Gradiance Online Accelerated Learning

Zayd

- Home Page
- Assignments Due
- Progress Report
- Handouts
- Tutorials
- Homeworks
- Lab Projects
- Log Out

Help

Submission number: 85624
Submission certificate: ED374057
Submission time: 2014-05-14 04:29:52 PST (GMT - 8:00)

Number of questions: 25
Positive points per question: 4.0
Negative points per question: 0.0
Your score: 28

1. The operation Perm(w), applied to a string w, is all strings that can be constructed by permuting the symbols of w in any order. For example, if w = 101, then Perm(w) is all strings with two 1's and one 0, i.e., Perm(w) = {101, 110, 011}. If L is a regular language, then Perm(L) is the union of Perm(w) taken over all w in L. For example, if L is the language L(0*1*), then Perm(L) is all strings of 0's and 1's, i.e., L((0+1)*).

If L is regular, Perm(L) is sometimes regular, sometimes context-free but not regular, and sometimes not even context-free. Consider each of the following regular expressions R below, and decide whether Perm(L(R)) is regular, context-free, or neither:

1. (01)*
2. 0*+1*
3. (012)*
4. (01+2)*

a) Perm(L((01)*)) is regular.

b) Perm(L((01)*)) is context-free but not regular.

c) Perm(L((01)*)) is not context-free.

d) Perm(L((01+2)*)) is not context-free.

You did not answer this question.

2. We wish to perform the reduction of acceptance by a Turing machine to MPCP, as described in Section 9.4.3 (p. 407). We assume the TM *M* satisfies Theorem 8.12 (p. 346): it never moves left from its initial position and never writes a blank. We know the following:

1. The start state of *M* is *q*.
2. *r* is the accepting state of *M*.
3. The tape symbols of *M* are 0, 1, and B (blank).
4. One of the moves of *M* is δ(q,0) = (p,1,L).

Which of the following is DEFINITELY NOT one of the pairs in the MPCP instance that we construct for the TM *M* and the input 001?

a) (#, #q001#)

b) (q0, 1p)

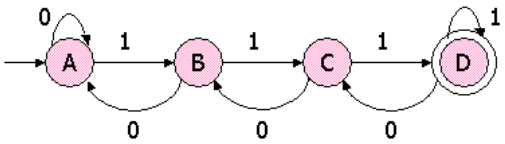
c) (0r, r)

d) (0q0, p01)

Answer submitted: **b)**

You have answered the question correctly.

3. Examine the following DFA:



This DFA accepts a certain language L. In this problem we shall consider certain other languages that are defined by their tails, that is, languages of the form (0+1)*w, for some particular string *w* of 0's and 1's. Call this language L(w). Depending on *w*, L(w) may be contained in L, disjoint from L, or neither contained nor disjoint from L (i.e., some strings of the form *xw* are in L and others are not).

Your problem is to find a way to classify *w* into one of these three cases. Then, use your knowledge to classify the following languages:

1. L(1111001), i.e., the language of regular expression (0+1)*1111001.
2. L(11011), i.e., the language of regular expression (0+1)*11011.
3. L(110101), i.e., the language of regular expression (0+1)*110101.
4. L(00011101), i.e., the language of regular expression (0+1)*00011101.

a) L(00011101) is disjoint from L.

b) L(11011) is contained in L.

c) L(110101) is contained in L.

d) L(110101) is disjoint from L.

You did not answer this question.

4. If we convert the context-free grammar G:

```
S → AS | A
A → 0A | 1B | 1
B → 0B | 0
```

to a pushdown automaton that accepts L(G) by empty stack, using the construction of Section 6.3.1, which of the following would be a rule of the PDA?

a) $\delta(q,0,B) = \{(q,B), (q,\varepsilon)\}$

b) $\delta(q,\varepsilon,B) = \{ (q,0)\}$

c) $\delta(q,\varepsilon,S) = \{(q,AS)\}$

d) $\delta(q,\varepsilon,A) = \{(q,0A), (q,1B), (q,1)\}$

Answer submitted: **d)**

You have answered the question correctly.

