The University of Texas at Dallas
CS 6364
Artificial Intelligence
Fall 2020
Instructor: Dr. Sanda Harabagiu
Grader/ Teaching Assistant: Maxwell Weinzierl

<u>Homework 3: 200 points (30 points extra-credit)</u>
Issued October 12, 2020
Due November 9, 2020 before midnight
Submit only in eLearning

Name: Shankaranarayanan Kallidaikuruchi Ramakrishnan
NetID: sxk190109

<u>PROBLEM 1:</u> Inference in Propositional Logic (**55 points**)

a/ A propositional 2-CNF expression is a conjunction of clauses, each containing exactly two literals, e.g.:
$(X \vee Y) \wedge (\neg X \vee Z) \wedge (\neg Y \vee W) \wedge (\neg Z \vee G) \wedge (\neg W \vee G)$
(15 points) Prove using resolution that the above sentence entails G.

<u>Solution:</u>
Clauses:
S1: $(X \vee Y)$
S2: $(\neg X \vee Z)$
S3: $(\neg Y \vee W)$
S4: $(\neg Z \vee G)$
S5: $(\neg W \vee G)$
$\alpha = G$
$\neg\alpha = \neg G$

Step1:
Applying resolution to S1 and S3 with Y, we get
X1: $X \vee W$

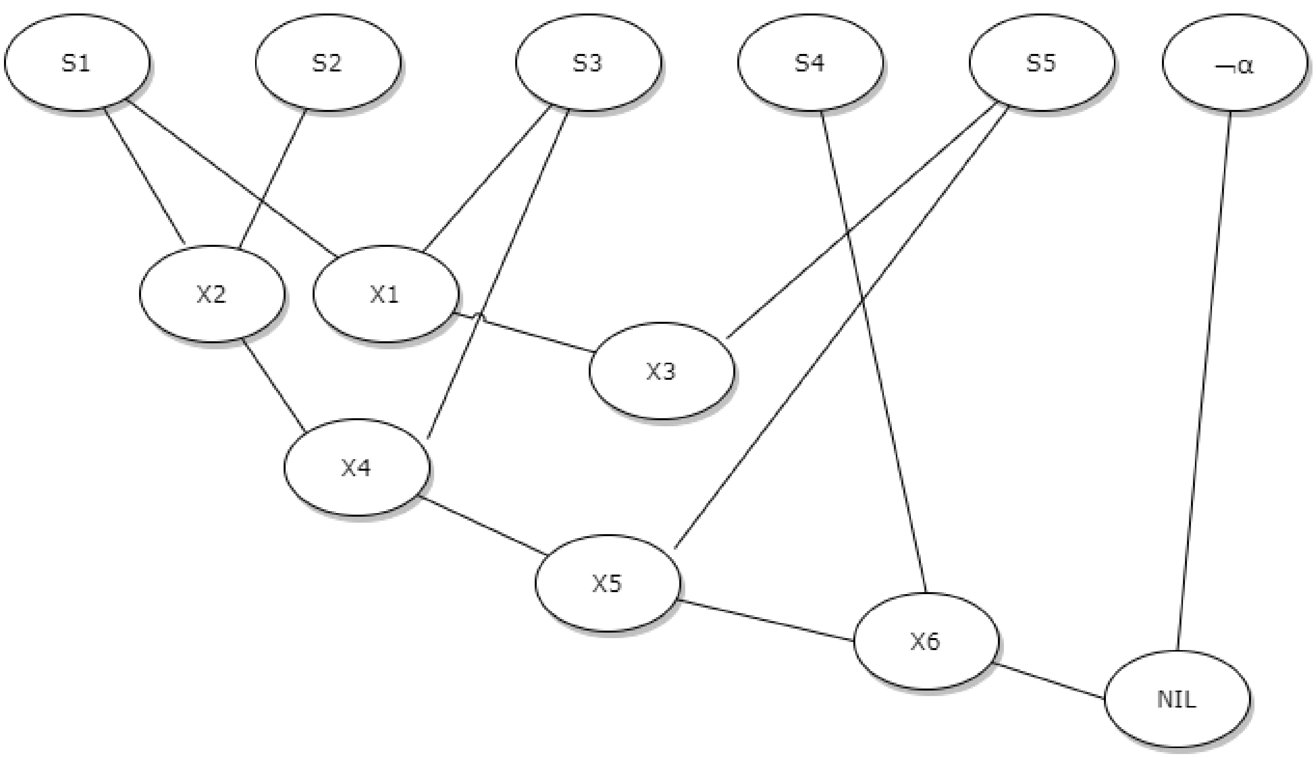
Step2:
Pairing S1 and S2 with X, we get
X2: Y $\vee$ Z
Step3:
Pairing X1 and S5, we get
X3: X $\vee$ G

