CS 6364 HW 3

Rohith Peddi, RXP190007

November 11, 2020

1 INFERENCE IN PROPOSITIONAL LOGIC

1.1

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.

SOLUTION

Given sentences in the KB :

$(X \vee Y) \wedge (\neg X \vee Z) \wedge (\neg Y \vee W) \wedge (\neg Z \vee G) \wedge (\neg W \vee G)$ and $\alpha = G \implies \neg\alpha = \neg G$

As the current KB and query is in CNF.
We proceed by considering the clauses in the knowledge base and query:

1. $(X \vee Y)$
2. $(\neg X \vee Z)$
3. $(\neg Y \vee W)$
4. $(\neg Z \vee G)$
5. $(\neg W \vee G)$
6. $\neg G$

Applying the RESOLUTION rule:

1. $(C_5 : (\neg W \vee G), C_3 : (\neg Y \vee W)) \implies X_1 : (G \vee \neg Y)$
2. $(X_1 : (G \vee \neg Y), C_1 : (X \vee Y)) \implies X_2 : (G \vee X)$
3. $(X_2 : (G \vee X), C_2 : (\neg X \vee X) \implies X_3 : (G \vee Z))$
4. $(X_3 : (G \vee Z), C_4 : (\neg Z \vee G)) \implies X_4 : (G)$
5. $(X_4 : G, C_6 : \neg G) \implies NIL$

Thus α can be inferred from the KB.

1.2

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

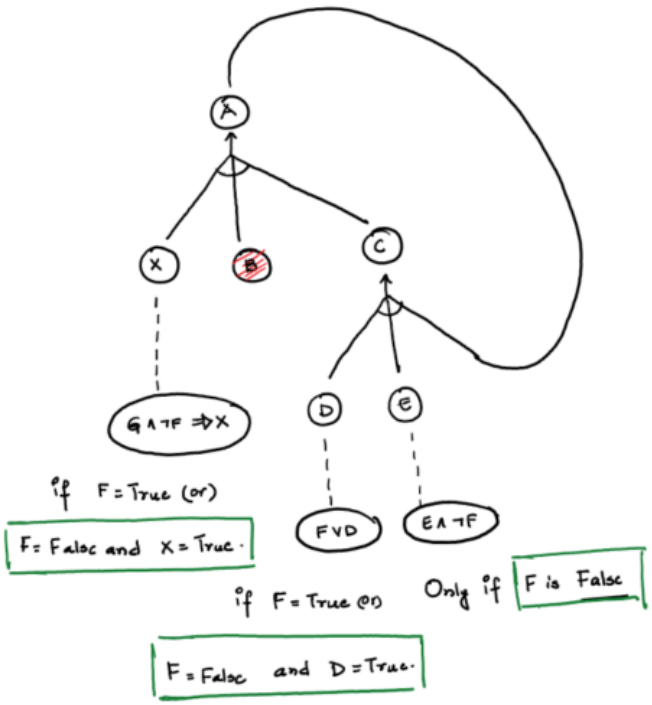
(15 points) Use backward-chaining inference to prove the query A.

SOLUTION

Observation: Not all the sentences are in the KB are Horn Clauses.

1. $S_1 : X \wedge B \wedge C \implies A$ [HORN CLUASE]
2. $S_2 : A \wedge D \wedge E \implies C$ [HORN CLAUSE]
3. $S_3 : B$ [HORN CLUASE]
4. $S_4 : E \wedge \neg F$ [Can be considered as two separate clauses where each is HORN CLAUSE, but here it is not a HORN CLAUSE]
5. $S_5 : F \vee D$ [Not a HORN CLAUSE]
6. $S_6 : G \wedge \neg F \implies X$ [Not a HORN CLAUSE]
7. $S_7 : G$ [HORN CLAUSE]

Considering AND-OR graph for HORN CLAUSES gives the following diagram and for others we build truth tables.