5. h is a homomorphism from the alphabet {a,b,c} to {0,1}. If h(a) = 01, h(b) = 0, and h(c) = 10, which of the following strings is in $h^{-1}(010010)$?

a) *abba*

b) *abab*

c) *bcba*

d) *cbba*

You did not answer this question.

6. Convert the grammar:

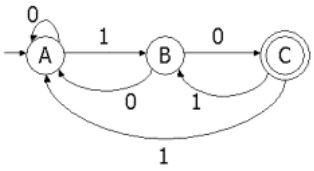
```
S → A | B | 2
A → C0 | D
B → C1 | E
C → D | E | 3
D → E0 | S
E → D1 | S
```

to an equivalent grammar with no unit productions, using the construction of Section 7.1.4 (p. 268). Then, choose one of the productions of the new grammar from the list below.

a) $S \rightarrow D$
b) $S \rightarrow 3$
c) $A \rightarrow 3$
d) $A \rightarrow C1$

You did not answer this question.

7. When we convert an automaton to a regular expression, we need to build expressions for the labels along paths from one state to another state that do not go through certain other states. Below is a nondeterministic finite automaton with three states. For each of the six orders of the three states, find regular expressions that give the set of labels along all paths from the first state to the second state that never go through the third state.



Then identify one of these expressions from the list of choices below.

a) 0*1 represents the paths from A to B that do not go through C.
b) 0(0+10)* represents the paths from B to A that do not go through C.
c) 1 represents the paths from C to B that do not go through A.
d) 0(10)* represents the paths from B to A that do not go through C.

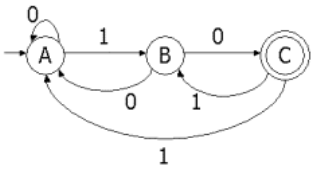
You did not answer this question.

8. The intersection of two CFL's need not be a CFL. Identify in the list below a pair of CFL's such that their intersection is not a CFL.

a) $L_1=\{aba^ib^nc^iba \mid n>0, i>0\}$
$L_2=\{aba^nb^ic^iba \mid n>0, i>0, j>0\}$
b) $L_1=\{a^nb^nc^ic \mid n>0, i>0\}$
$L_2=\{a^ja^nb^ic^i \mid n>0, i>0, j>0\}$
c) $L_1=\{aba^nb^nc^i \mid n>0, i>0\}$
$L_2=\{aba^nb^jc^i \mid n>0, i>0, j>0\}$
d) $L_1=\{aba^nb^nc^nba \mid n>0, i>0\}$
$L_2=\{aba^nb^ic^iba \mid n>0, i>0, j>0\}$

You did not answer this question.

9. Convert the following nondeterministic finite automaton:



to a DFA, including the dead state, if necessary. Which of the following sets of NFA states is **not** a state of the DFA that is accessible from the start state of the DFA?

a) {A,B}
b) {B}
c) {B,C}
d) {A,C}

You did not answer this question.

10. Consider the languge L={a}. Which grammar defines L?

a) G_1:S → d|a, A → c|b|ε

b) G_1:S → AC|a|ab, A → c|ε

c) G_1:S → AC|a, A → c|b|ε

d) G_1:S → AB|BC, A → b, C → a

You did not answer this question.

11. The Independent-Set problem is shown to be NP-complete in the text (Theorem 10.18, Section 10.4.2, p. 460) by a reduction from 3SAT. In the text, the Node-Cover problem is proved to be NP-complete (Theorem 10.20, Section 10.4.3, p. 463) by a reduction from the Independent-Set problem.

