Gradiance Online Accelerated Learning

Zayd

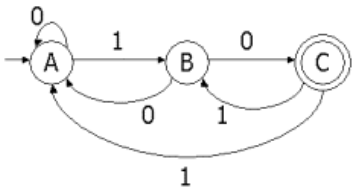
- Home Page
- Assignments Due
- Progress Report
- Handouts
- Tutorials
- Homeworks
- Lab Projects
- Log Out

Help

Submission number: 86209
Submission certificate: EJ400998
Submission time: 2014-05-15 01:04:44 PST (GMT - 8:00)

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

1. 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) 1+101 represents the paths from C to B that do not go through A.
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*1 represents the paths from A to B that do not go through C.

You did not answer this question.

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

a) G_1:S → AC|C, A → b
b) G_1:S → AB|C|a, A → b, C → a
c) G_1:S → d|a, A → c|b|ε
d) G_1:S → AB|a, A → c, B → ε

You did not answer this question.

3. A *linear grammar* is a context-free grammar in which no production body has more than one occurrence of one variable. For example, A → 0B1 or A → 001 could be productions of a linear grammar, but A → BB or A → A0B

could not. A *linear language* is a language that has at least one linear grammar.

The following statement is false: "The concatenation of two linear languages is a linear language." To prove it we use a counterexample: We give two linear languages L_1 and L_2 and show that their concatenation is not a linear language. Which of the following can serve as a counterexample?

a) $L_1 = \{w|w=aaa(aaa)^{n-1}(ab)$, where n is a positive integer$\}$
$L_2 = \{w|w=c(aaa)^n$, where n is a positive integer$\}$

b) $L_1 = \{w|w=(ac)^n b^n$, where n is a positive integer$\}$
$L_2 = \{w|w=c^n d^n$, where n is a positive integer$\}$

c) $L_1 = \{w|w=aaa(aaa)^{n-1}(ab)$, where n is a positive integer$\}$
$L_2 = \{w|w=c(aaa)^n c^n d^n$, where n is a positive integer$\}$

d) $L_1 = \{w|w=aaac^{n-1}(ab)$, where n is a positive integer$\}$
$L_2 = \{w|w=c(aaa)^n$, where n is a positive integer$\}$

You did not answer this question.

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

δ(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	$\{\}$	$\{\}$	$\{\}$

Deduce what *M* does on any input of 0's and 1's. Demonstrate your understanding by identifying, from the list below, the ID that CANNOT be reached on some number of moves from the initial ID $q_0$10100101.

a) 0111111111q_1

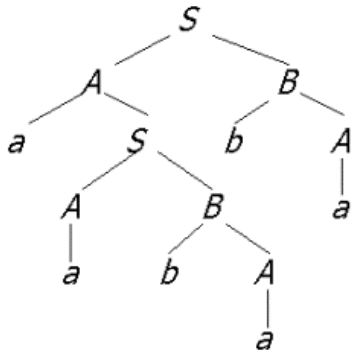
b) 0q_f1110101

c) 0111$q_2$0101

d) 0$q_2$01101101

You did not answer this question.

5. 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) aABbA
b) aaBba
c) aSbA
d) aABB

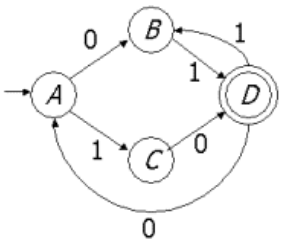
You did not answer this question.

6. The homomorphism *h* is defined by h(a) = 01 and h(b) = 10. What is h (baab)?

a) baab
b) 10101010
c) 100110
d) 10010110

You did not answer this question.

7. Examine the following DFA:



Identify in the list below the string that this automaton accepts.

a) 01111
b) 0010110
c) 0110
d) 10001

You did not answer this question.

8. The proof that the Independent-Set problem is NP-complete depends on a construction given in Theorem 10.18 (p. 460), which reduces 3SAT to Independent Sets. Apply this construction to the 3SAT instance:

(u+v+w)(-v+-w+x)(-u+-x+y)(x+-y+z)(u+-w+-z)

Note that - denotes negation, e.g., -v stands for the literal NOT v. Also, remember that the construction involves the creation of nodes denoted [*i*,*j*]. The node [*i*,*j*] corresponds to the *j*th literal of the *i*th clause. For example, [1,2] corresponds to the occurrence of *v*.