Considering truth table for the clauses S_4, S_5, S_6

F	D	$S_5 : F \vee D$	$S_4 : E \wedge \neg F$	$S_6 : G \wedge \neg F$
F	F	F	T (if E is True)	T (if X is True, As G = True)
F	T	**T**	**T (if E is True)**	**T (if X is True, As G = True)**
T	F	T	F	T (regardless of X)
T	T	T	F	T (regardless of X)

BACKWARD CHAINING PROCESS:

Step-1:

Suppose A is True

$$S_1 \implies X =_{mustbe} True, B =_{mustbe} True; C =_{mustbe} True$$

Step-2:

From S_3 we have a proof that $B = True$

Step-3:

Suppose C is True

$$S_2 \implies D =_{mustbe} True; E =_{mustbe} True$$

Step-4:

From S_7 we have a proof that $G = True$

Step-5:

If D is True, regardless of F, given S_5 we know that $F \vee D$ is True

Step-6:

If E is True and F is False given S_4 we know that $E \wedge \neg F$ is True

Step-7:

As we know, G is True, If F is False and X is True, given S_6 we know that $G \wedge \neg F \implies X$ is True.

Step-8:

From S_3 we have a proof that B is True.

Therefore our assumption that A is True was correct.

1.3

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)

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)

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

Considering sentences:

1. $S_1 : (X \vee Y) \wedge (\neg Z \vee \neg W \vee Q)$
2. $S_2 : X \vee Y$

Proof

Consider $A = (X \vee Y), B = (\neg Z \vee \neg W \vee Q)$
Now rewriting sentences using the substitutions gives us :

1. $S_1 : (A \wedge B)$
2. $S_2 : A$

Sentence S_1 is more restrictive than sentence S_2. So, $S_1 \models S_2$

Can also be verified by considering the validity of $S_1 \implies S_2$

Step-1: Aplpying Implication Elimination

$$(A \wedge B) \implies A$$

Step-2: Aplpying De Morgan's rule by pushing $\neg$ inwards

$$\neg(A \wedge B) \vee A$$

Step-3: Applying distributive law

$$(\neg A \vee \neg B) \vee A$$

Step-4: Applying Identity Law

$$T \vee \neg B$$

Step-6: Final result [VALIDITY!!]

$$T$$

2. Considering the sentences

1. $S_1 : (X \vee Y) \wedge (\neg Z \vee \neg W \vee Q)$
2. $S_3 : (X \vee Y \vee Z) \wedge ((Y \wedge Z \wedge W) \implies Q)$

Proof

$S_3 : (X \vee Y \vee Z) \wedge ((Y \wedge Z \wedge W) \implies Q)$ can be simplified to

$S_3 : (X \vee Y \vee Z) \wedge (\neg Y \vee \neg Z \vee \neg W \vee Q)$ [implication elimination and De Morgan's rules]

Considering the following substitutions $A = (X \vee Y)$ and $B = (\neg Z \vee \neg W \vee Q)$

1. $S_1 : A \wedge B$
2. $S_2 : (A \vee Z) \wedge (B \vee Y)$

$$S_1 \models S_2$$

Step-1:

$$(A \wedge B) \models (A \vee Z) \wedge (Y \vee B)$$

Step-2: Applying the rule Distributive rule

$$(A \wedge B) \models [(A \vee Z) \wedge Y] \vee [(A \vee Z) \wedge B]$$

Step-3: Applying the rule Distributive rule

$$(A \wedge B) \models [(A \vee Z) \wedge Y] \vee [(A \wedge B) \vee (Z \wedge B)]$$

Step-4: It is of the form :

$(A \wedge B) \models (A \wedge B) \vee [(A \vee Z) \wedge Y] \vee (Z \wedge B)$

Step-5:

As, $X \models X \vee Y$ in the above equation $(A \wedge B) \models (A \wedge B) \vee \mathbf{(L)}$ where $L = [(A \vee Z) \wedge Y] \vee (Z \wedge B)$

Step-6:

In all the models where $A \wedge B$ is T, $(A \wedge B) \vee [(A \vee Z) \wedge Y] \vee (Z \wedge B)$ shall also be T