We can prove the Node-Cover problem to be NP-complete by a reduction from 3SAT directly. For variable *a*, let *a*' denote NOT(*a*). Given a 3-CNF expression,

$E = (x_1 + x_2 + x_3')(x_1 + x_2' + x_4)(x_3' + x_4' + x_5)(x_1 + x_3 + x_5')$

construct a graph G using the transformation given by Theorem 10.18. Let us denote a node in G by the corresponding (clause, literal) pair (See Figure 10.8 on p. 461).

Which of the following is a node cover for G and of the smallest possible size for a node cover for G?

a) {(1,1), (1,2), (2,2), (2,3), (3,2), (3,3), (4,3), (4,2)}

b) {(1,1), (1,2), (1,3), (2,2), (2,3), (3,2), (3,3), (4,1), (4,2)}

c) {(1,1), (1,3), (2,1), (2,3), (3,1), (3,2), (4,1), (4,2)}

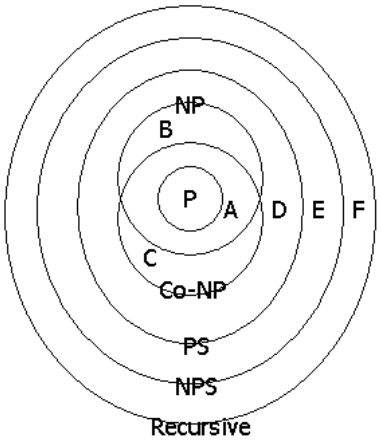
d) { (1,2), (1,3), (2,2), (2,3), (3,2), (3,3), (4,1), (4,2),(4,3)}

You did not answer this question.

12. Consider the following problems:

1. SP (Shortest Paths): given a weighted, undirected graph with nonnegative integer edge weights, given two nodes in that graph, and given an integer limit *k*, determine whether the length of the shortest path between the nodes is *k* or less.
2. WHP (Weighted Hamilton Paths): given a weighted, undirected graph with nonnegative integer edge weights, and given an integer limit *k*, determine whether the length of the shortest Hamilton path in the graph is *k* or less.
3. TAUT (Tautologies): given a propositional boolean formula, determine whether it is true for all possible truth assignments to its variables.
4. QBF (Quantified Boolean Formulas): given a boolean formula with quantifiers for-all and there-exists, such that there are no free variables, determine whether the formula is true.

In the diagram below are seven regions, P and A through F.



Place each of the four problems in its correct region, on the assumption that NP is equal to neither P nor co-NP nor PS.

a) Problem TAUT is in region P.

b) Problem WHP is in region D.

c) Problem TAUT is in region C.

d) Problem SP is in region B.

Answer submitted: **c)**

You have answered the question correctly.

13. Design the minimum-state DFA that accepts all and only the strings of 0's and 1's that end in 010. To verify that you have designed the correct automaton, we will ask you to identify the true statement in a list of choices. These choices will involve:

1. The number of *loops* (transitions from a state to itself).
2. The number of transitions into a state (including loops) on input 1.
3. The number of transitions into a state (including loops) on input 0.

Count the number of transitions into each of your states ("in-transitions") on input 1 and also on input 0. Count the number of loops on input 1 and on input 0. Then, find the true statement in the following list.

a) There are three states that have one in-transition on input 0.

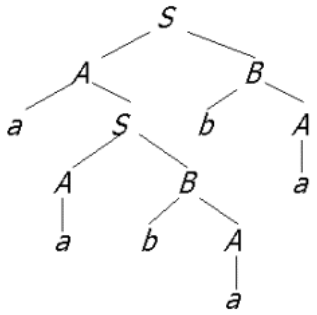
b) There are two states that have no in-transitions on input 0.

c) There is one state that has two in-transitions on input 1.

d) There is one state that has one in-transition on input 1.

You did not answer this question.