Step4:
Pairing X2 and S3
X4: Z $\vee$ W

Step5:
Pairing X4 and S5
X5: Z $\vee$ G

Step6:
Pairing X5 and S4
X6: G

Step7:
Applying Resolution to X6 and $\neg\alpha$
X7: NIL



Therefore, given sentences entails G

b/ Given the following Knowledge Base:

1. $X \wedge B \wedge C \Rightarrow A$
2. $A \wedge D \wedge E \Rightarrow C$
3. B
4. $E \wedge \neg F$
5. $F \vee D$
6. $G \wedge \neg F \Rightarrow X$
7. G

(15 points) Use backward-chaining inference to prove the query A.

<u>Solution</u>

Observation: Not all sentences in KB are in Horn Clauses!

S4: $E \wedge \neg F$

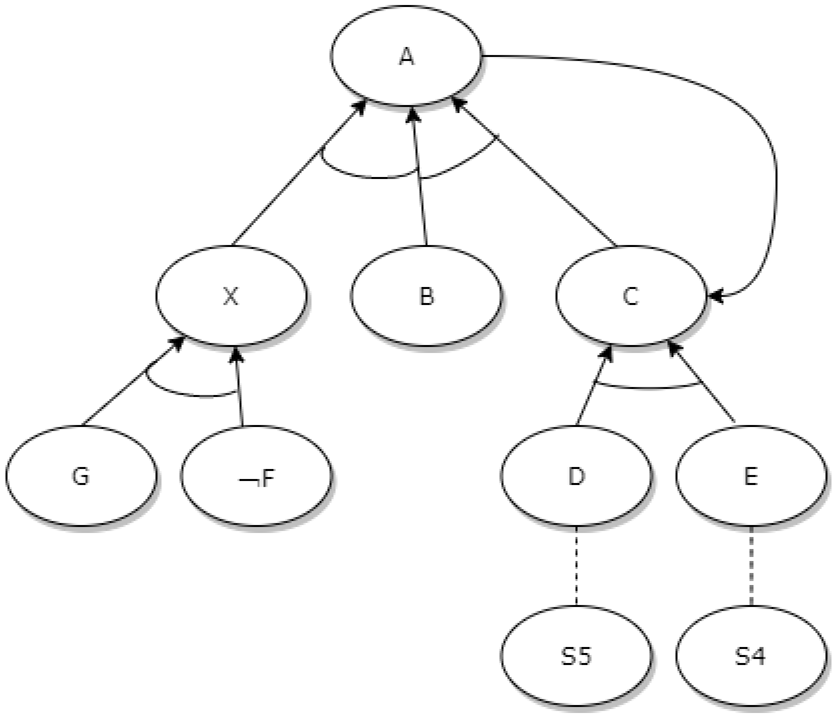
S5: $F \vee D$

We can build truth table for S4 and S5

E	F	S4
F	F	**F**
F	T	**F**
T	F	**T**
T	T	**F**

D	F	S5
F	F	**F**
F	T	**T**
T	F	**T**
T	T	**T**

AND-OR Graph:



Step 1:
Suppose A is True
According to S1, X, B and C are True
Step 2:
B is True from S3 in KB.
Step 3:
Suppose X is True
¬F is True from S6 in KB and G is True from S8 in KB.
Step 4:
Suppose C is True
According to S2 in K: A, D and E are True.
Step 5:
If D = True regardless of F, then D shall be true given S5
Step 6:
If ¬F is True and E = True, then E shall be true given S4
Step 7:
Since we assumed A is True, we get C is True.

