Gradiance Online Accelerated Learning

Zayd

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

Help

Submission number: 86185
Submission certificate: DJ060276
Submission time: 2014-05-15 01:25:31 PST (GMT - 8:00)

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

1. Here are the transitions of a deterministic pushdown automaton. The start state is q_0, and *f* is the accepting state.

State-Symbol	a	b	ε
q_0-Z_0	(q_1,AAZ_0)	(q_2,BZ_0)	(f,ϵ)
q_1-A	(q_1,AAA)	(q_1,ϵ)	-
q_1-Z_0	-	-	(q_0,Z_0)
q_2-B	(q_3,ϵ)	(q_2,BB)	-
q_2-Z_0	-	-	(q_0,Z_0)
q_3-B	-	-	(q_2,ϵ)
q_3-Z_0	-	-	(q_1,AZ_0)

Describe informally what this PDA does. Then, identify below, the one input string that takes the PDA into state q_3 (with any stack).

a) bababba
b) abbba
c) baabba
d) ababba

Answer submitted: **b)**

You have answered the question correctly.

2. Here is the transition function of a simple, deterministic automaton with start state A and accepting state B:

	0	1
A	A	B
B	B	A

We want to show that this automaton accepts exactly those strings with an odd number of 1's, or more formally:

$$\delta(A,w) = B \text{ if and only if } w \text{ has an odd number of 1's.}$$

Here, δ is the extended transition function of the automaton; that is, $\delta(A,w)$ is the state that the automaton is in after processing input string w. The proof of the statement above is an induction on the length of w. Below, we give the proof with reasons missing. You must give a reason for each step, and then demonstrate your understanding of the proof by classifying your reasons into the following three categories:

A)
Use of the inductive hypothesis.

B)
Reasoning about properties of deterministic finite automata, e.g., that if string $s = yz$, then $\delta(q,s) = \delta(\delta(q,y),z)$.

C)
Reasoning about properties of binary strings (strings of 0's and 1's), e.g., that every string is longer than any of its proper substrings.

Basis ($|w| = 0$):

(1)
$w = \varepsilon$ because ________

(2)
$\delta(A,\varepsilon) = A$ because ________

(3)
ε has an even number of 0's because ________

Induction ($|w| = n > 0$)

(4)
There are two cases: (a) when $w = x1$ and (b) when $w = x0$ because ________
Case (a):

(5)
In case (a), w has an odd number of 1's if and only if x has an even number of 1's because ________

(6)
In case (a), $\delta(A,x) = A$ if and only if w has an odd number of 1's because ________

(7)
In case (a), $\delta(A,w) = B$ if and only if w has an odd number of 1's because ________
Case (b):

(8)
In case (b), w has an odd number of 1's if and only if x has an odd number of 1's because ________

(9)
In case (b), $\delta(A,x) = B$ if and only if w has an odd number of 1's because ________

(10)
In case (b), $\delta(A,w) = B$ if and only if w has an odd number of 1's because ________

a) (9) for reason C.
b) (9) for reason A.
c) (2) for reason A.
d) (5) for reason B.

You did not answer this question.

3. Programming languages are often described using an extended form of context-free grammar, where square brackets are used to denote an optional construct. For example, $A \rightarrow B[C]D$ says that an *A* can be replaced by a *B* and a *D*, with an optional *C* 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 productions:

```
A → BC[DEFG]HI | BCDE[FGH]I
```

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

a) $A \rightarrow BCA_1I$
$A_1 \rightarrow DEFG \mid FGH$

b) $A \rightarrow BCDEFGHI \mid BCHI \mid BCDEI$

c) $A \rightarrow BCA_1I$
$A_1 \rightarrow DEFG \mid FGH \mid \varepsilon$

d) $A \rightarrow BCA_1HI \mid BCDEA_2I$
$A_1 \rightarrow DEFG$
$A_2 \rightarrow FGH$

You did not answer this question.

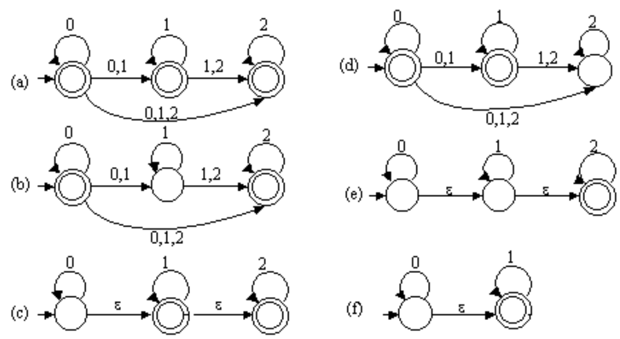
4. G_1 is a context-free grammar with start symbol S_1, and no other nonterminals whose name begins with "S." Similarly, G_2 is a context-free grammar with start symbol S_2, and no other nonterminals whose name begins with "S." S_1 and S_2 appear on the right side of no productions. Also, no nonterminal appears in both G_1 and G_2.