14. The parse tree below represents a leftmost derivation according to the grammar `S → AB, A → aS|a, B → bA`.

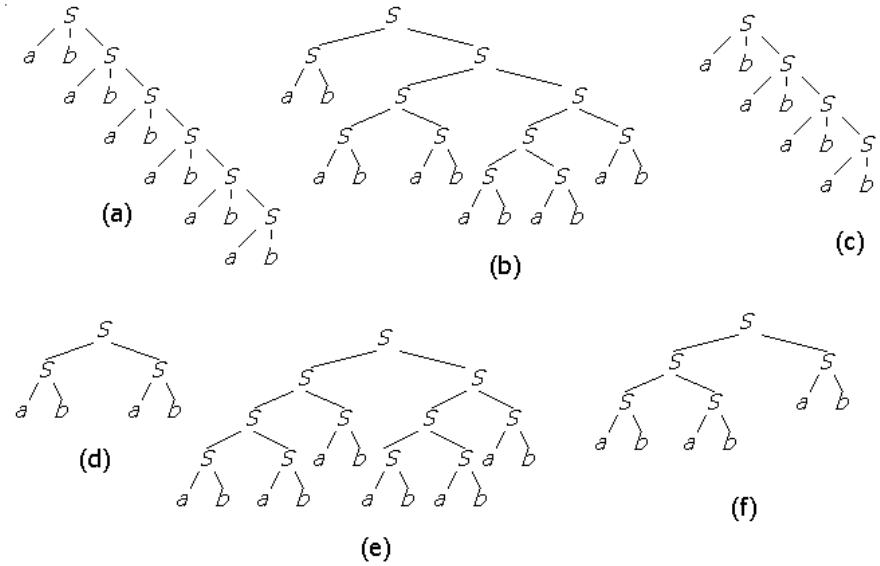


Which of the following is a left-sentential form in this derivation?

a) aaABBB
b) aabaB
c) aSba
d) aAbaba

You did not answer this question.

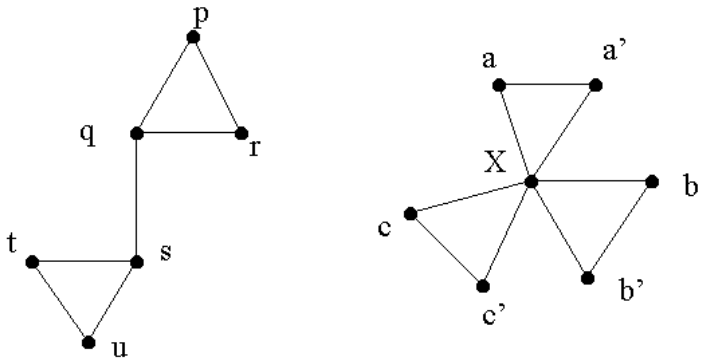
15. Which of the parse trees below yield the same word?



a) a and d
b) a and c
c) c and d
d) a and b

You did not answer this question.

16. The graph k-coloring problem is defined as follows: Given a graph G and an integer *k*, is G k-colorable?, i.e. can we assign one of *k* colors to each node of G such that no edge has both of its ends colored by the same color. The graph 3-coloring problem is NP-complete and this fact can be proved by a reduction from 3SAT.



As a part of the reduction proof, we assume that there are three colors GREEN, RED and BLUE. For variable *a*, let *a*' denote NOT(a). We associate with each variable *a*, a "gadget" consisting of 3 nodes labeled *a*, *a*' and X. This gadget is shown at the right in the diagram above. X is common for all variables and is labeled blue. If *a* is TRUE (respectively FALSE) in a particular assignment, node *a* is colored green (respectively red) and node *a*' is colored red (respectively green).

With each clause we associate a copy of the gadget at the left in the diagram above, defined by the graph H = (V,E) where V = {*p*,*q*,*r*,*s*,*t*,*u*} and E = {(*p*,*q*), (*q*,*r*), (*r*,*p*), (*s*,*t*), (*t*,*u*), (*u*,*s*), (*q*,*s*)}. Nodes *t*, *u* and *r* from this copy of the gadget are connected to the nodes constituting the literals for the clause, in left to right order.

