Midterm

Monday, February 27, 2023 9:02 AM

Exam Instructions

The exam is composed of ten problems.
(Each problem is worth 20 points.)

Please write an answer for each problem and label each part of the answer with the problem number and in some cases the appropriate letter (a-d).
You may type your answers into a document
or write them with pen and paper.

Once you are ready to submit, please put your answers into
a single PDF file and upload the file to Learning Suite
(under the Midterm Exam assignment link).

Please upload your solutions to Learning Suite
by 10:00 pm on February 28, 2023.

Please follow the limits below as you work on the exam:

1. Open Book
2. Open Notes
3. Closed Everything Else (internet, other people, etc)

Please don't get help from others as you work on the exam.
Please don't share the contents of the exam with anyone else.
I am counting on you and your honor to prevent cheating on this exam.

Thank you!

Problem 1. (20 points)

Consider the following grammar and input string.
The terminals in the grammar are enclosed in single quotes.
Follow parts a-d below to show the first four steps of a
stack-based parser when reading the string using the grammar.

Grammar:

L -> D T
T -> ',' L | epsilon
D -> '0' | '1'

String:

1,0

The initial contents of the stack and the input are:

Stack	Input
L#	1,0#

answer	Stack	Input	Operation
	L#	1,0#	
a)	DT#	1,0#	Use Grammar Rule on Stack
b)	1T#	1,0#	Use Grammar Rule on Stack
c)	T#	,0#	Match / Pop off of both stack and input
d)	,L#	,0#	Use Grammar Rule on Stack
	L#	0#	Match / Pop off of both stack and input
	DT#	0#	Use Grammar Rule on Stack
	0T#	0#	Use Grammar Rule on Stack
	T#	#	Match / Pop off of both stack and input
	#	#	Use Grammar Rule on Stack - **FINISH**

a. Give the contents of the stack and the input after one parser step.

b. Give the contents of the stack and the input after a second parser step.

c. Give the contents of the stack and the input after a third parser step.

d. Give the contents of the stack and the input after a fourth parser step.

Problem 2. (20 points)

The following grammar is an ambiguous grammar for expressions.
The terminals in the grammar are enclosed in single quotes.
Write a new grammar for the same language that is not ambiguous.
Construct the grammar so that it enforces correct operator precedence and associativity.
The precedence of the operators is given by the precedence table.
(Each row in the table is a different level of precedence.)
All binary operators are left-associative except for '^' which is right-associative.

E -> '!' E
E -> E '==' E
E -> E '!=' E
E -> E '^' E
E -> '(' E ')'
E -> 'x'

Precedence Table

highest	!
	^
lowest	== !=

E-> E '==' A
E -> E '!=' A
E -> A

A -> B '^' A
A->B

B -> '!' B
B -> C

C-> '(' E ')'
C-> 'x'

Problem 3. (20 points)

Consider the following grammar.
The terminals in the grammar are enclosed in single quotes.
Follow parts a-d below to give the contents of some of the positions
in the parse table for a table-driven parser for the grammar.

E -> A X
X -> '=' E
X -> epsilon
A -> B Y
Y -> '<' B Y
Y -> '>' B Y
Y -> epsilon
B -> C Z
Z -> '++' Z
Z -> epsilon
C -> '(' E ')'
C -> '4'

	=	<	>	++	(	)	4	#
E					c: AX		AX	
X	= E	ε	ε	ε	ε	ε	ε	d: ε
A					BY		BY	
Y	ε	< BY	> BY	ε	ε	ε	ε	
B					C Z		C Z	
Z	ε	ε	ε	a: ++Z	ε	ε	ε	
C					(E)	b: BLANK/ ERROR	4	

a. Give the content of the parse table at the single position on the row indexed by the nonterminal **Z** and the column indexed by the terminal **'++'** ++Z

b. Give the content of the parse table at the single position on the row indexed by the nonterminal **C** and the column indexed by the terminal **')'** BLANK / ERROR

c. Give the content of the parse table at the single position on the row indexed by the nonterminal **E** and the column indexed by the terminal **'('** AX

d. Give the content of the parse table at the single position on the row indexed by the nonterminal **X** and the column indexed by the endmarker **'#'** ε

Problem 4. (20 points)

Consider the following grammar and input string.
The nonterminals in the grammar are { E }.
The terminals are { (,),+,-,*,/,x,2 }. The start symbol is E.
Follow parts a-d below to show the first four steps of
a Derivation of the string using the grammar.