Step-7: Final Conclusion

$$\implies S_1 \models S_3$$

3. Considering the sentences

1. $S_1 : (X \vee Y) \wedge (\neg Z \vee \neg W \vee Q)$
2. $S_4 : (X \vee Y) \wedge (\neg W \vee Q)$

Proof 1:

Considering the following substitution $A = (X \vee Y), B = (\neg W \vee Q)$ Rewriting the sentences using the substitution gives us:

1. $S_1 : A \wedge (\neg Z \vee B)$
2. $S_4 : (A \wedge B)$

$$S_1 \models S_2$$

Step-1

$$(A \wedge (\neg Z \vee B)) \models (A \wedge B)$$

Step-2: Applying distributive law

$$(A \wedge \neg Z) \vee (A \wedge B) \models (A \wedge B)$$

Step-3: Applying distributive law $(A \wedge \neg Z) \vee (A \wedge B)$ shall be true in more models than $(A \wedge B)$

$$\implies S_1 \not\models S_2$$

It can be proved by considering the **validity of** $S_1 \models S_4$

Step-4: For simplification, Consider the substitution $A \wedge B = P$

$$\implies [(A \wedge \neg Z) \vee (P) \implies P]$$

Step-5: Applying implication elimination

$$\neg((A \wedge \neg Z) \vee (P)) \vee P$$

Step-5: Moving $\neg$ inwards

$$(\neg(A \wedge \neg Z) \wedge \neg P) \vee P$$

Step-6: Moving $\neg$ inwards

$$((\neg A \vee Z) \wedge \neg P) \vee P$$

Step-7: Applying Distributive law

$$((\neg A \vee Z) \vee P) \wedge (\neg P \vee P)$$

Step-8: Applying Identity law

$$(\neg A \vee Z \vee P) \wedge (T)$$

Step-9: As $T \wedge X = X$ applying the same gives

$$\neg A \vee Z \vee P$$

Step-10: Replacing P with $A \wedge B$

$$Z \vee [\neg A \vee (A \wedge B)]$$

Step-11: Applying the Distributive law

$$Z \vee [(\neg A \vee A) \wedge (\neg A \vee B)]$$

Step-12: Applying the identity law

$$Z \vee [T \wedge (\neg A \vee B)]$$

Step-13: As $T \wedge X = X$ applying the same gives

$$Z \vee \neg A \vee B$$

Step-14: As A, B contains all different literals, above clause can never be VALID!!. It can be satisfiable.

$$S_1 \not\models S_4$$

1.4

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)**
Sentence 2: (Tall $\vee$ Gorgeous) $\vee \neg$(Tall $\Rightarrow$ Gorgeous)) **(5 points)**

SOLUTION

Considering

$$S_1 : [(Smart \vee Beautiful) \implies (Interesting \vee Boring)] \iff [(Smart \implies Interesting) \vee (Beautiful \implies Boring)]$$

Easiest way to check whether the sentence is valid or satisfiable or neither is to construct a truth table. Considering the intermediate expressions as follows:

$$S_{11} : (Smart \vee Beautiful)$$
$$S_{12} : (Interesting \vee Boring)$$
$$S_{13} : (Smart \implies Interesting)$$
$$S_{14} : (Beautiful \implies Boring)$$
$$S_{15} : (Smart \vee Beautiful) \implies (Interesting \vee Boring)$$
$$S_{16} : (Smart \implies Interesting) \vee (Beautiful \implies Boring)$$

Beautiful	Boring	Interesting	Smart	S_{11}	S_{12}	S_{15}	S_{13}	S_{14}	S_{16}	S_1
F	F	F	F	F	F	T	T	T	T	T
F	F	F	T	T	F	F	F	T	T	F
F	F	T	F	F	T	T	T	T	T	T
F	F	T	T	T	T	T	T	T	T	T
F	T	F	F	F	T	T	T	T	T	T
F	T	F	T	T	T	T	F	T	T	T
F	T	T	F	F	T	T	T	T	T	T
F	T	T	T	T	T	T	T	T	T	T
T	F	F	F	T	F	F	T	F	T	F
T	F	F	T	T	F	F	F	F	F	T
T	F	T	F	T	T	T	T	F	T	T
T	F	T	T	T	T	T	T	F	T	T
T	T	F	F	T	T	T	T	T	T	T
T	T	F	T	T	T	T	F	T	T	T
T	T	T	F	T	T	T	T	T	T	T
T	T	T	T	T	T	T	T	T	T	T