We wish to combine the symbols and productions of G_1 and G_2 to form a new grammar G, whose language is the union of the languages of G_1 and G_2. The start symbol of G will be S. All productions and symbols of G_1 and G_2 will be symbols and productions of G. Which of the following sets of productions, added to those of G, is guaranteed to make L(G) be $L(G_1)$ [union] $L(G_2)$?

a) $S \rightarrow S_1S_3,\ S_3 \rightarrow S_2$
b) $S \rightarrow S_1S_2 \mid S_2S_1$
c) $S \rightarrow S_3S_2,\ S_3 \rightarrow S_1 \mid \varepsilon$
d) $S \rightarrow S_1 \mid S_3S_2,\ S_3 \rightarrow \varepsilon$

You did not answer this question.

5. Identify which automata define the same language and provide the correct counterexample if they don't. Choose the correct statement from the list below.



a) (a) and (f) define the same language.
b) (b) and (f) define the same language.
c) (b) and (d) do not define the same language and the following counterexample shows it. String 0012 is accepted by one and not by the other.
d) (c) and (b) do not define the same language and the following counterexample shows it. String 0012 is accepted by one and not by the other.

You did not answer this question.

6. Which of the following grammars derives a subset L_s of the language: L= {x | (i) x contains a and c in proportion 4:3, (ii) x does not begin with c and (iii) there are no two consecutive c's} such that L_s is missing at most a finite number of strings from L.

a) $S \rightarrow \varepsilon$, $S \rightarrow$ SaScSaScSaScSaS, $S \rightarrow A$, $A \rightarrow$ acaca
b) $S \rightarrow$ acacaca, $S \rightarrow$ SaScSaScSaScSaS, $S \rightarrow$ SaSaSaScSaScSa
c) $S \rightarrow$ acacaca, $S \rightarrow$ SaSaSaScSaScSaS
d) $S \rightarrow$ acacaca , $S \rightarrow$ SaScSaScSaScSaS, $S \rightarrow$ SaSaSaScSaScSaS

You did not answer this question.

7. Let *h* be the homomorphism defined by h(a) = 01, h(b) = 10, h(c) = 0, and h(d) = 1. If we take any string *w* in (0+1)*, $h^{-1}(w)$ contains some number of strings, N(w). For example, $h^{-1}(1100)$ = {ddcc, dbc}, i.e., N(1100) = 2. We can calculate the number of strings in $h^{-1}(w)$ by a recursion on the length of *w*. For example, if w = 00x for some string x, then N(w) = N(0x), since the first 0 in w can only be produced from *c*, not from *a*.

Complete the reasoning necessary to compute N(w) for any string *w* in (0+1)*. Then, choose the correct value of N(01100110).

a) 8

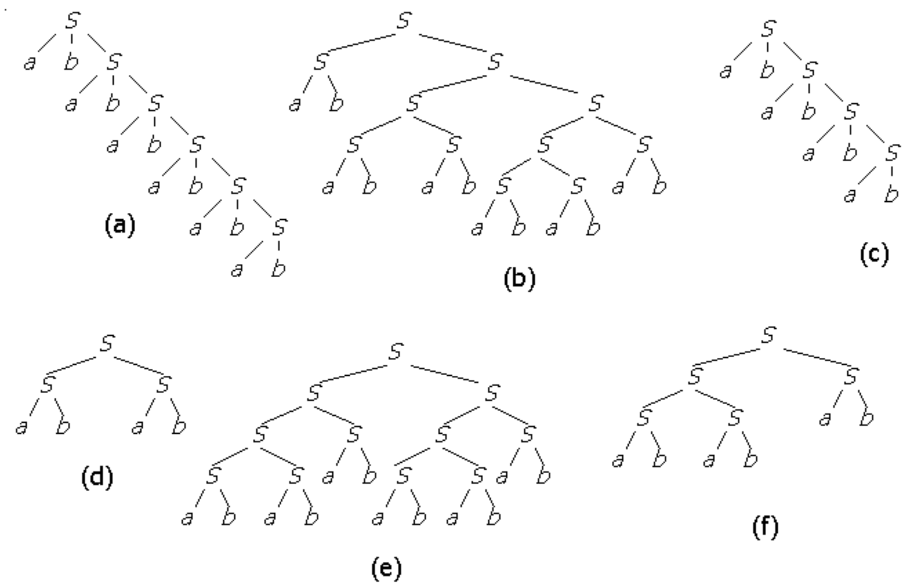
b) 34

c) 16

d) 128

You did not answer this question.