Therefore, our assumption that A is True was correct.

c/ Use propositional logic inference rules to decide which of the following sentences are entailed by the Sentence 1: $(X \vee Y) \wedge (\neg Z \vee \neg W \vee Q)$

Sentence 2: $(X \vee Y)$

Sentence 3: $(X \vee Y \vee Z) \wedge (Y \wedge Z \wedge W \Rightarrow Q)$
Sentence 4: $(X \vee Y) \wedge (\neg W \vee Q)$

To get full credit you need to write if:

i. S1 Entails S2 or not, and if so, which propositional inference rules you have applied to reach the conclusion. Detail the results of each inference rule. **(5 points)**

Solution:
S1: $(X \vee Y) \wedge (\neg Z \vee \neg W \vee Q)$
S2: $(X \vee Y)$
By using AND elimination on S1, we get S2.
Hence, we can say that S1 entails S2.

ii. S1 Entails S3 or not, and if so, which propositional inference rules you have applied to reach the conclusion. Detail the results of each inference rule. **(5 points)**

Solution:
S1: $(X \vee Y) \wedge (\neg Z \vee \neg W \vee Q)$
S3: $(X \vee Y \vee Z) \wedge (Y \wedge Z \wedge W \Rightarrow Q)$
By using OR introduction in S1, we get
S1: $(X \vee Y \vee Z) \wedge (\neg Y \vee \neg Z \vee \neg W \vee Q)$
By using de Morgan, we get
S1: $(X \vee Y \vee Z) \wedge (\neg(Y \wedge Z \wedge W) \vee Q)$
By introducing implication in the above, we get
S1: $(X \vee Y \vee Z) \wedge (Y \wedge Z \wedge W \Rightarrow Q)$
We can say that S1 entails S3.

iii. S1 Entails S4 or not, and if so, which propositional inference rules you have applied to reach the conclusion. Detail the results of each inference rule. **(5 points)**

Solution:
S1: $(X \vee Y) \wedge (\neg Z \vee \neg W \vee Q)$
S4: $(X \vee Y) \wedge (\neg W \vee Q)$
In S1 and S4, entailment depends on $(\neg Z \vee \neg W \vee Q)$, since we cannot eliminate Z. We can say that S1 does not entail S4.

d/ Demonstrate whether the following sentences are valid, satisfiable, or neither. Motivate and detail your demonstrations.

Sentence 1: ((Smart $\vee$ Beautiful) $\Rightarrow$ (Interesting $\vee$ Boring)) $\Leftrightarrow$ ((Smart $\Rightarrow$ Interesting) $\vee$ (Beautiful $\Rightarrow$ boring)) (5 points)

Solution

We can use Truth table to check for validity and satisfiability

Smart	Beautiful	Interesting	Boring	Smart ∨ Beautiful	Interesting ∨ Boring	((Smart ∨ Beautiful) ⇒ (Interesting ∨ Boring)	Smart ⇒ Interesting	Beautiful ⇒ boring	(Smart ⇒ Interesting) ∨ (Beautiful ⇒ boring)	Sentence 1
T	T	T	T	T	T	T	T	T	T	T
T	T	T	F	T	T	T	T	F	T	T
T	T	F	T	T	T	T	F	T	T	T
T	T	F	F	T	F	F	F	F	F	T
T	F	T	T	T	T	T	T	T	T	T
T	F	T	F	T	T	T	T	T	T	T
T	F	F	T	T	T	T	F	T	T	T
T	F	F	F	T	F	F	F	T	T	F
F	T	T	T	T	T	T	T	T	T	T
F	T	T	F	T	T	T	T	F	T	T
F	T	F	T	T	T	T	T	T	T	T
F	T	F	F	T	F	F	T	F	T	F
F	F	T	T	F	T	T	T	T	T	T
F	F	T	F	F	T	T	T	T	T	T
F	F	F	T	F	T	T	T	T	T	T
F	F	F	F	F	F	T	T	T	F	F

Since we are getting False for 3 models, the **sentence is not Valid**.
Since we are getting True for other models, **the sentence is satisfiable**.

<u>Sentence 2:</u> (Tall ∨ Gorgeous) ∨ ¬(Tall ⇒ Gorgeous) **(5 points)**

Solution

Gorgeous	Tall	((Tall ∨ Gorgeous) ∨ ¬(Tall ⇒ Gorgeous))
F	F	F
F	T	T
T	F	T
T	T	T

Since we are getting False for 1 model, the **sentence is not Valid**.
Since we are getting True for other models, **the sentence is satisfiable**.