CONCLUSION: **It is not valid but it is satisfiable.**

Considering

$$S_2 : (Tall \vee Gorgeous) \vee \neg(Tall \implies Gorgeous)$$

Let us simplify the above expression.

Step-1: Implication elimination

$$S_2 : (Tall \vee Gorgeous) \vee \neg(\neg Tall \vee Gorgeous)$$

Step-2: Bring $\neg$ inwards [Applying De Morgan's rules]

$$S_2 : (Tall \vee Gorgeous) \vee (Tall \wedge \neg Gorgeous)$$

Step-3: Distributing $\vee$ over $\wedge$

$$S_2 : [(Tall \vee Gorgeous) \vee Tall] \wedge [(Tall \vee Gorgeous) \vee \neg Gorgeous)]$$

Step-4: Applying commutativity property

$$S_2 : [Gorgeous \vee (Tall \vee Tall)] \wedge [Tall \vee (Gorgeous \vee \neg Gorgeous)]$$

Step-5: Applying identity rules

$$S_2 : [Gorgeous \vee Tall] \wedge [Tall \vee (T)]$$

Step-6: Applying identity rules

$$S_2 : [Gorgeous \vee Tall] \wedge \mathbf{T}$$

Step-7: Applying identity rules

$$S_2 : Gorgeous \vee Tall$$

Not satisfiable when (Gorgeous = F and Tall = F)

CONCLUSION: **It is not valid but it is satisfiable**

2 LOGIC REPRESENTATIONS

2.1

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?

i. $(R \wedge E) \Leftrightarrow C$
ii. $R \Rightarrow (E \Leftrightarrow C)$
iii. $R \Rightarrow ((C \Rightarrow E) \vee \neg E)$

Explain why. **(15 points)**

SOLUTION

Given: A person who is a radical is electable if he/she is conservative, but otherwise is not electable.

2.1.1

S1: $(R \wedge E) \iff C$

According to sentence 1, the **bi implication** asserts

1. Radical and Electable persons are conservatives
2. Conservatives are radical and electable

But our given sentence does not say anything about the point 2 i,e all conservatives being radical and electable.

So, this is **not a correct propositional logic representation** of the sentence.

2.1.2

S2: $R \implies (E \iff C)$

This sentence asserts :
If a person is radical then he/she is electable if and only if he/she is conservative.
This seems to be perfectly inline with our sentence asserting

1. Radical person is electable if he/she is conservative.
2. Radical person is not electable if he/she is not conservative.

Yes, it is a correct representation of the sentence

2.1.3

S3: $R \implies ((C \implies E) \vee \neg E)$

Simplifying the above sentence gives us :

Step-1: Eliminating implications

$$S_3 : \neg R \vee ((\neg C \vee E) \vee \neg E$$

Step-2: Moving $\neg$ inwards

$$S_3 : \neg R \vee \neg C \vee E \vee \neg E$$

Step-3: Applying commutativity property

$$S_3 : \neg R \vee \neg C \vee (E \vee \neg E)$$

Step-4: Applying identity rules

$$S_3 : \neg R \vee \neg C \vee \mathbf{T}$$

FINAL EXPRESSION : **T**

Given sentence is true for all the models.