8. Consider the grammar G: S → SS, S → ab. Which of the following strings is a word of L(G) AND is the yield of one of the parse trees for grammar G in the figure below?



a) abababababababab

b) abab

c) abababab

d) ababSabab

You did not answer this question.

9. The Turing machine M has:

- States q and p; q is the start state.
- Tape symbols 0, 1, and B; 0 and 1 are input symbols, and B is the blank.
- The following next-move function:

State	Tape	Move
	Symbol	
q	0	(q,0,R)
q	1	(p,0,R)
q	B	(q,B,R)
p	0	(q,0,L)
p	1	none (halt)
p	B	(q,0,L)

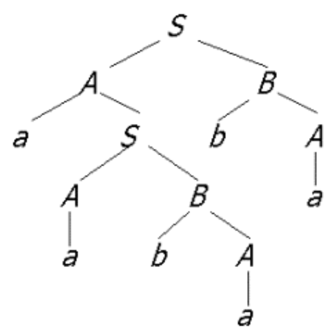
Simulate M on the input 1010110, and identify one of the ID's (instantaneous descriptions) of M from the list below.

a) 0000q010
b) 00000p10
c) 001q0110
d) 101q0110

Answer submitted: **b)**

You have answered the question correctly.

10. The folowing is a parse tree in some unknown grammar G:



Which of the following productions is **definitely not** a production of G?

a) $S \rightarrow aC$
b) $B \rightarrow CD$
c) None of the other choices.
d) $S \rightarrow AB$

You did not answer this question.

11. 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 (1) here. The proof is an induction on n, the number of steps in the derivation S =>* w. Here is an outline of the proof, with reasons omitted. You need to supply the reasons.

Basis:

1) If n=1, then w is ε because _______.

2) w is in L because _______.

Induction:

3) Either (a) $S \Rightarrow aS \Rightarrow^{n-1} w$ or (b) $S \Rightarrow aSbS \Rightarrow^{n-1} w$ because _______.

4a) In case (a), w = ax, and $S \Rightarrow^{n-1} x$ because _______.

5a)
In case (a), x is in L because _______.

6a)
In case (a), w is in L because _______.

4b)
In case (b), w can be written w = aybz, where S =>^{p} y and S =>^{q} z for some p and q less than n because _______.

5b)
In case (b), y is in L because _______.

6b)
In case (b), z is in L because _______.

7b)
In case (b), w is in L because _______.

For which of the steps above the appropriate reason is contained in the following argument:
"All n-step derivations of w produce either ε (for n=1) or use one of the productions with at least one nonterminal in the body (for n > 1). In case the production S → aS is used, then w=ax with x being produced by a (n-1)-step derivation. In case the production S → aSbS is used then w=aybz with y and z being produced by derivations with number of steps less than n."

a) 2

b) 6a

c) 4a

d) 5b

You did not answer this question.

12. Let G be the grammar:

```
S → SS | (S) | ε
```

L(G) is the language BP of all strings of balanced parentheses, that is, those strings that could appear in a well-formed arithmetic expression. We want to prove that L(G) = BP, which requires two inductive proofs:

1. If w is in L(G), then w is in BP.
2. If w is in BP, then w is in L(G).

We shall here prove only the second. You will see below a sequence of steps in the proof, each with a reason left out. These reasons belong to one of three classes:

A)
Use of the inductive hypothesis.

B)
Reasoning about properties of grammars, e.g., that every derivation has at least one step.

C)
Reasoning about properties of strings, e.g., that every string is longer than any of its proper substrings.

The proof is an induction on the length of w. You should decide on the reason for each step in the proof below, and then identify from the available choices a correct pair consisting of a step and a kind of reason (A, B, or C).

Basis: Length = 0.

(1)
The only string of length 0 in BP is ε because ________

(2)
ε is in L(G) because ________
Induction: |w| = n > 0.