Grammar:

E -> (E + E) | (E - E) | (E * E) | (E / E)
E -> x | 2

String:

((x / 2) * 2)

The initial string of grammar symbols in the Derivation is: **E**

a. Give the string of grammar symbols that results from applying **one** grammar rule to the **initial string**. (E * E)

b. Give the string of grammar symbols that results from applying **one** grammar rule to the **leftmost** nonterminal in the answer for **part a**. ((E / E) * E)

c. Give the string of grammar symbols that results from applying **one** grammar rule to the **leftmost** nonterminal in the answer for **part b**. ((x / E) * E)

d. Give the string of grammar symbols that results from applying **one** grammar rule to the **leftmost** nonterminal in the answer for **part c**. ((x / 2) * E)

Problem 5. (20 points)

For each string and regular expression (regex) pair given in parts a-d,
answer 'yes' if the string matches the pattern defined by the regex,
or 'no' if the string doesn't match.

a.

string: abaa
regex: b*(ab|aa)* YES

b.

string: abaa
regex: ab*a* YES

c.

string: abab
regex: a(ba)*b* YES

d.

string: abab
regex: b(aa)*(ba)* NO

Problem 6. (20 points)

Let the following propositional variables represent the associated propositions.

J - Joe took the jewelry
L - Mrs. Krasov lied
C - A crime was committed
T - Mr. Krasov was in town

Given these premises:

If Joe took the jewelry or Mrs. Krasov lied, a crime was committed.
Mr. Krasov was not in town.
If a crime was committed, Mr. Krasov was in town.

And this conclusion:

Therefore Joe did not take the jewelry.

a. Write the premises and the conclusion as logic expressions.

b. Write the premises as clauses (disjunctions of literals).

c. Prove the conclusion using resolution and proof-by-contradiction.
Give a justification for each line in the proof.
Each line in the proof must be either a premise or
the result of applying the resolution rule.
Do not use any other inference rules in the proof.

a)

Premises
========
J v L -> C
!T
C - > T

Conclusion
==========
!J

b)

!J v !L v C
!T
!C v T

c)

2 ways:

1	J	PBC
2	!J v !L v C	Premise
3	!L v C	Resolution: 1,2
4	!C v T	Premise
5	!L v T	Resolution: 3,4
6	!T	Premise
7	T	Resolution: 5,6
8	**FALSE**	**Resolution: 6,7**

1	J	PBC
2	!J v !L v C	Premise
3	!L v C	Resolution: 1,2
4	C	Resolution: 2,3
5	!C v T	Premise
6	!T	Premise
7	T	Resolution: 5,6
8	**FALSE**	**Resolution: 6,7**

Problem 7. (20 points)

Consider the following facts, rules, and queries.

Facts:

```
child('Tim','Bea').
child('Sue','Ned').
child('Ned','Bea').
child('Ann','Tim').
```

Rules:

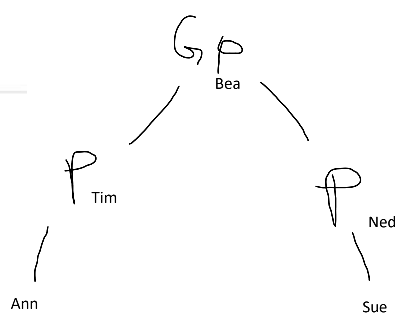
```
cousin(C1,C2) :- grandchild(C1,GP), grandchild(C2,GP).
grandchild(C,GP) :- child(C,P), child(P,GP).
```

Queries:

```
cousin('Ann','Sue')?
```

a. Write the rules as clauses (disjunctions of literals).

b. Write one formal proof of the query using **contradiction, instantiation, and resolution.**
Give a justification for each line in your proof.
Each step must be a premise or the result of instantiation or resolution.



co() = cousin()
gc() = grandchild()
ch() = child()

c1 = child1
c2 = child2
gp = grandparent
p = parent

a)

co(c1,c2) or NOT gc(c1,gp) or NOT gc(c2,gp)
gc(c,gp) or NOT ch(c,p) or NOT ch(p,gp)

b)

1	NOT co(Ann, Sue)	Negate Query
2	co(Ann,Sue) or NOT gc(Ann,Bea) or NOT gc(Sue,Bea)	Instantiation: Rule1
3	NOT gc(Ann,Bea) or NOT gc(Sue,Bea)	Resolution: 1,2
4	gc(Ann,Bea) or NOT ch(Ann,Tim) or NOT ch(Time,Bea)	Instantiation: Rule2
5	NOT ch(Ann,Tim) or NOT ch(Time,Bea)	Resolution: 2,3
6	ch(Ann, Tim)	Premise
7	NOT ch(Tim,Bea)	Resolution: 5,6
8	ch(Tim,Bea)	Premise
9	**FALSE**	**Resolution: 7,8**

Problem 8. (20 points)

In this problem you will use mathematical induction to prove (for all integers n >= 1):

P(n): 1*2*3 + 2*3*4 + ... + n(n+1)(n+2) = n(n+1)(n+2)(n+3)/4

(We will complete the base case for you as follows.)
Consider the base case when n = 1.
In this case, the summation on the left side of P(n) contains just the first term (1*2*3) and has the value 6.
When n has value 1, the right side of P(n) becomes 1(1+1)(1+2)(1+3)/4 and also has the value 6.
Thus, the statement is true for the base case.