PROBLEM 2: (**25 points**) Logic Representations a/ According to political pundits, a person who is a radical (R) is electable (E) if he/she is conservative (C) , but otherwise is not electable. Which of the following are correct representations in propositional logic of this assertion? Explain why. **(15 points)**

i. $(R \wedge E) \Leftrightarrow C$

Solution:

It means that a person is both radical and electable can be conservative and a person who is conservative is both radical and electable. It is not a correct representation of the given assertion.

ii. $R \Rightarrow (E \Leftrightarrow C)$

Solution:

It means that a person who is radial, is electable if and only if the person is conservative. It is a correct representation of the assertion

iii. $R \Rightarrow ((C \Rightarrow E) \vee \neg E)$

Solution:

By applying implication elimination, we get $R \Rightarrow (C \vee E \vee \neg E)$. $E \vee \neg E$ is always True, which means that a person is electable regardless if that person is conservative. Hence it is an incorrect representation.

b/ Unification: For each pair of literals, find the Most General Unifier and the Most General Common Substitution Instance:

(**2 points**) {P(x), P(A)}

Solution:

Most General Unifier: ϴ= {x/A}
Most General Common Substitution Instance: x= A

(**4 points**) {P[f(x), y, g(y)], P[f(x), z, g(x)]}

Solution:

Most General Unifier: ϴ= {y/x}
Most General Common Substitution Instance: y=x

(**4 points**) {P[f(x, g(A,y)), g(A,y)], P[f(x,z),z]}

Solution:

Most General Unifier: ϴ = {g(A, y)/z }
Most General Common Substitution Instance: g(A, y) = z

<u>PROBLEM 3:</u> First-Order Logic (FOL) representations (**40 points**)
Write in FOL the following statements by defining first your vocabulary (i.e. predicates, constants, variables, functions, etc):

1. (**2 points**) *Some leaves turn red each Fall*.

    Leaf(x): x is a leaf
    Red(x): x is red
    Season(x): x is season
    $\exists$x [Leaf (x) $\wedge$ Season (Fall) $\wedge$ Red (x))]

2. (**3 points**) *Some trees lose all their leaves when winter comes.*

Tree(x): x is a tree
Leaf (x, y): y is a leaf of tree x
Lose (x, y):x is a tree that loses the leaf y
Season (x): x is a season
$\exists$x $\forall$y[Tree(x) $\wedge$ Leaf (x, y) $\wedge$ Season(Winter)=> Lose (x, y)]

3. (**2 points**) *Flowers are always nice and they smell lovely*.

Flower(x): x is a flower
Nice(x): x is nice
Smell_Lovely(x): x smells lovely
$\forall$ x[flower(x)=>Nice(x) $\wedge$ Smell_Lovely(x)]

4. (**2 points**) *One flower does not bring Spring.*

Flower(x): x is a flower
Season(x): x is a season
$\exists$ x $\neg$ [(Flower(x) => Season(Spring)]

5. (**2 points**) *Every flower fades at some point.*

Flower(x): x is a flower
Fade(x): x is something that fades
$\forall$ x [Flower(x)=> Fade(x)]

6. (**2 points**) *Only one flower is left in the vase.*

Flower(x): x is a flower
Vase(x): x is left inside vase
∃ x ∀ y (Flower(x) => Vase(x)) ∧ (Flower(y)=> Vase(y) => x=y

7. (**2 points**) *Every person that buys flowers is sensitive.*

Person(x): x is a person
Flower(y): y is a flower
Buys (x, y): x buys y
Sensitive(x): x is sensitive
∀x ∃ y [Person(x) ∧ Flower(y) ∧ Buys (x, y) => Sensitive(x)]

8. (**3 points**) *Poets are sensitive but they do not buy flowers, they write beautiful poems.*

Poet(x): x is a poet
Flower(y): y is a flower
Buys (x, y): x buys y
Sensitive(x): x is sensitive
Poem(x): x is a poem
Beautiful(x): x is beautiful
Writes(x,y): x writes y
∀x Poet(x) => Sensitive(x) ∧ (∀y [Flower(y) => ¬Buys(x, y)]) ∧ (∀z [Poem(z)∧ Writes(x, z) => Beautiful(z)])

9. (**2 points**) *No poet will kill an animal.*

Poet(x): x is a poet
Animal(x): x is animal
Kill (x, y): x kills y
∀x,y [Poet(x) ∧Animal(y)=> ¬ Kill (x, y)]

10. (**3 points**) *There is an agent who sells policies only to people who are not insured.*

