York University- Department of Computer Science and Engineering

SC/CSE 3401 3.00 – Functional and Logic Programming

Solutions to assignment 1

1) (5 marks) Consider the following formulae:

$$A: \quad p \to q \to r$$
$$B: \quad p \to r \to q$$
$$C: \quad p \land \neg(r \to q)$$

Using truth tables, show which of the following sets are satisfiable and why?

a) {A, B}
b) {B, C}
c) {A, B, C}

Answer.

	p	q	r	$q \to r$	A: $p \to q \to r$	$r \to q$	B: $p \to r \to q$	$\neg(r \to q)$	C: $p \land \neg(r \to q)$
1	**f**	**f**	**f**	t	t	t	t	**f**	**f**
2	**f**	**f**	t	t	t	**f**	t	t	**f**
3	**f**	t	**f**	**f**	t	t	t	**f**	**f**
4	**f**	t	t	t	t	t	t	**f**	**f**
5	t	**f**	**f**	t	t	t	t	**f**	**f**
6	t	**f**	t	t	t	**f**	**f**	t	t
7	t	t	**f**	**f**	**f**	t	t	**f**	**f**
8	t	t	t	t	t	t	t	**f**	**f**

a) Set {A, B} is satisfiable. All rows of the truth table except rows 6 and 7 show states which can make both formulae true.
b) Set {B, C} is not satisfiable (it is inconsistent). No state (none of the rows in the truth table) can make all formulae in this set true.
c) Set {A,B,C} is not satisfiable either. It has an inconsistent subset.

2) (5 marks) Consider the following formula in the sets domain:

$$(\forall x)(\forall y)(\forall z)((\{a\} \cup x \subseteq y \cup x) \land (y \cup x \subseteq z \cup x) \to (\{a\} \cup x \subset z \cup x))$$

(a) List all the terms in the above formula that are not object variable or object constants.
(b) List all atomic formulas in the above formula.
(c) Is the formula semantically true? If yes, say why and if no, provide a counter example.

Answer.

(a) The terms in the above equation which are not object variables or object constants, are terms constructed with functions. They are:

$$\{a\} \bigcup x, \quad y \bigcup x, \quad z \bigcup x$$

(b) The atomic formulas in above formula are:

$$\{a\} \bigcup x \subseteq y \bigcup x, \quad y \bigcup x \subseteq z \bigcup x, \quad \{a\} \bigcup x \subset z \bigcup x$$

(c) The above formula is not semantically true. Here is a counter-example:

Assume x is { } (the empty set) and both y and z are {a}. If the formula is true for all x, y, z it must be true for the assumed values of x, y, z as well. Substituting in the formula we have:

$$((\{a\} \bigcup x \subseteq y \bigcup x) \wedge (y \bigcup x \subseteq z \bigcup x) \rightarrow (\{a\} \bigcup x \subset z \bigcup x))$$
$$\Rightarrow ((\{a\} \bigcup \{\} \subseteq \{a\} \bigcup \{\}) \wedge (\{a\} \bigcup \{\} \subseteq \{a\} \bigcup \{\}) \rightarrow (\{a\} \bigcup \{\} \subset \{a\} \bigcup \{\}))$$
$$\Rightarrow (\{a\} \subseteq \{a\}) \wedge (\{a\} \subseteq \{a\}) \rightarrow (\{a\} \subset \{a\})$$

The left side of the implication is true and the right side is false, resulting in a false. We therefore showed with a counterexample that the formula is not true.

---

3)(6 marks) Convert the following formulae in propositional logic to logic programming clauses. Indicate which ones are Horn clauses. Clearly show ALL steps. $((p \rightarrow q) \vee r) \equiv (q \wedge s)$

Answer.

Convert to CNF first :

$$((p \rightarrow q) \vee r) \equiv (q \wedge s)$$

1 - Remove implications and equivalences

$$\Rightarrow \big(((p \rightarrow q) \vee r) \rightarrow (q \wedge s)\big) \wedge \big((q \wedge s) \rightarrow ((p \rightarrow q) \vee r)\big)$$
$$\Rightarrow \big(\neg((\neg p \vee q) \vee r) \vee (q \wedge s)\big) \wedge \big(\neg(q \wedge s) \vee ((\neg p \vee q) \vee r)\big)$$

