b, c\}$ it corresponds to.
Solution: In deriving a sentence in $L(G)$, the first step must always be $v_0 \mapsto bw$, and the final step must involve $w \mapsto bc$ (since this is the only production with only terminal symbols on the right). The only choice is how many times to use the step $w \mapsto bbw$. This step can be used $n$ times, where $n = 0, 1, 2, \ldots$. The sentence produced is then $a(b^2)^nbc = ab^{2n+1}c$. Thus

$$L(G) = \{ab^{2n+1}c \mid n = 0, 1, 2, \ldots\}.$$ 

The regular expression corresponding to $L(G)$ is $a(bb)^*bc$. (The $*$ represents $n$ as above.)

(e) Find a Moore machine $M = (S, I, \mathcal{F}, v_0, T)$ which produces this language $L(G)$. Draw the state diagram and the labeled digraph for $M$.

Solution:

Here is a description of the states of the Moore machine in words.

- $s_0$ is the starting state. No inputs have been received yet.
- At $s_1$, there has been an initial $a$ input possibly followed by an even number of $b$’s.
- At $s_2$, there has been an initial $a$ input, followed by an odd number of $b$’s (the first $b$ comes from $s_1$, and an even number is supplied by the loop between $s_2$ and $s_1$).
- $s_3$ is the only acceptance state. An initial $a$, an odd number of $b$’s and a final $c$ have been input here.
• $s_4$ is the garbage state. Any wrong input at any stage lands here, and any further input makes it remain here. (Because of the structure of the regular expression $a(bb)^*bc$, any wrong move leads to the garbage state.)

The state transition table of this Moore machine is

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$s_1$</td>
<td>$s_4$</td>
<td>$s_4$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$s_4$</td>
<td>$s_2$</td>
<td>$s_4$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$s_4$</td>
<td>$s_1$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$s_4$</td>
<td>$s_4$</td>
<td>$s_4$</td>
</tr>
<tr>
<td>$s_4$</td>
<td>$s_4$</td>
<td>$s_4$</td>
<td>$s_4$</td>
</tr>
</tbody>
</table>

5. Consider the rooted tree $(T, v_0)$ given below.

(a) List all the level-2 vertices of the tree.

**Solution:** $v_4, v_5, v_6, v_7$.

(b) List all the leaves of the tree.

**Solution:** $v_1, v_{12}, v_9, v_5, v_{10}, v_{11}, v_7$.

(c) List all the siblings of $v_3$.

**Solution:** $v_1, v_2$.

(d) Draw the digraph of the subtree $T(v_4)$ with root $v_4$.

**Solution:** This is the subtree with vertices $v_4, v_8, v_9, v_{12}$. 
6. Given the BNF representation for the master syntax diagram given below. You may provide nonterminal symbols as needed (in addition to \( v_0 \)) to use in the BNF productions, and you may use several BNF statements if needed.

**Solution:** The solution below adds two more nonterminal symbols \( w \) and \( y \). These extra symbols will be placed at the junctions in the master diagram where a recursive loop joins another arrow. There are 2 such places in this diagram: In the upper center-left (at the junction right after the \( a \) and \( b \), we place the symbol \( w \)). Also, in the lower right of the master diagram, we place a \( y \) at this recursive junction. (See the diagram below.) To see what happens to the nonterminal symbols \( v_0, w, y \), now we trace along the possible paths in the master diagram, ending whenever we come to a nonterminal symbol (or the end of an arrow of course).
We find the following production relations, in BNF notation. (Recall \( \Lambda \) represents the empty string.)

\[
\begin{align*}
\langle v_0 \rangle & ::= \ ab\langle w \rangle \\
\langle w \rangle & ::= \ \Lambda \mid \langle y \rangle \\
\langle y \rangle & ::= \ d\langle w \rangle \mid dd\langle y \rangle \mid dc\langle y \rangle
\end{align*}
\]

Below are the syntax diagrams of \( v_0, w, y \):

It is also possible to simplify this expression by getting rid of \( w \). In this case, the production relations will be

\[
\begin{align*}
\langle v_0 \rangle & ::= \ ab \mid ab\langle y \rangle \\
\langle y \rangle & ::= \ d \mid d\langle y \rangle \mid dd\langle y \rangle \mid dc\langle y \rangle
\end{align*}
\]
7. Construct the digraph of the positional binary tree for the following doubly linked list. Label each vertex with the corresponding data.