Consider a clause in the 3-CNF expression: *a* + *b*' + *c*. We must color the above gadget using the colors red, blue and green in such a way that no two adjacent nodes get the same color. In each choice below, you are given an assignment for the variables *a*, *b*, and *c* and a possible assignment of colors to some nodes in the gadget above. Indicate the choice of colors that is valid for the given truth assignment:

a) (a=TRUE, b=FALSE, c=TRUE, t=red, u=blue, p=red)

b) (a=FALSE, b=TRUE, c=FALSE, t=red, r=green, p=blue)

c) (a=FALSE, b=TRUE, c=FALSE, s=blue, q=red, p=green)

d) (a=FALSE, b=TRUE, c=FALSE, t=green, u=blue, p=red)

Answer submitted: **a)**

Your answer is incorrect.

If t is red and u is blue, since s is adjacent to t and u, s is green. Since s is green and p is red, q being adjacent to s and p, is blue. Since q is blue and p is red, r being adjacent to p and q, is green. Since c is TRUE, it is green. But c is adjacent to r which is also green, which is impossible. The coloring problem is defined in Exercise 10.4.2 (p. 474).

17. In this question you are asked to consider the truth or falsehood of six equivalences for regular expressions. If the equivalence is true, you must also identify the law from which it follows. In each case the statement R = S is conventional shorthand for "L(R) = L(S)." The six proposed equivalences are:

1. 0*1* = 1*0*
2. 01φ = φ
3. ε01 = 01
4. (0* + 1*)0 = 0*0 + 1*0
5. (0*1)0* = 0*(10*)
6. 01+01 = 01

Identify the correct statement from the list below.

Note: we use φ for the empty set, because the correct symbol is not recognized by Internet Explorer.

a) ε01 = 01 follows from the annihilator law for concatenation.

b) ε01 = 01 follows from the commutative law for concatenation.

c) (0* + 1*)0 = 0*0 + 1*0 follows from the associative law for concatenation.

d) (0* + 1*)0 = 0*0 + 1*0 follows from the distributive law of concatenation over union.

Answer submitted: **d)**

You have answered the question correctly.

18. A nondeterministic Turing machine M with start state q_0 and accepting state q_f has the following transition function:

$\delta(q,a)$	0	1	B
q_0	$\{(q_1,0,R)\}$	$\{(q_1,0,R)\}$	$\{(q_1,0,R)\}$
q_1	$\{(q_1,1,R), (q_2,0,L)\}$	$\{(q_1,1,R), (q_2,1,L)\}$	$\{(q_1,1,R), (q_2,B,L)\}$
q_2	$\{(q_f,0,R)\}$	$\{(q_2,1,L)\}$	$\{\}$
q_f	$\{\}$	$\{\}$	$\{\}$

Simulate all sequences of 5 moves, starting from initial ID $q_0$1010. Find, in the list below, one of the ID's reachable from the initial ID in EXACTLY 5 moves.

a) $0q_2111$

b) $0111q_1$

c) $0q_f110$

d) $011111q_1$

You did not answer this question.

19. The NOT-ALL-EQUAL 3SAT problem is defined as follows: Given a 3-CNF formula F, is there a truth assignment for the variables such that each clause has at least one true literal and at least one false literal? The NOT-ALL-EQUAL 3SAT problem is NP-complete.

This question is about trying to reduce the NOT-ALL-EQUAL 3SAT problem to the MAX-CUT problem defined below to show the latter to be NP-complete.

A cut in an undirected graph G=(V,E) is a partitioning of the set of nodes V into two disjoint subsets V1 and V2. The size of a cut is the number of edges $e = (u,v)$ where u is in V1 and v is in V2. The MAX-CUT problem is defined as follows: Given an undirected graph G=(V,E) and a positive integer k, does G have a cut of size k or more?