After performing the construction, identify from the list below the one pair of nodes that does NOT have an edge between them.

a) [3,2] and [4,1]

b) [3,1] and [5,1]

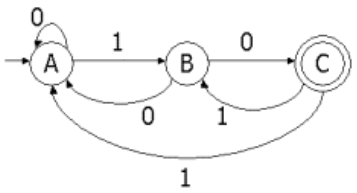
c) [2,3] and [4,1]

d) [5,1] and [5,2]

Answer submitted: **c)**

You have answered the question correctly.

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) {B}

b) {C}

c) {A,C}

d) {}

You did not answer this question.

10. Identify in the list below a sentence of length 6 that is generated by the grammar S → (S)S | ε

a))) ( ) ((

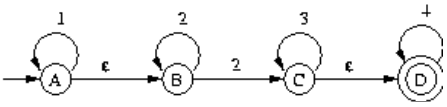
b) ( ) ( ) ( )

c) ( ) ( )) (

d)))) (((

You did not answer this question.

11. Here is a nondeterministic finite automaton with epsilon-transitions:



Suppose we use the extended subset construction from Section 2.5.5 (p. 77) to convert this epsilon-NFA to a deterministic finite automaton with a dead state, with all transitions defined, and with no state that is inaccessible from the start state. Which of the following would be a transition of the DFA?

Note: we use S-x->T to say that the DFA has a transition on input x from state S to state T.

a) {A,B,C}-3->{C,D}

b) {B,C,D}-3->{C}

c) {B,C,D}-2->{C,D}

d) {D}-3->{}

You did not answer this question.

12. Consider the grammar G and the language L:

G: S → AB | a | abC, A → b, C → abC | c

L: {w | w a string of a's, b's, and c's with an equal number of a's and b's}.

Grammar G does not define language L. To prove, we use a string that either is produced by G and not contained in L or is contained in L but is not produced by G. Which string can be used to prove it?

a) abba

b) ababc

c) abacccc

d) cacacab

You did not answer this question.

13. Suppose there are three languages (i.e., problems), of which we know the following:

1. L1 is in P.
2. L2 is NP-complete.
3. L3 is not in NP.

Suppose also that we do not know anything about the resolution of the "P vs. NP" question; for example, we do not know definitely whether P=NP. Classify each of the following languages as (a) Definitely in P, (b) Definitely in NP (but perhaps not in P and perhaps not NP-complete) (c) Definitely NP-complete (d) Definitely not in NP:

- L1 [union] L2.
- L1 ∩ L2.
- L2cL3, where c is a symbol not in the alphabet of L2 or L3 (i.e., the *marked concatenation* of L2 and L3, where there is a unique marker symbol between the strings from L2 and L3).
- The complement of L3.

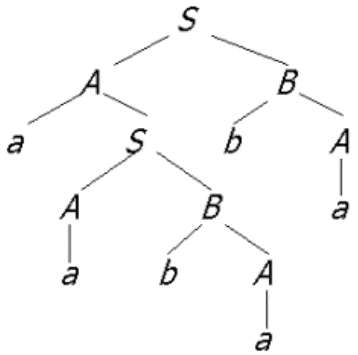
Based on your analysis, pick the correct, definitely true statement from the list below.

a) The complement of L3 is definitely not NP-complete.
b) L1 ∩ L2 is definitely NP-complete.
c) L1 [union] L2 is definitely in NP.
d) L1 [union] L2 is definitely NP-complete.

Answer submitted: **c)**

You have answered the question correctly.

14. Here is a parse tree that uses some unknown grammar G.



Which of the following productions is surely one of those for grammar G?

a) S → AbA
b) S → A
c) S → aSbA
d) B → bA

You did not answer this question.

15. Which of the following strings is NOT in the Kleene closure of the language {011, 10, 110}?

a) 1101010
b) 10110110
c) 11001010
d) 01110011

You did not answer this question.

16. 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 first. 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 number of steps in the derivation 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: One step.

(1)
The only 1-step derivation of a terminal string is S => ε because _____

(2)
ε is in BP because _____
Induction: An n-step derivation for some n>1.