<table>
<thead>
<tr>
<th>INDEX</th>
<th>LEFT</th>
<th>DATA</th>
<th>RIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>M</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>Q</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>T</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>V</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>K</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>G</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td>Y</td>
<td>0</td>
</tr>
</tbody>
</table>

Solution:

8. Construct a Moore machine $M$ that will accept any string ending in $ab$ from input strings of $a, b, c$. In other words, $I = \{a, b, c\}$, and $L(M) = (a \lor b \lor c)^*ab$. Draw the labeled digraph of $M$ and draw its state transition table.
We describe each of the states:

- $s_0$ is the starting state. In addition, it represents any state in which we must “start over.” Since we are looking to accept only strings ending in $ab$, if a $c$ is input or if a $b$ not preceded by an $a$ is input, we must start over.

- $s_1$ is the state at which the previous input is an $a$. Then if a $b$ is input next, we go the acceptance state.

- $s_2$ is the only acceptance state, and the 2 previous inputs must be $ab$.

The state transition table for this Moore machine is

<table>
<thead>
<tr>
<th></th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$s_1$</td>
<td>$s_0$</td>
<td>$s_0$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$s_1$</td>
<td>$s_2$</td>
<td>$s_0$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$s_1$</td>
<td>$s_0$</td>
<td>$s_0$</td>
</tr>
</tbody>
</table>

9. Consider the regular expression $0(0\vee 1)^*1$ over the input set $I = \{0, 1\}$.

(a) Construct a Moore machine with input set $I$ whose language corresponds to this regular expression.

Solution:
We describe the states:

- $s_0$ is the starting state.
- $s_1$ is reached after a 0 is entered (as long as the initial input is 0). Then if a 1 is entered, the state advances to the acceptance state $s_2$.
- $s_2$ is the only acceptance state. It is reached only if the initial input is 0 and the final input is 1.
- $s_3$ is the garbage state. If the initial input is 1 (i.e. not 0), then the machine goes to the garbage state and must stay there.

(b) Construct a Type 3 grammar $G = (V, I, s_0, \rightarrow)$ corresponding to the Moore machine constructed the previous part.

**Solution:** $G = (V, I, s_0, \rightarrow)$, where $V = I \cup S$, $S = \{s_0, s_1, s_2, s_3\}$. The production relation $\rightarrow$ is, in BNF notation

$$
\begin{align*}
\langle s_0 \rangle &::= 0\langle s_1 \rangle | 1\langle s_3 \rangle \\
\langle s_1 \rangle &::= 0\langle s_1 \rangle | 1\langle s_2 \rangle | 1 \\
\langle s_2 \rangle &::= 0\langle s_1 \rangle | 1\langle s_2 \rangle | 1 \\
\langle s_3 \rangle &::= 0\langle s_3 \rangle | 1\langle s_3 \rangle
\end{align*}
$$

(Note that in this grammar, if the garbage state $s_3$ is ever reached, then further production leads to an infinite loop.)

10. (a) Let $g: \mathbb{Z} \rightarrow \{0, 1, 2\}$ be the mod 3 function. Complete the follow-
ing table:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$g(n)$</td>
<td>$g(2n + 0)$</td>
<td>$g(2n + 1)$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

For example, the lower left corner means that if $n$ is an integer with $g(n) = 2$, then we must have $g(2n + 1) = 2$. This can be proved as follows: $g(n) = 2$ if and only if there is an integer $k$ so that $n = 3k + 2$. Then

$$2n + 1 = 2(3k + 2) + 1 = 6k + 5 = (6k + 3) + 2 = 3(2k + 1) + 2,$$

and so the mod 3 function $g$ applied to $2n + 1$ is 2.

**Solution:** The extra entries are in bold above. For the first row, right entry, if $g(n) = 0$, then $g(2n + 1) = 1$. This is because

$$n = 3k \implies 2n + 1 = 2(3k) + 1 = 3(2k) + 1$$

and so $2n + 1$ is 1 mod 3.

The second row, center entry: if $g(n) = 1$, then $g(2n + 0) = 2$, since

$$n = 3k + 1 \implies 2n + 0 = 2(3k + 1) = 3(2k) + 2$$

and so $2n + 0$ is 2 mod 3.

The second row, right entry: if $g(n) = 1$, then $g(2n + 1) = 0$, since

$$n = 3k + 1 \implies 2n + 1 = 2(3k + 1) + 1 = 3(2k + 1)$$

and so $2n + 1$ is 0 mod 3.

The third row, center entry: if $g(n) = 2$, then $g(2n + 0) = 1$, since

$$n = 3k + 2 \implies 2n + 0 = 2(3k + 2) = 3(2k + 1) + 1$$

and so $2n + 0$ is 1 mod 3.