Given a 3CNF expression E, we create the graph G = (V,E) using the transformation given by Theorem 10.18 in Section 10.4.2 on p. 460 of the text. Then given an assignment A, create a cut C in G by partitioning the set of nodes V as follows: the nodes corresponding to the uncomplemented literals are in set V1 and those corresponding to the complemented variables are in set V2.

For variable a, let a' denote NOT(a). Let

$E = (a + b + c)(a + b' + c)(a' + b' + d)(c' + d' + e)$

be an instance of NOT-ALL-EQUAL 3SAT. Suppose a cut separates the true nodes from false nodes according to some truth assignment applied to E. How many edges between nodes corresponding to the literals in the same clause are cut? How many other edges are cut? Find out how the cut-size can be computed for an arbitrary instance of NOT-All-EQUAL 3SAT. Then for the instance E, determine in which of the cases below, the cut-size C corresponds to the satisfiable assignment given.

a) a = T, b = T, c = F, d = T, e = T, C = 15

b) a = F, b = T, c = F, d = F, e = T, C = 13

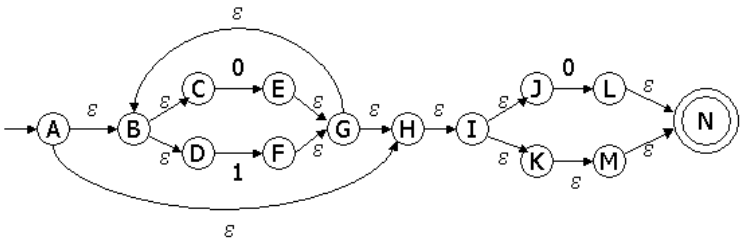
c) a = T, b = F, c = F, d = T, e = F, C = 7

d) a = T, b = F, c = T, d = F, e = T, C = 15

Answer submitted: **a)**

You have answered the question correctly.

20. Here is an epsilon-NFA:



Suppose we construct an equivalent DFA by the construction of Section 2.5.5 (p. 77). That is, start with the epsilon-closure of the start state A. For each set of states *S* we construct (which becomes one state of the DFA), look at the transitions from this set of states on input symbol 0. See where those transitions lead, and take the union of the epsilon-closures of all the states reached on 0. This set of states becomes a state of the DFA. Do the same for the transitions out of *S* on input 1. When we have found all the sets of epsilon-NFA states that are constructed in this way, we have the DFA and its transitions.

Carry out this construction of a DFA, and identify one of the states of this DFA (as a subset of the epsilon-NFA's states) from the list below.

a) BCD

b) ABCD

c) IJKMN

d) ABCDHIJKMN

You did not answer this question.

21. Consider the grammar G1: S → ε, S → aS, S → aSbS and the language L that contains exactly those strings of a's and b's such that every prefix has at least as many a's as b's. We want to prove the claim: G1 generates all strings in L.

We take the following inductive hypothesis to prove the claim:

For n < k, G1 generates every string of length n in L.

To prove the inductive step we argue as follows:

"For each string w in L either _____(a1) or _____(a2) holds. In both cases we use the inductive hypothesis and one of the rules to show that string w can be generated by the grammar. In the first case we use rule S → aS and in the second case we use rule S → aSbS."

Which phrases can replace the _____ so that this argument is correct?

a) a1: each prefix has equal number of a's and b's.
a2: there is a b in string w such that the part of the string until the b belongs in L by inductive hypothesis and the part after this b belongs in L by inductive hypothesis.

b) a1: there is a b in string w such that for the part of the string until the b (b also included) each prefix has as many a's as b's and for the part after b each prefix has as many a's as b's.
a2: each prefix has more a's than b's.

c) a1: each prefix has equal number of b's and a's.
a2: w can be written as w=aw'bw" where for both w' and w" it holds that each prefix has as many a's as b's.

d) a1: w can be written as w=aw' where for each prefix of w' has as many a's as b's.
a2: there is a b in string w such that the part of the string until the b (b also included) has all prefixes with as many a's as b's and the part after this b has all prefixes with as many a's as b's.