(3)
The derivation $S =>^n w$ is either of the form
(a) $S => SS =>^{n-1} w$ or of the form
(b) $S => (S) =>^{n-1} w$
because _____
Case (a):

(4)
w = xy, for some strings x and y such that $S =>^p x$ and $S =>^q y$, where $p<n$ and $q<n$ because _____

(5)
x is in BP because _____

(6)
y is in BP because _____

(7)
w is in BP because _____
Case (b):

(8)
w = (z) for some string z such that $S =>^{n-1} z$ because _____

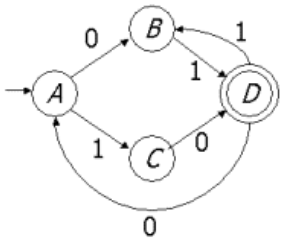
(9)
z is in BP because _____

(10)
w is in BP because _____

a) (2) for reason B
b) (9) for reason A
c) (7) for reason B
d) (2) for reason A

You did not answer this question.

17. Here is a finite automaton:



Which of the following regular expressions defines the same language as the finite automaton? Hint: each of the correct choices uses component expressions. Some of these components are:

1. The ways to get from A to D without going through D.
2. The ways to get from D to itself, without going through D.
3. The ways to get from A to itself, without going through A.

It helps to write down these expressions first, and then look for an expression that defines all the paths from A to D.

a) (01+10)(11+0(01+10))*

b) (01+10)(11*+0(01+10))*

c) (01+10)(11+(01+10)0)*

d) ((01+10)0)*(01+10)(11)*

Answer submitted: **a)**

You have answered the question correctly.

18. For the grammar:

```
S → AB | CD
A → BC | a
B → AC | C
C → AB | CD
D → AC | d
```

1. Find the generating symbols. Recall, a grammar symbol is *generating* if there is a deriviation of at least one terminal string, starting with that symbol.

2. Eliminate all *useless productions* --- those that contain at least one symbol that is not a generating symbol.

3. In the resulting grammar, eliminate all symbols that are not *reachable* --- they appear in no string derived from S.

In the list below, you will find several statements about which symbols are generating, which are reachable, and which productions are useless. Select the one that is FALSE.

a) B is generating.

b) B → C is useless.

c) S is not generating.

d) D is not reachable.

Answer submitted: **a)**

You have answered the question correctly.

19. We can represent questions about context-free languages and regular languages by choosing a standard encoding for context-free grammars (CFG's) and another for regular expressions (RE's), and phrasing the question as recognition of the codes for grammars and/or regular expressions such that their languages have certain properties. Some sets of codes are decidable, while others are not.

In what follows, you may assume that G and H are context-free grammars with terminal alphabet {0,1}, and R is a regular expression using symbols 0 and 1 only. You may assume that the problem "Is L(G) = (0+1)*?", that is, the problem of recognizing all and only the codes for CFG's G whose language is all strings of 0's and 1's, is undecidable.

There are certain other problems about CFG's and RE's that are decidable, using well-known algorithms. For example, we can test if L(G) is empty by finding the pumping-lemma constant n for G, and checking whether or not there is a string of length n or less in L(G). It is not possible that the shortest string in L(G) is longer than n, because the pumping lemma lets us remove at least one symbol from a string that long and find a shorter string in L(G).

You should try to determine which of the following problems are decidable, and which are undecidable:

- Is Comp(L(G)) equal to (0+1)*? [Comp(L) is the complement of language L with respect to the alphabet {0,1}.]
- Is Comp(L(G)) empty?
- Is L(G) intersect L(H) equal to (0+1)*?
- Is L(G) union L(H) equal to (0+1)*?
- Is L(G) finite?
- Is L(G) contained in L(H)?
- Is L(G) = L(H)?
- Is L(G) = L(R)?
- Is L(G) contained in L(R)?
- Is L(R) contained in L(G)?

Then, identify the true statement from the list below:

a) "Is L(G) finite?" is decidable.

b) "Is Comp(L(G)) equal to (0+1)*?" is undecidable.

c) "Is L(G) = L(R)?" is decidable.

d) "Is L(H) contained in L(G)?" is decidable.

Answer submitted: **b)**

Your answer is incorrect.

Hint: notice that this question is the same as asking whether L(G) is empty. Testing emptiness of a context-free language is discussed in Section 7.4.3 (p. 302).