(b) Consider the Moore machine with input set $I = \{0, 1\}$ given below.
We think of elements of $I^*$ as binary integers. Provide a proof by mathematical induction that the Moore machine accepts exactly those binary integers which are divisible by 3. (Hint: we may identify $s_0$ as the set of integers that are 0 mod 3, $s_1$ as the set of integers which are 1 mod 3, and $s_2$ as the set of integers which are 2 mod 3. What does this have to do with the table you completed above?)

**Solution:** Here is the statement to be proved by mathematical induction:

$P(m)$ For a string $w \in I^*$ of length $m$ representing a base-2 integer $n_w$ with $m$ digits, the mod 3 function $g$ determines the state of $f_w(s_0)$ by the following table:

<table>
<thead>
<tr>
<th>$g(n_w)$</th>
<th>$f_w(s_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$s_0$</td>
</tr>
<tr>
<td>1</td>
<td>$s_1$</td>
</tr>
<tr>
<td>2</td>
<td>$s_2$</td>
</tr>
</tbody>
</table>

First note that if $w = a_{m-1} \ldots a_0$ is a binary number with $m$ digits, then

$$n_w = a_{m-1}2^{m-1} + \ldots + a_12^1 + a_02^0.$$  

For the empty string $\Lambda$, then $n_\Lambda$ should represent adding up no terms, which is 0.

Base case of the induction ($m = 0$): $P(0)$. The only string of length 0 is $\Lambda$, and we have that $g(n_\Lambda) = g(0) = 0$, while $f_\Lambda(s_0) = 1_{s_0}(s_0) = s_0$. So the basis step $P(0)$ is checked in this case.

(If this discussion about the empty string is too slick for you, you could also do $m = 1$ as the base case: $P(1)$. The only 2 strings of length 1 are the single bits 0 and 1. Check $g(n_0) = g(0) = 0$, 

$$n_0 = 1 \cdot 2^0 = 1.$$  

Then $f_0(s_0) = 1_{s_0}(s_0) = s_0$. So the base step $P(1)$ is checked in this case.)
\[ f_0(s_0) = s_0; \quad g(n_1) = g(1) = 1, \quad f_1(s_0) = s_1. \] So the case \( P(1) \) works as well.)

Now for the induction step: Assume \( P(k) \) is true for \( k \geq 0 \), and we will use this to prove \( P(k + 1) \).

The key idea is to notice what happens to the binary numbers when another bit is added to the end. If \( w \) is a sentence in \( I^* \) represented the integer \( n_w \) in base 2, then \( w \cdot 0 \) represents the integer \( n_{w,0} = 2n_w = 2n_w + 0 \). (The corresponding fact in base ten is that if write a 0 at the end of a decimal integer \( p \), the new integer is \( 10p \).) Similarly, \( n_{w,1} = 2n_w + 1 \). (If \( d \) is a decimal digit and \( p \) is an integer in base 10, then \( d \) written after \( p \) produces the integer \( 10p + d \).)

The induction step is then checked by the fact that the table on the left computed above

\[
\begin{array}{c|cc}
  & g(n) & g(2n + 0) & g(2n + 1) \\
\hline
0 & 0 & 1 & \\
1 & 2 & 0 & \\
2 & 1 & 2 & \\
\end{array}
\]

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_0 )</td>
<td>( s_0 )</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>( s_2 )</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>( s_1 )</td>
</tr>
</tbody>
</table>

corresponds exactly to the state transition table on the right (which comes from the digraph of the Moore machine). Let \( w \) be a string in \( I^* \) of length \( k \). Thus, when a 0 is added to the end of the string \( w \), the integer \( n_w \) goes to \( 2n_w + 0 \), and \( g(n_w) \) goes to \( g(2n_w) \) according to the table on the left. On the other hand, when a 0 is added to the string \( w \), the state \( f_w(s_0) \) goes to the next state \( f_{w,0}(s_0) = f_0(f_w(s_0)) \) according to the table on the right. The tables match up exactly, and this shows that if the digit 0 is added to \( w \) to form a binary integer of \( k + 1 \) digits, then this part of \( P(k + 1) \) holds.

Similarly, when a 1 is added to the end of \( w \), the corresponding values also match up, and thus we have the following: If \( w \) is a string of \( k \) bits, and \( P(k) \) holds, then either adding a 0 or a 1 to the end gives an integer for which \((*)\) holds as well. Since every string of \( k + 1 \) bits is formed from adding a 0 or 1 to a \( k \)-bit string, this shows that \( P(k + 1) \) holds.