So, it is **not a correct propositional logic representation** of the sentence

2.2

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)}

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

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

SOLUTION:

2.2.1

Finding the most general unifier for $X : P(x), Y : P(A)$

X.args	x
Y.args	A

$$UNIFY(x, A) \implies \theta = \{x/A\}$$

$$MGU : \theta = \{x/A\}$$

$$MGCS = P(A)$$

2.2.2

Finding the most general unifier for $X : P[f(x), y, g(y)], Y : P[f(x), z, g(x)]$

X.args	f(x)	y	g(y)
Y.args	f(x)	z	g(x)

$$UNIFY(f(x), f(x)) \implies \theta = \{\}$$

$$UNIFY(y, z) \implies \theta = \{y/z\}$$

$$UNIFY(g(y), g(x)) \implies UNIFY(y, x, \theta)$$

$$UNIFY(z, x, \theta) \implies \theta = \{y/x, z/x\}$$

$$MGU : \theta = \{y/x, z/x\}$$

$$MGCS : P[f(x), x, g(x)]$$

Finding the most general unifier for

$$X : P[f(x), z, g(x)], Y : P[f(x), y, g(y)]$$

X.args	f(x)	z	g(x)
Y.args	f(x)	y	g(y)

$$UNIFY(f(x), f(x)) \implies \theta = \{\}$$

$$UNIFY(z, y) \implies \theta = \{z/y\}$$

$$UNIFY(g(x), g(y)) \implies \theta = \{z/y, x/y\}$$

$$MGU : \theta = \{z/y, x/y\}$$

$$MGCS : P[f(y), y, g(y)]$$

[MATCHED!!!]

2.2.3

Finding the most general unifier for $X : P[f(x, g(A, y)), g(A, y)], Y : P[f(x, z), z]$

X.args	f(x, g(A,y))	g(A,y)
Y.args	f(x, z)	z

First UNIFY results in Unification of list of arguments of first function.

$$UNIFY(f(x, g(A, y)), f(x, z)) \implies UNIFY([x, g(A, y)], [x, z])$$

$$UNIFY(x, x)) \implies \theta = \{\}$$

$$UNIFY(g(A, y), z) \implies \theta = \{z/g(A, y)\}$$

From first list

$$UNIFY(g(A, y), z) \implies \theta = \{z/g(A, y)\}$$

$$MGU : \theta = \{z/g(A, y)\}$$

$$MGCS : P[f(x, g(A, y)), g(A, y)]$$

3 FIRST ORDER LOGIC REPRESENTATION

Write in FOL the following statements by defining first your vocabulary (i.e. predicates, constants, variables, functions, etc):

3.1

(2 points) Some leaves turn red each Fall.

SOLUTION:

VOCABULARY	
Leaf(x)	x is a leaf
Fall(y)	Fall of year y
turn_red(x,y)	x turns red in the Fall of year y

$$\forall x Fall(x) \implies (\exists y Leaf(y) \wedge turn\_red(y, x))$$

3.2

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

SOLUTION:

VOCABULARY	
Leaf(x,y)	x is a leaf of tree y
Tree(x)	x is a tree
lose(x,y,z)	x loses y at time z
winter_comes(t)	time t when winter comes

$$\forall t winter\_comes(t) \implies [\exists x \forall y Tree(x) \wedge Leaf(y, x) \implies lose(x, y, t)]$$

3.3

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

SOLUTION:

VOCABULARY	
Flowers(x)	[OBJECT] x is a flower
always_nice(x)	[PROPERTY] x is always nice
smell_lovely(x)	[PROPERTY] x smells lovely

$$\forall x Flower(x) \implies always\_nice(x) \wedge smell\_lovely(x)$$

3.4

(2 points) One flower does not bring Spring.

SOLUTION:

VOCABULARY	
Set(x)	[OBJECT] x is a set of flowers
brings(x, y)	[PREDICATE] x brings y
SPRING	[CONSTANT]
Count(x)	[Function] count of elements in x

$$\forall x Set(x) \wedge (Count(x) = 1) \implies \neg bring(x, SPRING)$$

3.5

(2 points) Every flower fades at some point.