Agent(x): x is agent
Insured(x): x is insured
Sell_policy(x, y): x sells policies to y
∃x Agent(x) => ∀y [¬ Insured(y) => Sell_policy(x, y)]

11. (**2 points**) *There is a barber who shaves all men in town who do not shave themselves.*

Barber(x): x is a barber

Man(y): y is a man
Shave (x, y): x shaves y
∀x ∀y[Barber(x)∧ Man(y) ∧ ¬ Shave (x, y) => Shave (x, y)]

12. (**10 points**) *A person born outside the US, one of who has at least one parent who is a US citizen by birth is a US citizen by descent.*

Person(x): x is a person
Born (x, y): x is born in y
Parent (x, y): x is parent of y
Citizen (x, y): x is citizen of y
∀x ∃y [Person(x)∧ ¬ Born (x, US) ∧ Parent (y, x) ∧Citizen (y, US) => Citizen (x, US)]

13. (**2 points**) *There is a flower that smells nice in the house.*

Flower(x): x is a flower
House(y): y is a house
Inside (x, y): x is inside y
Smell_Nice(x): x is a flower that smells nice
∃x,y Flower(x)∧House(y)∧ Inside (x, y) ∧Smell-Nice(x)

14. (**3 points**) *John bought only two flowers.*

Flower(x): x is a flower
Bought(x, y): x bought y
∃x,y [Flower(x) ∧ Bought(John, x) ∧ Bought(John, y) ∧ Flower(y)] ∧ ∀z [Bought(John, z) ∧ Flower(z)]=> (x=z ∨ y=z)]

<u>PROBLEM 4:</u> Refutation in First-Order Logic (**80 points**)

The purpose of this assignment is to give you experience in proving facts with the resolution method and in exposing you to Prover9, an automatic theorem prover that can help you devise your refutations.
Consider the following helpful pointers for using Prover9:

Installation (https://www.cs.unm.edu/~mccune/mace4/gui/v05.html) For linux users, install python-wxtools also.

Help Manual (https://www.cs.unm.edu/~mccune/prover9/manual/2009-02A/)

Simple tutorial (www.cs.utsa.edu/~bylander/cs5233/prover9-intro.pdf)

You are asked to solve the following puzzle.

1. Anyone who rides a Harley is a rough character.
2. Every biker rides [something that is] either a Harley or a BMW.
3. Anyone who rides any BMW is a yuppie.
4. Every yuppie is a lawyer.
5. Any nice girl does not date anyone who is a rough character.
6. Mary is a nice girl, and John is a biker.
7. (Conclusion) If John is not a lawyer, then Mary does not date John.

i. (**14 points)** Represent these clauses in first order logic, using only these predicates:

Harley(x) , Rides(x,y) , Rough(x) , Biker(x) , BMW(x) , Yuppie(x) , Lawyer(x) , Nice(x) , Date(x,y)

Solution:

1. ∀x ∀y [Rides (x, y) ∧Harley(y)=>Rough(x)]
2. ∀x [Biker(x)=> ∃y Rides (x, y)∧(BMW(y)V Harley(y))]
3. ∀x ∀y [Rides (x, y)∧ BMW(y)=> Yuppie(x)]
4. ∀x [Yuppie(x)=> Lawyer(x)]
5. ∀x ∀y [Nice(x)=> ¬ (Date (x, y) ∧Rough(y))]
6. Nice (Mary) ∧Biker (John)
7. ¬Lawyer(John)=> ¬Date(Mary, John)

ii. (**14 points)** Convert the logic sentences to clause form, skolemizing as necessary.

1. Removing ∀ and Applying implication elimination and de Morgan.
   S1: ¬Ride (x, y) ∨¬Harley(y)∨ Rough(x)
2. Eliminate ∃ by skolemizing using y = f(x)
   S2: ¬Biker(x) ∨ Rides(x, f(x))
   S3: ¬Biker(x) ∨ Harley(y) ∨ BMW(y)
3. Applying implication elimination and de Morgan
   S4: ¬Rides (x, y) ∨ ¬BMW(y) ∨ Yuppie(x)
4. Applying implication elimination and de Morgan
   S5: ¬Yuppie(x) ∨ Lawyer(x)
5. Applying implication elimination and de Morgan
   S6: ¬Nice(x) ∨ ¬Date(x, y) ∨ ¬Rough(y)
6. S7: Nice (Mary)
7. S8: Biker (John)

iii. **(42 points)** Prove by hand whether the conclusion is true by using resolution refutation (i.e. negate the conclusion and show its unsatisfiability with the rest of the knowledge base). Make sure to document the substitutions you use.

