Final Examination, Cmput 325
Dec 17, 2001

LAST NAME ___________________________________

FIRST NAME ___________________________________

STUDENT ID ___________________________________

INSTRUCTIONS:

- This is a closed book exam. The time is 2 hours, and the marks total is 100.
- Partial marks will be given only if I can determine that you had a good idea how to do it. There will be no partial marks for "efforts". So, a good strategy is to try to do well on the questions that you think you know how to answer.
- In answering a short answer question, usually some key words or one sentence could be sufficient.
- Use the back of a page when you need more space to write your answers. Remember to draw an arrow pointing to it.
- You can expect to have your grades by the evening of Dec 19th, posted to the course web page. Note that I will be out of town from Dec 21st, and return when the winter semester starts.
- Good luck, and enjoy the holidays.

QUESTION 1 (30 marks): ______________________

QUESTION 2 (20 marks): ______________________

QUESTION 3 (20 marks): ______________________

QUESTION 4 (22 marks): ______________________

QUESTION 5 (8 marks): ______________________

TOTAL:_____________________ OUT OT 100

1. [30 marks]

Consider a definition of `path/2` on a graph, by the logic program P below.

```
path(X,Y) :- connected(X,Y).
path(X,Z) :- connected(X,Y), path(Y,Z).
connected(X,Y) :- arc(X,Y).
connected(X,Y) :- arc(Y,X).
arc(1,2).
arc(2,2).
arc(2,4).
arc(3,2).
arc(5,6).
```

[4 marks] When P is considered a logical formula, it entails some other formulas. Show **all** of the ground instances of `connected(X,Y)` such that for each of these instances α, we have $P \models \alpha$.

[6 marks] Suppose you set up Prolog in the trace mode. Show the trace generated by Prolog for the first answer to the goal: `?- path(1,3).` Below we start with the first couple of calls. (You don't need to show the level numbers generated in a trace).

```
?- path(1,3).
     Call: path(1,3) ?
     Call: connected(1,3) ?
```

[4 marks] Consider propositional logic. Let $\Gamma = \{\neg a \vee (b \wedge c),\ a \vee \neg b \vee \neg c,\ b \vee \neg c\}$. Is it the case that $\Gamma \models a$? Verify your answer using a truth table.

[4 marks] For each pair of lists below (in Prolog syntax), indicate whether they are unifiable. In the case that you answer is yes, also show a unifier.

```
(i)   [A,B,C]    [[a,b]|L]    your answer _________________________________

(ii)  [a,b|[]]   [[A,B]]      your answer _________________________________
```

[4 marks] Suppose in constructing a Prolog program, we write a clause

```
p(X) :- q1(X), !, q2(X).
```

Now, someone argues that you should have place the cut at the end, as

```
p(X) :- q1(X), q2(X), !.
```

Construct an example that shows these two ways of placing the cut in this clause gives you different effects. You should first write a program, and then add each of the above to it to produce two programs, so that the two programs behave differently for the goal `?- p(W)`. *Requirement: excluding the clauses above, your constructed program should have no more than 5 clauses.*

[2 marks] Consider the following Prolog program.

```
p([],Q,Q).
p([A|L],S,SL) :-
    A > 5,
    !,
    p(L,[A|S],SL).
p([A|L],S,SL) :- p(L,S,SL).
```

Show all the answers generated by Prolog for the following goal

```
?- p([2,5,7,9,6],[],Res).
```

[6 marks] Given the following program, for each goal below, show the result of Prolog execution. In the case that the goal is proved, you should also show the bindings for the variables in the goal. You only need to show the first answer generated by Prolog.

```
r(e, L).
r(c(X,L1),c(X,L2)) :- r(L1,L2).
q(L1,L2) :- r(L1,L2).
q(L1,c(Y,L2)) :- q(L1,L2).
```

```
?- q(c(d,e), c(a,c(b,W))).
```

```
?- q(c(u,W), c(U,c(b,d))).
```

```
?- q(c(a,b), c(A,c(B,e))).
```

2. (20 marks)

[2 marks] What are the three ingredients of a constraint satisfaction problem (CSP)? That is, when you formulate a problem as a CSP, what are the three things you must specify?

[2 marks] What is a *constraint store*? You may answer this question by saying who is processing the constraint store at runtime.