(3)
w is of the form (x)y, where (x) is the shortest proper prefix of w that is in BP, and y is the remainder of w because ________

(4)
x is in BP because ________

(5)
y is in BP because ________

(6)
|x| < n because ________

(7)
|y| < n because ________

(8)
x is in L(G) because ________
(9)
y is in L(G) because ________
(10)
(x) is in L(G) because ________
(11)
w is in L(G) because ________

a) (4) for reason C
b) (6) for reason A
c) (10) for reason C
d) (4) for reason A

You did not answer this question.

13. Suppose a problem P_1 reduces to a problem P_2. Which of the following statements can we conclude to be TRUE based on the above?

a) If P_2 is undecidable, then it must be that P_1 is decidable.
b) If P_1 is undecidable, then it must be that P_2 is undecidable.
c) If P_2 is non-RE, then it must be that P_1 is non-RE.
d) If P_1 is decidable, then it must be that P_2 is undecidable.

Answer submitted: **b)**

You have answered the question correctly.

14. Here is a context-free grammar:

```
S → AB | CD
A → BG | 0
B → AD | ε
C → CD | 1
D → BB | E
E → AF | B1
F → EG | 0C
G → AG | BD
```

Find all the nullable symbols, and then use the construction from Section 7.1.3 (p. 265) to modify the grammar's productions so there are no ε-productions. The language of the grammar should change only in that ε will no longer be in the language.

a) F → EG | 0C
b) G → AG | BD | A | B | D
c) G → AG | BD | A | G | B | D | ε
d) S → AB | CD | A | B | C

You did not answer this question.

15. There is a Turing transducer *T* that transforms problem P1 into probem P2. *T* has one read-only input tape, on which an input of length *n* is placed. *T* has a read-write scratch tape on which it uses O(S(n)) cells. *T* has a write-only output tape, with a head that moves only right, on which it writes an output of length O(U(n)). With input of length *n*, *T* runs for O(T(n)) time before halting. You may assume that each of the upper bounds on space and time used are as tight as possible.

A given combination of S(n), U(n), and T(n) may:

1. Imply that *T* is a polynomial-time reduction of P1 to P2.
2. Imply that *T* is NOT a polynomial-time reduction of P1 to P2.
3. Be impossible; i.e., there is no Turing machine that has that combination of tight bounds on the space used, output size, and running time.

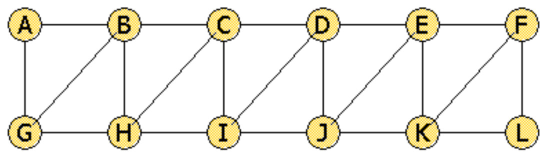
What are all the constraints on S(n), U(n), and T(n) if *T* is a polynomial-time reducer? What are the constraints on feasibility, even if the reduction is not polynomial-time? After working out these constraints, identify the true statement from the list below.

a) $S(n) = n^2$; $U(n) = 1$; $T(n) = n^{10}$ is a polynomial-time reduction.
b) $S(n) = n$; $U(n) = n^2$; $T(n) = 2^n$ is not physically possible.
c) $S(n) = \log_2 n$; $U(n) = n^2$; $T(n) = n^4$ is possible, but not a polynomial-time reduction.
d) $S(n) = n^2$; $U(n) = 1$; $T(n) = n^{10}$ is not physically possible.

Answer submitted: **a)**

You have answered the question correctly.

16. What is the size of a minimal node cover for the graph below?



Identify one of the minimal node covers below.
a) {B,D,E,F,G,H,J,K}
b) {A,C,E,H,I,L}
c) {A,C,D,F,G,H,J,K}
d) {A,B,C,D,E,F,H,J,L}

You did not answer this question.

17. The language of regular expression (0+10)* is the set of all strings of 0's and 1's such that every 1 is immediately followed by a 0. Describe the complement of this language (with respect to the alphabet {0,1}) and identify in the list below the regular expression whose language is the complement of L ((0+10)*).
a) (0+1)*1(ε+11(0+1)*)
b) (0+10)*(1+11(0+1)*)
c) (0+10)*11(0+10)*
d) (0+10)*1(ε+11(0+1)*)

You did not answer this question.

18. The polynomial-time reduction from SAT to CSAT, as described in Section 10.3.3 (p. 452), needs to introduce new variables. The reason is that the obvious manipulation of a boolean expression into an equivalent CNF expression could exponentiate the size of the expression, and therefore could not be polynomial time.

Suppose we apply this construction to the expression (u+(vw))+x, with the parse implied by the parentheses. Suppose also that when we introduce new variables, we use y1, y2,...