Answer submitted: **d)**

You have answered the question correctly.

22. Let us denote a problem X as NP-Easy if it is polynomial-time reducible to some problem Y that is in NP. Let us denote as NP-Equivalent, the class of problems that are both NP-Easy and NP-Hard. Let A,B,C,D

and E be problems such that A is NP-Hard, B is NP-Complete, C is NP-Equivalent, D is NP-Easy and E is in NP. Which of the following statements is TRUE?

a) If C is in P then so is A.

b) If C is in P then P=NP.

c) If P=NP then A is in P.

d) If E is in P then so is A.

Answer submitted: **b)**

You have answered the question correctly.

23. Here is a grammar, whose variables and terminals are NOT named using the usual convention. Any of *R* through *Z* could be either a variable or terminal; it is your job to figure out which is which, and which could be the start symbol.

```
R → ST | UV
T → UV | W
V → XY | Z
X → YZ | T
```

We do have an important clue: There are no useless productions in this grammar; that is, each production is used in some derivation of some terminal string from the start symbol. Your job is to figure out which letters definitely represent variables, which definitely represent terminals, which could represent either a terminal or a nonterminal, and which could be the start symbol. Remember that the usual convention, which might imply that all these letters stand for either terminals or variables, does not apply here.

a) *T* must be the start symbol.

b) *X* must be the start symbol.

c) *Y* must be a terminal.

d) *U* must be a variable.

You did not answer this question.

24. Let L be the language of all strings of a's and b's such that no prefix (proper or not) has more b's than a's. Let G be the grammar with productions

```
S → aS | aSbS | ε
```

To prove that L = L(G), we need to show two things:

1. If S =>* w, then w is in L.
2. If w is in L, then S =>* w.

We shall consider only the proof of (2) here. The proof is an induction on n, the length of w. Here is an outline of the proof, with reasons omitted. You need to supply the reasons.

Basis:

1) If n=0, then w is ε because _____.

2) S =>* w because _____.

Induction:

3) Either (a) w can be written as w=aw' where for w' each prefix has as many a's as b's or (b) w can be written as w=aw'bw" where for both w' and w" hold that each prefix has as many a's as b's because _____.

4a) In case (a), w' is in the language because _____.

5a) In case (a), S =>* w' because _____.

6a) In case (a), S =>* w because _____.

4b) In case (b), both w' and w" are in the language because _____.

5b) In case (b), S =>* w' because _____.

6b) In case (b), S =>* w" because _____.

7b)

In case (b), S =>* w because _____.

For which of the steps above is the appropriate reason "by the inductive hypothesis"?

a) 7b

b) 2

c) 5a

d) 3

You did not answer this question.

25. Programming languages are often described using an extended form of context-free grammar, where curly brackets are used to denote a construct that can repeat 0, 1, 2, or any number of times. For example, A → B{C}D says that an *A* can be replaced by a *B* and a *D*, with any number of *C*'s (including 0) between them. This notation does not allow us to describe anything but context-free languages, since an extended production can always be replaced by several conventional productions.

Suppose a grammar has the extended production:

$A \rightarrow 0\{B\}1$

Convert this extended production to conventional productions. Identify, from the list below, the conventional productions that are equivalent to the extended production above.

a) $A \rightarrow 0BA_1 1$
$A_1 \rightarrow BA_1 \mid \varepsilon$

b) $A \rightarrow 0A_1 1$
$A_1 \rightarrow A_1 B \mid \varepsilon$

c) $A \rightarrow 0A_1 B1$
$A_1 \rightarrow A_1 B \mid \varepsilon$

d) $A \rightarrow 0A_1 1$
$A_1 \rightarrow A_1 B \mid B$

You did not answer this question.