2 - Move negations inward

$$\Rightarrow \big((p \wedge \neg q \wedge \neg r) \vee (q \wedge s)\big) \wedge \big((\neg q \vee \neg s) \vee (\neg p \vee q \vee r)\big)$$

3 - Distribute OR over AND

$$\Rightarrow \big((p \vee (q \wedge s)) \wedge (\neg q \vee (q \wedge s)) \wedge (\neg r \vee (q \wedge s))\big) \wedge \big(\neg q \vee \neg s \vee \neg p \vee q \vee r\big)$$
$$\Rightarrow (p \vee q) \wedge (p \vee s) \wedge (\neg q \vee q) \wedge (\neg q \vee s) \wedge (\neg r \vee q) \wedge (\neg r \vee s) \wedge \big(\neg q \vee \neg s \vee \neg p \vee q \vee r\big)$$

Write as logic programming clauses :

$$\Rightarrow \begin{cases} p;q:-. \\ p;s:-. \\ q:-q. \\ s:-q. \\ q:-r. \\ s:-r. \\ q;r:-q,s,p. \end{cases}$$

The first, second, and seventh clauses are not Horn clauses. The rest are Horn clauses. Since all the resulting clauses are not Horn clauses, converting to logic programming clauses fails.

---

4) (8 marks) Convert the following formula in predicate logic <u>to logic programming clauses</u>. Indicate which ones are Horn clauses. Clearly show <u>ALL</u> steps (m and n are predicates):

$$\big((\exists x)(\forall y)m(x,y)\big)\rightarrow\big((\forall x)(\exists y)n(y,x)\big)$$

Answer.

Convert to CNF first:	$\big((\exists x)(\forall y)m(x,y)\big)\rightarrow\big((\forall x)(\exists y)n(y,x)\big)$
1- Remove implication	$\Rightarrow \neg\big((\exists x)(\forall y)m(x,y)\big)\vee\big((\forall x)(\exists y)n(y,x)\big)$
2- Move negations inward	$\Rightarrow \big((\forall x)(\exists y)\neg m(x,y)\big)\vee\big((\forall x)(\exists y)n(y,x)\big)$
3- Rename variables	$\Rightarrow \big((\forall x)(\exists y)\neg m(x,y)\big)\vee\big((\forall z)(\exists w)n(w,z)\big)$
4- Prenex Normal Form	$\Rightarrow (\forall x)(\exists y)(\forall z)(\exists w)\big(\neg m(x,y)\vee n(w,z)\big)$
5- Skolemizing	$\Rightarrow (\forall x)(\forall z)\big(\neg m(x,f(x))\vee n(g(z),z)\big)$
6- Distribute OR over AND (nothing to do)	
7- Remove Universal Quantifiers	$\Rightarrow \neg m(x,f(x))\vee n(g(z),z)$

Writing as Logic Programming Clauses: $n(g(z),z):-m(x,f(x)).$

This is a Horn clause (a rule).

---

5) (5 marks) (a) Using what we covered about arithmetic in Prolog, write a simple predicate **convert(Pounds, Kilos)** that can convert weight in pounds to kilograms. (b) Write a query to get your weight in kilograms given your weight in pounds. (c) Write a query to get your weight in pounds given your weight in Kilos. This query will not return an answer to you, why?

Answer.

```
(a)  convert(Pounds, Kilos):- Kilos is Pounds * 0.45359.
```

```
(b) :- convert(200, K).
(c) :- convert(P, 100).
```

This will return an error. The operator 'is' evaluates the right hand side and matches it with its left argument. Yet, the right hand side of 'is' is not instantiated, therefore evaluation is not possible.

---

6) (14 marks) Consider the following program:

C0: findS(X, s(L1, L2)):- findL(X, L1).

C1: findS(X,s(L1,L2)):- findL(X, L2).