[5 marks] The following is a constraint logic program without constraints. The purpose is to test your understanding of the basic solving method of CLPFD. Show **all** the answers that will be returned by Sicstus Prolog for the goal: `?- solve(S)`. Show them in the order in which they are generated.

```
:- use_module(library(clpfd)).
solve(Sol) :-
    Sol = [V1,V2,V3],
    domain([V1],2,3),
    domain([V2,V3],5,6),
    labeling([],Sol).

% goal
?- solve(S).
```

[5 marks] Given a list of variables over some domain, we want these variables to be assigned so that the resulting list is in increasing order. Let's write a constraint logic program using CLPFD for this.

```
:- use_module(library(clpfd)).
ordered(N,Sol) :-
    length(Vars,N),  % length/2 is a built-in predicate
    domain(Vars,1,9),
    constraint(Vars),
    labeling([],Vars),
    Sol=Vars.
```

You should define `constraint(Vars)` so that `Vars` is a list of values in increasing order. For example, if `N=5` then `Sol=[1,2,3,4,5]` is a solution. *Requirement: since in general the value of N can be arbitrary, you should give a recursive definition.*

[6 marks] Consider three variables and their domains as given below

X: its domain is $\{2,3,4,5,6\}$
Y: its domain is $\{4,5,6,7\}$
Z: its domain is $\{4,5,6,7\}$

and the constraints

$$Y > Z + 1 \quad \text{and} \quad X \leq Y - 1$$

For each variable, indicate which of its domain values cannot participate in any solution. *Do not guess! Wrong answers can reduce your marks gained from correct answers.*

3. (20 marks) The purpose of this question it to demonstrate your competence in writing small Prolog programs. So, do not try to avoid solving the problem by thinking of using built-in predicates. But you may use the familiar built-in predicates append/3 and member/2 without defining them. The quality of your answer is as important as whether it works or not.

[6 marks] Given a list L of elements, each of which represents a course and its enrollment, e.g. `[cmput325, [john, lily, ken,...]]`, define a predicate

```
courses_taken(+L,+Name,-Courses)
```

such that given such a list L and a student Name, Courses is bound to a list of courses taken by the student.

[8 marks] This question could be difficult for some students. The subset/2 predicate given by Bratko is defined only for ordered sets. Now, we want to define a genuine subset predicate

```
subset(+S,-S1)
```

so that, upon backtracking, `S1` is bound to every subset of `S`. The order in which these subsets are generated is unimportant. *Requirement: Your program must not generate redundant subsets. Hint: My solution consists of five clauses.*

[2 marks] Given subset/2 as defined above, write a goal that uses findall/3 to find all subsets of a list, say [a,b,c,d].

[4 marks] Given a list L of lists, we want to remove those lists in L that are not minimal in the following sense: L1 in L is minimal if there is no other list L2 in L such that every element of L2 is an element in L1. Define a predicate

```
minimize(+L,-LL)
```

such that LL is L except that non-minimal lists in L are removed. *Hint: you may use the predicate subset/2 defined above.*

4. (22 marks)

[6 marks] Consider the following Lisp definition.

```
(defun f (L)
    (if (null (cdr L))
        L
        (cons (car L)
              (cons (f (cddr L)) (cadr L)))
    )
)
```

Show the result of each call below. (Be careful, the output may involve the dot notation.)

```
(f '(a b))          your answer ______________________

(f '(a b c))        your answer ______________________

(f '(a b c d))      your answer ______________________
```

[5 marks] Consider evaluating the following lambda expression using the context-based interpreter.

```
((lambda (f) (f 2 3)) (lambda (x y) (* x (+ y 1))))
```

(i) Show the context when the subexpression `(f 2 3)` is evaluated.

(ii) Show the context when the subexpression `(* x (+ y 1))` is evaluated.

[2 marks] Is the λ-expression $(\lambda xy \mid w)$ equivalent to $(\lambda x \mid w)$? Why?

[7 marks] Write a Lisp program

```
(defun prefix (L) ......)
```

which returns a list of all prefixes of L. The order in which the prefixes appear in the resulting list is unimportant. E.g.

```
(prefix '(a b c)) => (nil (a) (a b) (a b c))  or  ((a b c) (a b) (a) nil)
```

5. (8 marks)

[2 marks] Explain the purpose of each stack in the SECD machine.

`S:`

`E:`

`C:`

`D:`

[6 marks] Consider the expression

```
(if (> 7 6) (+ (- 4 3) 2) 9)
```

Compile this expression to SECD machine instructions, and then show how they are executed on the SECD machine.