Complete parts a-d below to show that the statement works for the inductive case.

a. State the inductive hypothesis.

b. State what is to be proven in the inductive step.

c. Give the result of using the inductive hypothesis (from part a) to make a substitution into the summation side of part b.

d. Use algebra to show that the result for part c is equivalent to the right side of part b.

a)

P(k) = 1*2*3 + 2*3*4 + ... + k(k+1)(k+2) = k(k+1)(k+2)(k+3)/4

b)

P(k+1) = 1*2*3 + 2*3*4 + ... + (k+1)((k+1)+1)((k+1)+2) = (k+1)((k+1)+1)((k+1)+2)((k+1)+3)/4

c)

k(k+1)(k+2)(k+3)/4 + (k+1)((k+1)+1)((k+1)+2) = (k+1)((k+1)+1)((k+1)+2)((k+1)+3)/4

d)

k(k+1)(k+2)(k+3)/4 + (k+1)(k+2)(k+3) = (k+1)(k+2)(k+3)(k+4)/4
=> k(k+1)(k+2)(k+3)/4 + 4(k+1)(k+2)(k+3)/4 = (k+1)(k+2)(k+3)(k+4)/4
=> (k+1)(k+2)(k+3) * (k + 4))/4 = (k+1)(k+2)(k+3)(k+4)/4

=> **(k+1)(k+2)(k+3)(k + 4)/4 = (k+1)(k+2)(k+3)(k+4)/4**

Problem 9. (20 points)

The domain of possible values for variables X and Y is **{ Jim, Ann, Sal, Pat, Tom }.**
The following facts define the values for which the child predicate is true.
The child predicate is false for all other cases.

child (Ann, Jim)
child (Sal, Jim)
child (Pat, Ann)
child (Tom, Sal)

Evaluate each expression using the domain values and predicates as defined and indicate if the expression is true or false.

a. (∃ X) child(X, Jim)

b. (∀ X) not child(X, Jim)

c. (∃ X)(∃ Y) (child(X, Jim) -> child(Y, X))

d. (∀ Y)(∀ X) (child(X, Jim) -> child(Y, X))

a)

someone is a child of Jim
Ann and Sal are both children of Jim
TRUE

b)

nobody is a child of Jim
Ann and Sal are both children of Jim
FALSE

c)

Somebody who is a child of Jim also has a child
Patt is a child of Ann, who is a child of Jim
TRUE

d)

If X is a child of Jim, then Y is a child of X.
Not necessarily true for all values. Sal is a child of Jim, and while Tom is a child of Tom, Pat is not a child of Tom.
FALSE

Problem 10. (20 points)

Follow parts a-d below to show the steps of algebraically transforming the initial expression into the final expression.

Initial expression:

(not A and not B) -> not B

Final expression:

True

In each step below, use only one application of one law (not including double negation, commutative, or associative laws) from the Table of Algebraic Laws.
Do not use any laws not provided in the table.
You may use double negation, commutative, and associative laws as an added part of any step without stating it's use.
In each step, the application of a law needs to lead toward the final expression.
Note that in each step you need to use one of the laws other than double negation, commutative, and associative.
In other words, if a step only uses double negation, commutative, and associative laws, you will not earn points for the step.

a. Starting with the initial expression, give the result of applying one law and identify the law that you used.

b. Give the result of applying one law to the expression given in **part a** and identify the law that you used.

c. Give the result of applying one law to the expression given in **part b** and identify the law that you used.

d. Give the result of applying one law to the expression given in **part c** and identify the law that you used.

Table of Algebraic Laws

P and True <=> P	Identity
P or False <=> P	Identity
P or True <=> True	Domination
P and False <=> False	Domination
P or P <=> P	Idempotent
P and P <=> P	Idempotent
not (not P) <=> P	Double Negation
P or Q <=> Q or P	Commutative
P and Q <=> Q and P	Commutative
(P or Q) or R <=> P or (Q or R)	Associative
(P and Q) and R <=> P and (Q and R)	Associative
P or (Q and R) <=> (PvQ) and (PvR)	Distributive
P and (Q or R) <=> (P^Q) or (P^R)	Distributive
not (P and Q) <=> not P or not Q	DeMorgan's
not (P or Q) <=> not P and not Q	DeMorgan's
P or not P <=> True	Negation
P and not P <=> False	Negation
P -> Q <=> not P or Q	Implication

	(NOT A ^ NOT B) -> NOT B	Initial
a	NOT(NOT A ^ NOT B) v NOT B	Implication
b	A v B v NOT B	DeMorgan's
c	A v TRUE	Negation
d	TRUE	Domination