Therefore, by induction, \( P(m) \) holds for all integers \( m \).
The language

\[ L(M) = \{ w \in I^* \mid f_w(s_0) = s_0 \} \]

since \( s_0 \) is the only acceptance value. The induction statement \( P(m) \) shows that for any string of bits \( w \) of any length \( m \), the elements of \( L(M) \) are exactly those \( w \) so that \( g(n_w) = 0 \). Since \( g \) is the mod 3 function, these are exactly those binary integers which are divisible by 3.

11. Consider the finite state machine whose state transition table is

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>s0</td>
<td>s0</td>
<td>s1</td>
<td>s2</td>
</tr>
<tr>
<td>s1</td>
<td>s1</td>
<td>s0</td>
<td>s0</td>
</tr>
<tr>
<td>s2</td>
<td>s2</td>
<td>s0</td>
<td>s1</td>
</tr>
</tbody>
</table>

List all the values of the transition function \( f_{abcc} \).

**Solution:**

<table>
<thead>
<tr>
<th>s</th>
<th>( f_{abcc}(s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>s0</td>
<td>s2</td>
</tr>
<tr>
<td>s1</td>
<td>s1</td>
</tr>
<tr>
<td>s2</td>
<td>s1</td>
</tr>
</tbody>
</table>

12. Evaluate the expression \( 3 \ 4 - 6 + 6\ 12 \div + \), which is in reverse Polish (postfix) notation.

**Solution:** Compute

\[
3 \ 4 - 6 + 6 \ 12 \div + \\
(-1) \ 6 + 6 \ 12 \div + \\
5 \ 6 \ 12 \div + \\
5 \ 0.5 + \\
5.5
\]

13. For the Moore machine \( M \) given below, construct a type 3 grammar \( G = (V, I, s_0, \rightarrow) \) so that \( L(G) = L(M) \). Express the production relation in terms of both \( \rightarrow \) notation and BNF notation.
Solution: $G = (V, I, s_0, \rightarrow)$ with $I = \{0, 1, 2\}$, $V = I \cup S$, $S = \{s_0, s_1, s_2, s_3\}$, and $\rightarrow$ given by

$$
s_0 \rightarrow 0s_0, \quad s_0 \rightarrow 1s_1, \quad s_0 \rightarrow 2s_3, \quad s_0 \rightarrow 2, \\
s_1 \rightarrow 0s_2, \quad s_1 \rightarrow 0, \quad s_1 \rightarrow 1s_0, \quad s_1 \rightarrow 2s_3, \quad s_1 \rightarrow 2, \\
s_2 \rightarrow 0s_2, \quad s_2 \rightarrow 0, \quad s_2 \rightarrow 1s_2, \quad s_2 \rightarrow 1, \quad s_2 \rightarrow 2s_2, \quad s_2 \rightarrow 2, \\
s_3 \rightarrow 0s_1, \quad s_3 \rightarrow 1s_1, \quad s_3 \rightarrow 2s_3, \quad s_3 \rightarrow 2
$$

In BNF notation, $\rightarrow$ is expressed as

$$
\langle s_0 \rangle ::= 0\langle s_0 \rangle | 1\langle s_1 \rangle | 2\langle s_3 \rangle | 2 \\
\langle s_1 \rangle ::= 0\langle s_2 \rangle | 0 | 1\langle s_0 \rangle | 2\langle s_3 \rangle | 2 \\
\langle s_2 \rangle ::= 0\langle s_2 \rangle | 0 | 1\langle s_2 \rangle | 1 | 2\langle s_2 \rangle | 2 \\
\langle s_3 \rangle ::= 0\langle s_1 \rangle | 1\langle s_1 \rangle | 2\langle s_3 \rangle | 2
$$
14. True/False. Circle T or F. No explanation needed. For (a)-(g)), refer to the digraph of the Moore machine $M = (S, I, F, s_0, T)$ represented in the previous problem.

(a) T F $f_{02}(s_1) = s_2$

**Solution:** T.

(b) T F $f_{1112}(s_0) = (f_{112} \circ f_{11})(s_0)$.

**Solution:** T. This can either be checked directly or you can recognize that this follows from the fact that $f_{w \cdot w'} = f_{w'} \circ f_w$ for $w = 11$ and $w' = 112$.

(c) T F $0100 \in L(M)$. (Recall $L(M)$ is the language of $M$.)

**Solution:** T. $f_{0100}(s_0) = s_2 \in T = \{s_2, s_3\}$.

(d) T F $121 \in L(M)$.

**Solution:** F. $f_{121}(s_0) = s_1 \notin T$.

(e) T F There is a Type 3 grammar $G$ with terminal symbols $I = \{0, 1, 2\}$ so that $L(M) = L(G)$.