20. A *k-clique* in a graph G is a set of *k* nodes of G such that there is an edge between every pair of nodes in the set. The problem *k*-CLIQUE is: Given a graph G and a positive integer *k*, does G have a *k*-clique?

We can prove *k*-CLIQUE to be NP-complete by reducing the 3SAT problem to it. Consider the 3-CNF expression:

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

[Note: *a*' denotes the negation NOT(*a*) of variable *a*.] Let G be the graph constructed from this expression as in Theorem 10.18 (p. 460). Then, let H be the *complement* of G, that is, the graph with the same nodes as G and an

edge between two nodes if and only if G DOES NOT have an edge between those two nodes. Let us denote the vertices of H by using the corresponding (clause, literal) pair. The node (i,j) corresponds to the j^{th} literal of the i^{th} clause. Which of the following nodes form a maximum-sized clique in H?

a) (1,2),(2,1),(4,3),(3,2),(4,2)}
b) {(1,1),(2,1),(2,3),(3,2),(4,3)}
c) {(1,2),(2,1),(3,1),(3,2),(4,3)}
d) {(1,2), (2,1), (3,2), (4,3)}

Answer submitted: **d)**

You have answered the question correctly.

21. The language L = {ss | s is a string of a's and b's} is not a context-free language. In order to prove that L is not context-free we need to show that for every integer n, there is some string z in L, of length at least n, such that no matter how we break z up as z = *uvwxy*, subject to the constraints |vwx| ≤ n and |vx| > 0, there is some $i \geq 0$ such that uv^iwx^iy is not in L.

Let us focus on a particular z = aabaaaba and n = 7. It turns out that this is the wrong choice of z for n = 7, since there are some ways to break z up for which we can find the desired *i*, and for others, we cannot. Identify from the list below the choice of u,v,w,x,y for which there is an *i* that makes uv^iwx^iy not be in L. We show the breakup of aabaaaba by placing four |'s among the a's and b's. The resulting five pieces (some of which may be empty), are the five strings. For instance, aa|b||aaaba| means u=aa, v=b, w=ε, x=aaaba, and y=ε.

a) |a|abaa|a|ba
b) a|aba|a|aba|
c) |aa|b|aa|aba
d) a|a|baaab|a|

Answer submitted: **d)**

You have answered the question correctly.

22. 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 is one state that has one in-transition on input 1.
b) There are two states that have one in-transition on input 0.
c) There are two states that have no in-transitions on input 0.
d) There is one state that has three in-transitions on input 1.

You did not answer this question.

23. In the following expressions, - represents negation of a variable. For example, -x stands for "NOT x"), + represents logical OR, and juxtaposition represents logical AND (e.g., (x+y)(y+z) represents (x OR y) AND (y OR z).

Identify the expression that is satisfiable, from the list below.

a) (z+-y)(z+y)(-z+-y)(-z+y)

b) (-z+y)(-z+-y)(y+x)(z)

c) (z+y)(-z)(z+-y)(y+x)

d) (z+-y)(-y+x)(-z)(-x+z)

Answer submitted: **d)**

You have answered the question correctly.

24. Consider the pushdown automaton with the following transition rules:

1. $\delta(q,0,Z_0) = \{(q,XZ_0)\}$
2. $\delta(q,0,X) = \{(q,XX)\}$
3. $\delta(q,1,X) = \{(q,X)\}$
4. $\delta(q,\varepsilon,X) = \{(p,\varepsilon)\}$
5. $\delta(p,\varepsilon,X) = \{(p,\varepsilon)\}$
6. $\delta(p,1,X) = \{(p,XX)\}$
7. $\delta(p,1,Z_0) = \{(p,\varepsilon)\}$

The start state is *q*. For which of the following inputs can the PDA first enter state *p* with the input empty and the stack containing XXZ_0 [i.e., the ID (p,ε,XXZ_0)]?

a) 0011011

b) 1100101

c) 111001

d) 1001101

You did not answer this question.

25. Which of the following grammars derives a subset of the language:

{x | x contains a and c in proportion 4:3 and there are no two consecutive c's}?

a) S → acacaca S → SaScSaScSaScSaS S → SaSaSaScSaScSaS

b) S → ε S → SaScSaScSaSaSaS

c) S → ε S → SacSaScSaScSaS

d) S → ε S → aScScaSaScSaS

You did not answer this question.