Negating the conclusion:

S9: ¬Lawyer(John)
S10: Date(Mary, John)

Resolving S2 and S8, we get:
X1: Rides (John, f(John))
ϴ= {x/ John}

Applying resolution to S3 and S8, we get:
X2: Harley(f(John)) ∨ BMW(f(John))
ϴ = {x/ John}

Applying resolution to S1 and X1, we get:
X3: ¬Harley(f(John)) ∨ Rough (John)
ϴ = {x/John, y/f(John)}

Resolving S4 and X1:
X4: BMW(f(John)) ∨ Yuppie (John)
ϴ = {x/John, y/ f(john)}

Applying resolution to S6 and S7, we get:
X5: ¬Date (Mary, y) ∨ ¬Rough(y)
ϴ = {x/Mary}

Resolving S10 and X5, we get:
X6: ¬Rough (John)
ϴ = {y/John}

Resolving S5 and S9:
X7: ¬Yuppie (John)
ϴ = {x/John}

Resolving X4 and X7:
X8: ¬BMW(f(John))

Resolving X3 and X6:
X9: ¬Harley (f(John))

Resolving X2 and X8:
X10: Harley (f (John))

Pairing X10 and X9: NIL

Therefore, if John is not a lawyer, Mary does not date him

iv.(**20 points)** Use Prover9 to perform automatically the refutation. Submit a report with three parts:

I. Assumptions and goal;
II. The input and output of prover9 (The input of prover 9 should be in plain text)
III. Conclusion

Solution:

Assumptions:

```
rides(x,y) & harley(y) -> rough(x).
biker(x) -> (exists y rides(x,y) & (bmw(y) | harley(y))).
rides(x,y) & bmw(y) -> yuppie(x).
yuppie(x) -> lawyer(x).
nice(x) -> -(date(x,y) & rough(y)).
nice(Mary) & biker(John).
```

