CS109B Final Examination

Tuesday, June 9, 12:15 – 3:15 pm

Directions

The exam is *open book*; any written materials may be used. Answer all 16 questions on the exam paper itself. The total number of points is 300. You may make additional comments at the bottoms of pages or at the end if you feel the need to clarify or justify your answers. Do not forget to **sign the pledge** below.

I acknowledge and accept the honor code.

____

Print your name here: ____

1		5		9		13	
2		6		10		14	
3		7		11		15	
4		8		12		16	

Problem 1: (15 points) Suppose we have an undirected graph of n nodes, whose names are $v_1, v_2, \ldots, v_n$. Suppose there is an edge between each pair of distinct nodes, and for all i and j the weight of the edge between nodes v_i and v_j is $i + j$. Describe below the minimum-weight spanning tree for this graph. You may wish to draw a picture and can do so on the lines or in the margin. If you wish to be considered for partial credit, be sure to include some justification.

Problem 2: (15 points) If G is an **acyclic** directed graph, Is it always possible to construct for G a depth-first search forest consisting of

a) A single tree (assuming we pick the starting node judiciously)? ________ Give a justification or a counterexample.

b) As many trees as G has nodes (again, assuming we pick the roots judiciously each time we need a new root)? ________ Give a justification or counterexample.

Problem 3: (15 points) Suppose G is a directed graph with nonnegative weights on the arcs, upon which we run Dijkstra's algorithm with source node v_1, and suppose that the order in which we settle the nodes is $v_1, v_2, \ldots, v_n$. Let us assume that every node is reachable from the source node and that there are no two nodes that have the same shortest distance from the source, so the order of settlement is unique and not subject to any choice of which node to settle in one round. Now suppose we add the arc $a \rightarrow b$, which doesn't exist in G, and give it the weight 0 (note that a and b are nodes in G). Call the new graph G^+. If we run Dijkstra's algorithm on G^+ with source node v_1, under what circumstances will the round on which node b is settled in G^+ differ from its round in G?

Briefly justify your answer.

Problem 4: (20 points) Consider the grammar

$$<A> \to <B>a<B>$$
$$<B> \to b<B> \mid <B>a \mid \epsilon$$

a) How many parse trees are there for the string $baab$? ________ Draw them in the space below (to save time, you may omit the triangular brackets around $<A>$ and $<B>$).

b) Give a regular expression that defines the language $L(<B>)$.

c) Give a regular expression that defines the language $L(<A>)$.

Problem 5: (15 points) Convert the nondeterministic automaton

2

$\{a, b\}$ $\{a, b\}$

$\to$ 1 b 4

b b

3

to a deterministic automaton in the space below.

Problem 6: (20 points) Give regular expressions for the following languages.

a) The set of all strings of a's whose length, when divided by 3, leaves a remainder of 1.

---

b) The set of strings of a's and b's that both begin and end with a.

---

c) The set of strings of a's and b's such that every a is either preceded immediately by a b or followed immediately by a b (or both).

Problem 7: (30 points) Consider the grammar G:

$$<A> \rightarrow <A>0<A> \mid 1$$

and let L be the language $L(<A>)$. Also, let M denote the language consisting of all those strings that start with 1 and alternate 0's and 1's, ending in 1, that is, the language of the regular expression $\mathbf{1(01)}^*$. We'd like to show that $L = M$. To do so, we prove by induction that $L \subseteq M$ and then prove by induction that $M \subseteq L$.

The first induction will be a complete induction on the height of a parse tree for string x in the grammar G. That is, we assume x is in L (and therefore has a parse tree) and we show it is in M. To show x is in M we only have to argue that it has the right form; we do not have to refer to the regular expression given for M.

a) What is a suitable basis case for the induction? ____________

b) Prove the basis case.

---

c) State the inductive step.

---

d) Prove the inductive step.

Now, we need to prove that if x is in M then x is also in L. We could try an induction on length of x, but since all strings in M are of odd length, there are some tricky details. Instead we shall prove what we need by induction on the number of 0's in x.

e) What is a suitable basis case for the induction? ________

f) Prove the basis case.

g) State the inductive step.

h) Prove the inductive step. You may wish to draw parse trees to support your proof and may use the lines or margins.

Problem 8: (15 points) The grammar G discussed in Problem 7 is not parsable by the methods discussed in this course.

a) Find another grammar for the same language L that *does* have a table-driven parser.

b) Give the parsing table for your grammar in the space below.

Problem 9: (15 points) Transform the logical expression $ab + cd$, which is a sum of products, into a product of sums (product of clauses) expression, using the algebraic laws from Section 12.8. Show each step on a line, and justify it, either by citing the number of an algebraic law from Section 12.8 or giving its name. Also, you may use the associative and commutative laws for `AND` and `OR` without stating them explicitly and without using a line to show the transformations.

Problem 10: (20 points) Below is a Karnaugh map.

$$\begin{array}{cc|cccc} & & & rs & & \\ & & 00 & 01 & 11 & 10 \\ \hline & 00 & 1 & 1 & 1 & 1 \\ pq & 01 & 1 & 0 & 0 & 1 \\ & 11 & 0 & 1 & 0 & 1 \\ & 10 & 1 & 1 & 0 & 1 \end{array}$$

a) List the prime implicants (do not list any nonprime implicants), giving the AND-of-literals form for each.