SOLUTION:

VOCABULARY	
Flowers(x)	[OBJECT] x is a flower
fades(x, y)	[PREDICATE] x fades at time t

$$\forall x \exists t Flower(x) \wedge fades(x,t)$$

3.6

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

SOLUTION:

VOCABULARY	
Flower(x)	[OBJECT] x is a flower
left(x, y)	[PREDICATE] x is left in y
Vase	[CONSTANT]

$$\exists x[Flower(x) \wedge left(x, Vase) \wedge [\forall y Flower(y) \wedge left(y, vase) \implies (x = y)]]$$

3.7

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

SOLUTION:

VOCABULARY	
Person(x)	[OBJECT] x is a person
buys(x, y)	[PREDICATE] x buys y
Flower(y)	[OBJECT] y is a flower
Sensitive(x)	[PROPERTY] x is sensitive

$$\forall x, y Person(x) \wedge Flower(y) \wedge buys(x,y) \implies Sensitive(x)$$

3.8

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

SOLUTION:

VOCABULARY	
Poet(x)	[OBJECT] x is a poet
buys(x, y)	[PREDICATE] x buys y
Flower(y)	[OBJECT] y is a flower
Sensitive(x)	[PROPERTY] x is sensitive
Poem(z)	[OBJECT] z is a poem
Beautiful(x)	[PROPERTY] x is beautiful
write(x,y)	[PREDICATE] x writes y

$$\forall x[Poet(x) \implies Sensitive(x) \wedge [\forall y Flower(y) \implies \neg buys(x,y)] \wedge [\forall z Poem(z) \wedge write(x,z) \implies Beautiful(z)]]$$

Or in a simpler case can also be written as :

VOCABULARY	
Poet(x)	[OBJECT] x is a poet
buy_flowers(x)	[PROPERTY] x buys flowers
Sensitive(x)	[PROPERTY] x is sensitive
write_beautiful_poems(x)	[PROPERTY] x writes beautiful poems

$$\forall x Poet(x) \implies Sensitive(x) \wedge \neg buy\_flowers(x) \wedge write\_beautiful\_poems(x)$$

3.9

(2 points) No poet will kill an animal.

SOLUTION:

VOCABULARY	
Poet(x)	[OBJECT] x is a poet
Animal(y)	[OBJECT] y is an animal
kill(x, y)	[PREDICATE] x kills y

$$\forall x, y Poet(x) \wedge Animal(y) \implies \neg kill(x, y)$$

3.10

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

SOLUTION:

VOCABULARY	
Agent(x)	[OBJECT] x is an agent
Policy(y)	[OBJECT] y is a policy
sells(x, y, z)	[PREDICATE] x sells y to z
Insured(z)	[PROPERTY] z is insured
Person(z)	[OBJECT] z is a person

$$\exists x Agent(x) \wedge \forall y, z[Policy(y) \wedge sells(x, y, z) \implies Person(z) \wedge \neg Insured(z)]$$

3.11

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

SOLUTION:

VOCABULARY	
Barber(x)	[OBJECT] x is a barber
Man(y)	[OBJECT] y is a man
shave(x, y)	[PREDICATE] x shaves y

$$\exists x Barber(x) \wedge \forall y[Man(y) \wedge \neg shaves(y, y) \implies shaves(x, y)]$$

3.12

(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.

SOLUTION:

VOCABULARY	
Person(x)	[OBJECT] x is a person
born(x, y)	[OBJECT] x is born in country y
citizen(x, y, z)	[PREDICATE] x is a citizen of y by z
Parent(x,y)	[PREDICATE] x is a parent of y
Birth	[CONSTANT]
US	[CONSTANT]
Descent	[CONSTANT]

$$\forall x Person(x) \wedge \neg born(x, US) \wedge [\exists y Parent(y, x) \wedge citizen(y, US, Birth)] \implies citizen(x, US, Descent)$$

3.13

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

SOLUTION:

VOCABULARY	
Flower(x)	[OBJECT] x is a flower
in(x, y)	[PREDICATE] x is in y
smells_nice(x)	[PROPERTY] x smells nice
house	[CONSTANT]

$$\exists x Flower(x) \land smells\_nice(x) \land in(x, house)$$

3.14

(3 points) John bought only two flowers.