Goal:

```
-lawyer(John) -> -date(Mary,John).
```

Input and Output:

```
============================== INPUT
=================================
assign(report_stderr,2).
set(ignore_option_dependencies).
if(Prover9).
% Conditional input included.
assign(max_seconds,60).
end_if.
if(Mace4).
% Conditional input omitted.
end_if.

formulas(assumptions).
rides(x,y) & harley(y) -> rough(x).
biker(x) -> (exists y rides(x,y)) & (bmw(y) | harley(y)).
rides(x,y) & bmw(y) -> yuppie(x).
```

```
yuppie(x) -> lawyer(x).
nice(x) -> -(date(x,y) & rough(y)).
nice(Mary) & biker(John).
end_of_list.

formulas(goals).
-lawyer(John) -> -date(Mary,John).
end_of_list.

============================== end of input
==========================

% Enabling option dependencies (ignore applies only on input).

============================== PROCESS NON-CLAUSAL FORMULAS
==========

% Formulas that are not ordinary clauses:
1 rides(x,y) & harley(y) -> rough(x) # label(non_clause).
[assumption].
2 biker(x) -> (exists y rides(x,y)) & (bmw(y) | harley(y)) #
label(non_clause).  [assumption].
3 rides(x,y) & bmw(y) -> yuppie(x) # label(non_clause).
[assumption].
4 yuppie(x) -> lawyer(x) # label(non_clause).  [assumption].
5 nice(x) -> -(date(x,y) & rough(y)) # label(non_clause).
[assumption].
6 nice(Mary) & biker(John) # label(non_clause).  [assumption].
7 -lawyer(John) -> -date(Mary,John) # label(non_clause) #
label(goal).  [goal].

============================== end of process non-clausal formulas
===

============================== PROCESS INITIAL CLAUSES
===============

% Clauses before input processing:

formulas(usable).
end_of_list.

formulas(sos).
-rides(x,y) | -harley(y) | rough(x).  [clausify(1)].
-biker(x) | rides(x,f1(x,y)).  [clausify(2)].
-biker(x) | bmw(y) | harley(y).  [clausify(2)].
-rides(x,y) | -bmw(y) | yuppie(x).  [clausify(3)].
-yuppie(x) | lawyer(x).  [clausify(4)].
```

```
-nice(x) | -date(x,y) | -rough(y).  [clausify(5)].
nice(Mary).  [clausify(6)].
biker(John).  [clausify(6)].
-lawyer(John).  [deny(7)].
date(Mary,John).  [deny(7)].
end_of_list.

formulas(demodulators).
end_of_list.

============================== PREDICATE ELIMINATION
=================

Eliminating rides/2
8 -biker(x) | rides(x,f1(x,y)).  [clausify(2)].
9 -rides(x,y) | -harley(y) | rough(x).  [clausify(1)].
Derived: -biker(x) | -harley(f1(x,y)) | rough(x).
[resolve(8,b,9,a)].
10 -rides(x,y) | -bmw(y) | yuppie(x).  [clausify(3)].
Derived: -bmw(f1(x,y)) | yuppie(x) | -biker(x).
[resolve(10,a,8,b)].

Eliminating biker/1
11 biker(John).  [clausify(6)].
12 -biker(x) | bmw(y) | harley(y).  [clausify(2)].
Derived: bmw(x) | harley(x).  [resolve(11,a,12,a)].
13 -biker(x) | -harley(f1(x,y)) | rough(x).  [resolve(8,b,9,a)].
Derived: -harley(f1(John,x)) | rough(John).  [resolve(13,a,11,a)].
14 -bmw(f1(x,y)) | yuppie(x) | -biker(x).  [resolve(10,a,8,b)].
Derived: -bmw(f1(John,x)) | yuppie(John).  [resolve(14,c,11,a)].

Eliminating yuppie/1
15 -bmw(f1(John,x)) | yuppie(John).  [resolve(14,c,11,a)].
16 -yuppie(x) | lawyer(x).  [clausify(4)].
Derived: -bmw(f1(John,x)) | lawyer(John).  [resolve(15,b,16,a)].

Eliminating nice/1
17 nice(Mary).  [clausify(6)].
18 -nice(x) | -date(x,y) | -rough(y).  [clausify(5)].
Derived: -date(Mary,x) | -rough(x).  [resolve(17,a,18,a)].

Eliminating lawyer/1
19 -bmw(f1(John,x)) | lawyer(John).  [resolve(15,b,16,a)].
20 -lawyer(John).  [deny(7)].
Derived: -bmw(f1(John,x)).  [resolve(19,b,20,a)].

Eliminating date/2
21 -date(Mary,x) | -rough(x).  [resolve(17,a,18,a)].
```

```
22 date(Mary,John).  [deny(7)].
Derived: -rough(John).  [resolve(21,a,22,a)].

Eliminating bmw/1
23 -bmw(f1(John,x)).  [resolve(19,b,20,a)].
24 bmw(x) | harley(x).  [resolve(11,a,12,a)].
Derived: harley(f1(John,x)).  [resolve(23,a,24,a)].

Eliminating harley/1
25 harley(f1(John,x)).  [resolve(23,a,24,a)].
26 -harley(f1(John,x)) | rough(John).  [resolve(13,a,11,a)].
Derived: rough(John).  [resolve(25,a,26,a)].

Eliminating rough/1
27 rough(John).  [resolve(25,a,26,a)].
28 -rough(John).  [resolve(21,a,22,a)].
Derived: $F.  [resolve(27,a,28,a)].

============================== end predicate elimination
=============

Auto_denials:  (no changes).

Term ordering decisions:
Predicate symbol precedence:  predicate_order([ ]).