---

b) Only a subset of the prime implicants are needed to cover the map, that is, to include all the points with 1's. Give a minimal such set below.

Problem 11: (20 points) A logical expression E is said to be *satisfiable* if there is some (at least one) truth assignment for the variables of E that makes E true. Put another way, E is satisfiable if and only if `NOT` E is *not* a tautology.

a) Give an example of an expression that is *not* satisfiable.

---

b) Each of the following expressions *is* satisfiable. For each, give a truth assignment that shows it is satisfiable.

i) $(p \rightarrow q) \equiv (q \rightarrow p)$.

---

ii) $(p + q)(\bar{p} + r)(\bar{q} + \bar{r})$.

Problem 12: (20 points) We wish to prove the tautology $\big((p \equiv q)\ \texttt{AND}\ (q \equiv r)\big) \rightarrow (p \equiv r)$ by resolution and proof-by-contradiction.

a) Express $p \equiv q$ as a product of clauses. *Hint*: Remember $\equiv$ can be expressed in terms of $\rightarrow$, which can in turn be expressed using AND-OR-NOT.

---

b) Express $q \equiv r$ as a product of clauses.

---

c) Express $\texttt{NOT}\ (p \equiv r)$ as a product of clauses.

---

d) Take the clauses produced in (a), (b), and (c), and show how to derive `false`, i.e., the empty clause.

Problem 13: (15 points) We would like to prove the generalized DeMorgan law

$$S(n)\text{: } \big(\texttt{NOT}(p_1 p_2 \cdots p_n)\big) \equiv (\bar{p_1} + \bar{p_2} + \cdots + \bar{p_n})$$

by induction on n. We shall provide the steps; you must fill in the reason why each step is valid, for example, giving a law from Section 12.8 of the text or invoking the inductive hypothesis, along with any other needed explanation.

BASIS: For $n = 1$, the generalized DeMorgan law says $(\texttt{NOT}\ p_1) \equiv \bar{p_1}$. This expression is a tautology because:

INDUCTION: We assume $S(n)$ and prove $S(n+1)$. To begin,

$$\Big(\texttt{NOT}\big((p_1 p_2 \cdots p_n)(p_{n+1})\big)\Big) \equiv \big(\texttt{NOT}(p_1 p_2 \cdots p_n) + \bar{p}_{n+1}\big) \tag{1}$$

Why?

Next, we can write

$$\Big(\texttt{NOT}\big((p_1 p_2 \cdots p_n)(p_{n+1})\big)\Big) \equiv (\bar{p}_1 + \bar{p}_2 + \cdots + \bar{p}_{n+1}) \tag{2}$$

Why?

While (2) is not identical to $S(n+1)$, it can be made so in a simple way. Explain:

Problem 14: (25 points) We would like to design a circuit that determines whether at least two of the n inputs $x_1, x_2, \ldots, x_n$ are 1. To do so in a divide and conquer way, we need to design a basis circuit for the case $n = 1$ and a way of combining two circuits for n inputs into one circuit for $2n$ inputs (we may assume n is a power of 2). To make this strategy work, we need to have our circuits produce more than a single output.

a) What outputs will suffice so that a divide-and-conquer design is possible? You should choose one of your outputs to be the desired answer ("at least two 1's"), and try to use as few additional outputs as you can. Describe their intuitive meanings below, giving them names for further reference.

b) In the space below design the basis circuit for $n = 1$.

c) In the space below, show how to combine two circuits for n inputs each into one $2n$-input circuit.

d) Write a recurrence relation for $D(n)$, the depth of your circuit for n inputs.

e) Give a big-oh solution to your recurrence.

Problem 15: (20 points) Consider the expression of predicate logic:

$$(\forall X)\Big((\exists Y)\big(arc(X,Y) \texttt{ AND } arc(Y,X)\big)\Big)$$

a) Identify all the atomic formulas.

---

b) Identify all the literals.

---

c) Which occurrences of Y are bound to the quantifier $(\exists Y)$? (Circle these occurrences.)

d) Which occurrences of X are bound to the quantifier $(\forall X)$? (Point to these occurrences with arrows.)

e) If G is a directed graph such that $arc(a,b)$ is true if and only if there is an arc in G from node a to node b, then what does the above expression assert about G?

Problem 16: (20 points) If a directed graph is acyclic, we do not need Dijkstra's algorithm to find the minimum distance from a source node to each other node; we can do so with an $O(m)$ algorithm (as customary, n is the number of nodes and m is the larger of the number of nodes and the number of arcs).

a) On the lines below give an informal or formal (pseudo-Pascal) description of such an algorithm. *Hint*: Start by finding a topological order for the nodes of the acyclic graph. Remember that the source node may not be able to reach every node (the distance is ∞ for the unreachable nodes). Also note that it is not necessarily the case that the source node will be first in the topological order.

b) Explain why your algorithm takes only $O(m)$ time.