After constructing the corresponding CNF expression, identify one of its clauses from the list below. Note: logical OR is represented by +, logical AND by juxtaposition, and logical NOT by -.
a) (y3+y2+u)
b) (y2+-y1+v)
c) (-y2+y1+w)
d) (-y2+y1+v)

Answer submitted: **b)**

You have answered the question correctly.

19. Consider the following identities for regular expressions; some are false and some are true. You are asked to decide which and in case it is false to provide the correct counterexample.

(a) R(S+T)=RS+RT
(b) (R*)*=R*
(c) (R*S*)*=(R+S)*
(d) (R+S)*=R*+S*
(e) S(RS+S)*R=RR*S(RR*S)*
(f) (RS+R)*R=R(SR+R)*

a) (d) is false and a counterexample is: R={a,ε},T={b}, S={a,ε}

b) (e) is true

c) (d) is false and a counterexample is: R={a},T={a}, S={b}

d) (b) is false and a counterexample is: R={ab},T={a}, S={b}

You did not answer this question.

20. Which of the following problems about a Turing Machine *M* does Rice's Theorem imply is undecidable?

a) Does *M* make more than three moves when started with a blank tape?

b) Does *M* ever write the symbol 1 on its tape?

c) Is the language of *M* context-free?

d) Is the language of *M* recursively enumerable?

Answer submitted: **d)**

Your answer is incorrect.

This property of languages is trivial --- it is true of all recursively enumerable languages. Thus, Rice's Theorem does not imply it is undecidable, and in fact it is a decidable problem. See the discussion of Rice's Theorem in Section 9.3.3 (p. 397).

21. Which among the following languages is not regular (cannot be defined by a regular expression or finite automaton)?

a) L={x | x=$a^m b^n$, n, m positive integers}

b) L={x | x=$a^m b^n c^k$, n, m, k positive integers}

c) L={x | x=$a^m (bc^k)^n$, n, m, k positive integers}

d) L={x | x=$(ab^4c)^n$, n a positive integer}

Answer submitted: **c)**

You have answered the question correctly.

22. Here is the transition table of a DFA:

	0	**1**
→A	E	D
*B	A	C
C	G	B
D	E	A
*E	H	C
F	C	B
G	F	E
H	B	H

Find the minimum-state DFA equivalent to the above. Then, identify in the list below the pair of equivalent states (states that get merged in the minimization process).

a) F and H

b) E and G

c) B and E

d) A and B

You did not answer this question.

23. Consider the grammars:

G_1: S → AB, A → aAA|ε , B → abBB|ε
G_2:S → CB, C → aCC|aC|a, B → abBB|abB|ab
G_3:S → CB|C|B| ε , C → aCC|aC|a, B → abBB|abB|ab
G_4:S → ASB|ε, A → aA|ε, B → abB|ε
G_5:S → ASB|AB, A → aA|a, B → abB|ab
G_6:S → ASB|aab, A → aA|a, B → abB|ab

Describe the language of each of these grammars. Then, identify from the list below a pair of grammars that define the same language?

a) G_3 and G_2

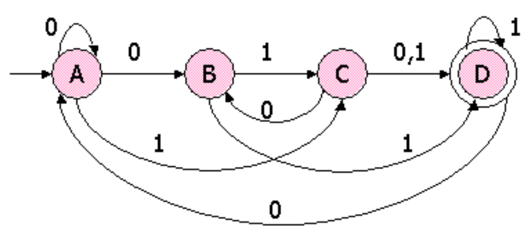
b) G_3 and G_5

c) G_3 and G_4

d) G_1 and G_5

You did not answer this question.

24. Here is a nondeterministic finite automaton:



Convert this NFA to a DFA, using the "lazy" version of the subset construction described in Section 2.3.5 (p. 60), so only the accessible states are constructed. Which of the following sets of NFA states becomes a state of the DFA constructed in this manner?

a) {A,D}

b) {A,B,C,D}

c) {C,D}

d) The empty set

You did not answer this question.

25. Suppose one transition rule of some PDA P is δ(q,0,X) = {(p,YZ), (r,XY)}. If we convert PDA P to an equivalent context-free grammar G in the manner described in Section 6.3.2 (p. 247), which of the following could be a production of G derived from this transition rule? You may assume *s* and *t* are states of P, as well as *p, q*, and *r*.

a) [qXr] → 0[qXs][sYr]

b) [qXr] → 0[rXs][qYr]

c) [qXr] → 0[qYs][sZp]

d) [qXr] → 0[rXs][sYr]

Answer submitted: **d)**

You have answered the question correctly.