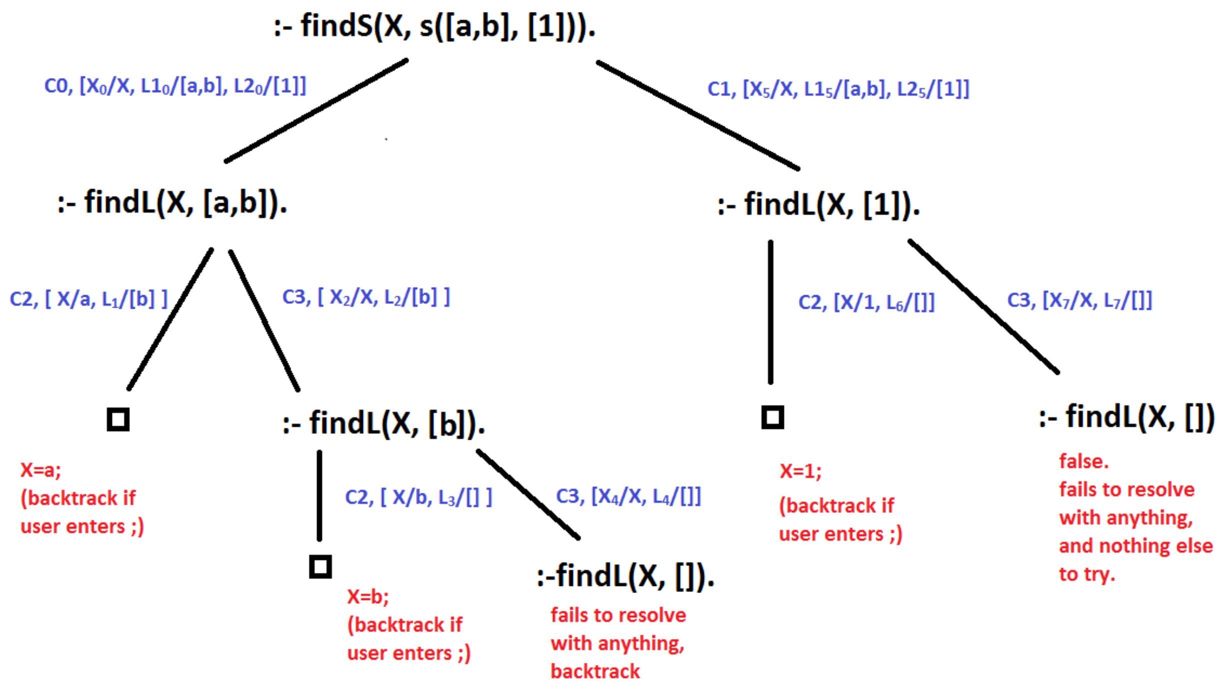
C2: findL(X, [X|L]).

C3: findL(X, [_|L]):- findL(X,L).

Draw the complete search tree for this query:

G0: :- findS(X, s([a,b], [1])).

Label all branches. Under each leaf node, mention why backtracking occurs. If there are any outputs by Prolog, indicate them too.


:- findS(X, s([a,b], [1])).
C0, [X0/X, L10/[a,b], L20/[1]]
C1, [X5/X, L15/[a,b], L25/[1]]
:- findL(X, [a,b]).
:- findL(X, [1]).
C2, [X/a, L1/[b]]
C3, [X2/X, L2/[b]]
C2, [X/1, L6/[]]
C3, [X7/X, L7/[]]
X=a; (backtrack if user enters ;)
:- findL(X, [b]).
X=1; (backtrack if user enters ;)
:- findL(X, [])
false. fails to resolve with anything, and nothing else to try.
C2, [X/b, L3/[]]
C3, [X4/X, L4/[]]
X=b; (backtrack if user enters ;)
:-findL(X, []).
fails to resolve with anything, backtrack

---

7) (10 marks) Write a Prolog program by the following design specifications that given a list of numbers, returns a list of remainders of division by 8. For example:

:- mod8([8, 9, 16, 15], [0,1,0,7]). will true.

:- mod8([1,20,10,11], L). will return L=[1,4,2,3].

(a) (3 marks) Write a recursive code.
(b) (3 marks) Write an iterative code with an accumulator. (It will return the list in the reverse order, for example the second example above will return L=[3,2,4,1])
(c) (4 marks) Write the above code with difference lists, similar to the parts example in class.

```
mod81([], []).
mod81([X|L1], [Y|L2]):- Y is X mod 8, mod81(L1, L2).


mod82(L1, L2):- mod82(L1,[], L2).
mod82([], A, A).
mod82([X|L1],A, L2):- Y is X mod 8, mod82(L1,[Y|A],L2).


mod83(L1, L2):- mod83(L1,[], L2).
mod83_f(X, Hole,[Y|Hole]):- Y is X mod 8.
mod83([], Hole, Hole).
mod83([X|L1],Hole, L2):- mod83_f(X, Hole1,L2) , mod83(L1,Hole,Hole1).
```