CS 386 Artificial Intelligence Lab
22/09/2015, Tuesday
Lab Handout 7

General Instructions

- Create a folder named **YOURROLLNUM_lab7 (e.g. 131050001_lab7)** in the home directory (or at a location of your choice). You need to submit a compressed (tar.gz or tgz only) copy of this folder at the end of this lab
- The deadline for submission of part A (or whatever you are able to complete) is end of today's lab (Tuesday, 22/09/2015, 5:00 PM). Solution for part B can be submitted till (Sunday, 27/09/2015, 11:55 PM). Separate links will be provided for both submissions. Your final submission should contain all the required(three) predicates.
- More specific submission instructions are provided later.
- The problem must be solved using Prolog only.
- You **can use** in-built library functions for solving these problems.
- Strictly follow the predicate name and convention as described in the problem description and examples.
- Make sure that you read and understand the submission instructions before making a final submission. Your submission will be automatically **graded**.
- Add comments to your code. Write your Name/Roll number on top of all your source files. Explain your code briefly.
- You are encouraged to post your queries/doubts about the handout (or anything else) on the **moodle discussion forum** during the lab as well.

PART A

This part has two subtasks.

ProblemA1: A (Well-formed) propositional formula is a sequence of symbols that is part of propositional logic. Wikipedia gives the following formalism to generate *wffs* :

- Each propositional variable is, on its own, a *wff*.
- If **A** is a *wff*, then **~A** is a *wff*.
- If **A** and **B** are *wffs*, and • is any binary connective, then (A • B) is a *wff*. Here • could be (but is not limited to) the usual operators ∨, ∧, →, or ↔.

For the purpose of this lab, we restrict ourselves to ~(**negation),** v (**disjunction**), ^(**conjunction)**, and => (**implication)** only. You need not consider *wffs* involving <=> (**equivalence**) or any other connective.

The op/3 predicate in prolog can be used to define new operators:
(http://www.swi-prolog.org/pldoc/man?predicate=op/3)

```
:- op(1100, fx, '~').
:- op(1200, yfx, '^').
:- op(1300, yfx, 'v').
:- op(1400, xfx, '=>').
```

These operators can now be used to represent *wffs* in Prolog. Following are few examples:

```
1. p => q
2. (p ^ q) v ( r => ~p)
3. ~(p => q v (r ^ ~w))
```

A propositional logic formula is said to be in **Conjunctive Normal Form** if it is expressed as conjunction of clauses, where a clause is disjunction of literals. For more details read : https://en.wikipedia.org/wiki/Conjunctive_normal_form

A propositional formula in CNF can be represented in prolog as a list of conjuncts. Every conjunct in turn is a list of disjuncts.

Example:

[[p, ~r], [r], [q, ~p]] is equivalent to (p v ~r) ^ (r) ^ (q v ~p).
[[p, ~q, s], [~r, q, s]] is equivalent to (p v ~q v s) ^ (~r v q v s).

Your goal for this lab is to write a prolog predicate `cnF(wff, Clauses)` which takes a *wff* as input, converts the formula into its **CNF** and return a list of **Clauses.** The conversion process can be described in terms of several sub procedures which successively transform the formula towards its **CNF.**

1. Move negation inward
2. Eliminate => in favor of ~, ^, and v.
3. Distribute v over ^ to produce CNF.

Example Queries :

```
?- cnF( ( ~(r ^ ~p) => q), Clauses).
Clauses = [[r,q],[~p,q]]

Step 1. Move Negation Inward.
(~r v p) => q

Step 2. Eliminate => in favor of ~, ^, and v.
~(~r v p) v q

Repeat Step 1
(r ^ ~p) v q

Step 3. Distribute v over ^ to produce CNF
(r v q) ^ (~p v q)

?cnF(~((p => q) ^ (q => r)), Clauses).
Clauses = [[p],[~q],[q],[r]]
```

The name of the script should be *yourrollnumber_problemA1.pl.*

ProblemA2: Given a list of clauses, your task is to find the list of variables used in the the clauses.

Example Queries:

```
?- get_variables([[r,q],[~p,q]], L).
L = [r, p, q]
```

```
?- get_variables([[a,b],[~a],[b,c]], L).
L = [a, b, c]
```

The name of the script should be *yourrollnumber_problemA2.pl*

Note:- The order of the output does not matter.

PART B

Your task in is to write a prolog predicate to find a satisfying assignment for a propositional logic formula in CNF.
Refer : https://en.wikipedia.org/wiki/Boolean_satisfiability_problem

sat(Clauses, PosLiterals).

e.g.
?- `sat( [ [a, b] , [~a], [b, c ] ], PosLiterals ).`
PosLiterals = [b];
PosLiterals = [b,c].

The list of clauses [[a, b] , [~a], [b, c]] represents (a∨b) ^ (~a) ^ (b∨c).
A possible assignment satisfying this formula is : a = 0, b = 1, c = 0.
Please note that multiple such assignments are possible. Your predicare should output all such assignments one by one. The order in which these assignments are output do not matter.

The name of the script should be *yourrollnumber_problemB1.pl*

Submission

Compress the folder yourrollnum_lab7 using the following command

```
tar -zcvf yourrollnum_lab7.tgz yourrollnum_lab6
```

Upload the generated .tgz file as your submission for this lab.