Function symbol precedence:  function_order([ ]).
After inverse_order:  (no changes).
Unfolding symbols: (none).

Auto_inference settings:
  % set(neg_binary_resolution).  % (HNE depth_diff=0)
  % clear(ordered_res).  % (HNE depth_diff=0)
  % set(ur_resolution).  % (HNE depth_diff=0)
    % set(ur_resolution) -> set(pos_ur_resolution).
    % set(ur_resolution) -> set(neg_ur_resolution).

Auto_process settings:  (no changes).

============================== PROOF
=================================

% Proof 1 at 0.01 (+ 0.01) seconds.
% Length of proof is 29.
% Level of proof is 8.
% Maximum clause weight is 0.
% Given clauses 0.
```

```
1 rides(x,y) & harley(y) -> rough(x) # label(non_clause).
[assumption].
2 biker(x) -> (exists y rides(x,y)) & (bmw(y) | harley(y)) #
label(non_clause).  [assumption].
3 rides(x,y) & bmw(y) -> yuppie(x) # label(non_clause).
[assumption].
4 yuppie(x) -> lawyer(x) # label(non_clause).  [assumption].
5 nice(x) -> -(date(x,y) & rough(y)) # label(non_clause).
[assumption].
6 nice(Mary) & biker(John) # label(non_clause).  [assumption].
7 -lawyer(John) -> -date(Mary,John) # label(non_clause) #
label(goal).  [goal].
8 -biker(x) | rides(x,f1(x,y)).  [clausify(2)].
9 -rides(x,y) | -harley(y) | rough(x).  [clausify(1)].
10 -rides(x,y) | -bmw(y) | yuppie(x).  [clausify(3)].
11 biker(John).  [clausify(6)].
12 -biker(x) | bmw(y) | harley(y).  [clausify(2)].
13 -biker(x) | -harley(f1(x,y)) | rough(x).  [resolve(8,b,9,a)].
14 -bmw(f1(x,y)) | yuppie(x) | -biker(x).  [resolve(10,a,8,b)].
15 -bmw(f1(John,x)) | yuppie(John).  [resolve(14,c,11,a)].
16 -yuppie(x) | lawyer(x).  [clausify(4)].
17 nice(Mary).  [clausify(6)].
18 -nice(x) | -date(x,y) | -rough(y).  [clausify(5)].
19 -bmw(f1(John,x)) | lawyer(John).  [resolve(15,b,16,a)].
20 -lawyer(John).  [deny(7)].
21 -date(Mary,x) | -rough(x).  [resolve(17,a,18,a)].
22 date(Mary,John).  [deny(7)].
23 -bmw(f1(John,x)).  [resolve(19,b,20,a)].
24 bmw(x) | harley(x).  [resolve(11,a,12,a)].
25 harley(f1(John,x)).  [resolve(23,a,24,a)].
26 -harley(f1(John,x)) | rough(John).  [resolve(13,a,11,a)].
27 rough(John).  [resolve(25,a,26,a)].
28 -rough(John).  [resolve(21,a,22,a)].
29 $F.  [resolve(27,a,28,a)].

============================== end of proof
==========================

============================== STATISTICS
============================

Given=0. Generated=1. Kept=0. proofs=1.
Usable=0. Sos=0. Demods=0. Limbo=0, Disabled=22. Hints=0.
Weight_deleted=0. Literals_deleted=0.
Forward_subsumed=0. Back_subsumed=0.
Sos_limit_deleted=0. Sos_displaced=0. Sos_removed=0.
New_demodulators=0 (0 lex), Back_demodulated=0. Back_unit_deleted=0.
Demod_attempts=0. Demod_rewrites=0.
```

```
Res_instance_prunes=0. Para_instance_prunes=0.
Basic_paramod_prunes=0.
Nonunit_fsub_feature_tests=0. Nonunit_bsub_feature_tests=0.
Megabytes=0.02.
User_CPU=0.01, System_CPU=0.01, Wall_clock=0.

============================== end of statistics
=====================

============================== end of search
=========================

THEOREM PROVED

Exiting with 1 proof.

Process 27908 exit (max_proofs) Mon Nov  9 21:45:27 2020
```

Conclusion:

From lines 27,28, we can conclude that, if John is not a lawyer, Mary does not date him.

Extra-credit: (**30 points)** Use Prover9 to automatically perform the refutation of the following:

The Pigs and Balloons Puzzle

(1) All, who neither dance on tight ropes nor eat penny-buns, are old.
(2) Pigs, that are liable to giddiness, are treated with respect.
(3) A wise balloonist takes an umbrella with him.
(4) No one ought to lunch in public who looks ridiculous and eats penny-buns.
(5) Young creatures, who go up in balloons, are liable to giddiness.
(6) Fat creatures, who look ridiculous, may lunch in public, provided that they do not dance on tight ropes.
(7) No wise creatures dance on tight ropes, if liable to giddiness.
(8) A pig looks ridiculous, carrying an umbrella.
(9) All, who do not dance on tight ropes, and who are treated with respect are fat.
Show that no wise young pigs go up in balloons.
-Lewis Carroll, Symbolic Logic,

Submit a report with three parts:

I. Assumptions and goal;

II. The input and output of prover9 (The input of prover 9 should be in plain text) III. Conclusion