**Solution:** T. This follows from Theorem 1 in Section 10.5.

(f) T F If $w \in I^*$ contains an odd number of 2’s, then we must have $w \in L(M)$.

**Solution:** F. $20 \notin L(M)$ since $f_{20}(s_0) = s_1 \notin T$.

(g) T F $f_{201}(s_1) = s_2$.

**Solution:** F. $f_{201}(s_1) = s_0$.

(h) T F If $(T, v_0)$ is a rooted tree on a set $A$, then the relation $T$ is irreflexive.

**Solution:** T. Theorem 2 in Section 7.1 (Otherwise, if $vTv$ for some vertex $v$ this creates a nonunique path from $v_0$ to $v$.)

(i) T F If $A = \{1, 2, 3, 4, 5, 6\}$ and $R$ is the relation $\{(1, 2), (1, 4), (3, 5), (3, 6)\}$, then $R$ is a tree on $A$.

**Solution:** F. If $R$ were a tree, then the root would be the unique element with in-degree 0. In this case, both 1 and 3 have in-degree 0, so $R$ cannot be a tree.
In BNF notation, $h v_0 \rightarrow (v_1) a$ is an acceptable production relation for a Type 1 phase structure grammar.

**Solution:** T. For a Type 1 grammar, all that is required is that the length of the string on the right $v_1 a$ be greater than or equal to the length of the string on the left $v_0$.

In BNF notation, $h v_0 \rightarrow (v_1) a$ is an acceptable production relation for a Type 2 phase structure grammar.

**Solution:** T. For a Type 2 grammar, all that is required is that the left hand side $v_0$ must be a single nonterminal symbol.

All the vertices of a complete binary tree have out-degree either 0 or 2.

**Solution:** T. Every vertex of a complete 2-tree which is not a leaf (out-degree 0) has out-degree exactly 2.

Every vertex of a tree has in-degree 1.

**Solution:** F. The root of a tree has in-degree 0, not 1.

In BNF notation, $h v_0 \rightarrow (v_1) a$ is an acceptable production relation for a Type 3 phase structure grammar.

**Solution:** F. For a Type 3 grammar, the right hand side $v_1 a$ must have the nonterminal symbol $v_1$ at the far right of the expression.

Let $G = (V, S, v_0, \rightarrow)$ be a phase structure grammar with $S = \{a, b, c\}$, $V = S \cup \{v_0, v_1\}$, and the production relation determined by

$$v_0 \rightarrow a v_0, \quad v_0 \rightarrow a v_1, \quad v_1 \rightarrow b c v_0, \quad v_1 \rightarrow c.$$ 

Then $v_0 \Rightarrow^\infty a b c$.

**Solution:** F. The only way a string containing $b c$ can be produced from $v_0$ is via $v_1 \rightarrow b c v_0$. But then the productions for $v_0$ all begin with $a$, so if $b c$ is in a sentence in the language, it must be followed by an $a$.

For the grammar $G$ in part (k), the regular expression for the language $L(G)$ is $(a \lor abc)^* c$.

**Solution:** F. $a b c c$ is in the regular set for this regular expression, but as in part (k), $a b c c$ is not in the language $L(G)$. 

(q) T F The string $xxy$ is in the regular set determined by the regular expression $(xx \lor (xy)^* \lor xy^*)^*$.

**Solution:** T. $x$ and $xy$ are both in the regular set for the expression $xy^*$. So both $x$ and $xy$ are in the regular set for the expression $xx \lor (xy)^* \lor xy^*$. The outermost $^*$ at the end of the expression $(xx \lor (xy)^* \lor xy^*)^*$ means that $xx \lor (xy)^* \lor xy^*$ can be repeated twice, the first time producing $x$ and the second time producing $xy$. Thus $xxy$ is in the regular set.

(r) T F $0^*(1 \lor \Lambda)$ is a regular expression over the set $I = \{0, 1\}$.

**Solution:** T. Recall $\Lambda$ the empty string can be used in regular expressions for any set.

(s) T F $37 \times 4 - 9 \times 65 \times 2 +$ is a valid mathematical expression in reverse Polish (postfix) notation.

**Solution:** F. There are not enough binary expressions. Compute

\[
\begin{align*}
37 & \times 4 - 9 \times 65 \times 2 + \\
21 & - 9 \times 65 \times 2 + \\
17 & 9 \times 65 \times 2 + \\
153 & 65 \times 2 + \\
153 & 30 2 + \\
153 & 32
\end{align*}
\]

So we are left with 2 numbers and no binary expression with which to combine them.