SOLUTION:

VOCABULARY	
Flower(x)	[OBJECT] x is a Flower
bought(x, y)	[PREDICATE] x bought y
John	[CONSTANT]

$$\exists x, y Flower(x) \land Flower(y) \land bought(John, x) \land bought(John, y) \land \neg(x = y) \land \forall z[Flower(x) \land bought(John, z) \implies (z = y) \lor (z = x)]$$

4 REFUTATION IN FIRST ORDER LOGIC

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.

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.

4.1

(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. $\forall x(\exists y Rides(x,y) \wedge Harley(y)) \implies Rough(x)$
2. $\forall x Biker(x) \implies \exists y Rides(x,y) \wedge (Harley(y) \vee BMW(y))$
3. $\forall x, y Rides(x,y) \wedge BMW(y) \implies Yuppie(x)$
4. $\forall x Yuppie(x) \implies Lawyer(x)$
5. $\forall x, y Nice(x) \wedge Rough(y) \implies \neg Date(x,y)$
6. $Nice(Mary) \wedge Biker(John)$
7. CONCLUSION : $\neg Lawyer(John) \implies \neg Date(Mary, John)$

4.2

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

SOLUTION

Conversion to clause form skolemizing as necessary invovles the 6 steps :

Considering $\forall x(\exists y Rides(x,y) \wedge Harley(y)) \implies Rough(x)$

Step-1 : Eliminate Implications
$\forall x \neg(\exists y Rides(x,y) \wedge Harley(y)) \vee Rough(x)$

Step-2 : Move $\neg$ inwards
$\forall x(\forall y \neg Rides(x,y) \vee \neg Harley(y)) \vee Rough(x)$

Step-3 : Standardize variables
$S_1 : \forall x_1, x_2 \neg Rides(x_1, x_2) \vee \neg Harley(x_2) \vee Rough(x_1)$

Step-4 : Skolemize
$S_1 : \forall x_1, x_2 \neg Rides(x_1, x_2) \vee \neg Harley(x_2) \vee Rough(x_1)$

Step-5 : Drop universal quantifiers
$S_1 : \neg Rides(x_1, x_2) \vee \neg Harley(x_2) \vee Rough(x_1)$

Step-6 : Distribute $\wedge$ over $\vee$
$S_1 : \neg Rides(x_1, x_2) \vee \neg Harley(x_2) \vee Rough(x_1)$

FINAL CNF :
$S_1 : [\neg Rides(x_1, x_2) \vee \neg Harley(x_2) \vee Rough(x_1)]$

Considering $S_2 : \forall x Biker(x) \implies \exists y Rides(x,y) \wedge (Harley(y) \vee BMW(y))$

Step-1 : Eliminate Implications

$S_2 : \forall x \neg Biker(x) \vee \exists y Rides(x, y) \wedge (Harley(y) \vee BMW(y))$

Step-2 : Move $\neg$ inwards
$S_2 : \forall x \neg Biker(x) \vee \exists y Rides(x, y) \wedge (Harley(y) \vee BMW(y))$

Step-3 : Standardize variables
$S_2 : \forall x_3 \neg Biker(x_3) \vee \exists x_4 Rides(x_3, x_4) \wedge (Harley(x_4) \vee BMW(x_4))$

Step-4 : Skolemize
$S_2 : \forall x_3 \neg Biker(x_3) \vee (Rides(x_3, f(x_3)) \wedge (Harley(f(x_3)) \vee BMW(f(x_3))))$

Step-5 : Drop universal quantifiers
$S_2 : \neg Biker(x_3) \vee (Rides(x_3, f(x_3)) \wedge (Harley(f(x_3)) \vee BMW(f(x_3))))$

Step-6 : Distribute
$S_2 : \neg Biker(x_3) \vee (Rides(x_3, f(x_3)) \wedge (Harley(f(x_3)) \vee BMW(f(x_3))))$
$S_2 : [\neg Biker(x_3) \vee Rides(x_3, f(x_3))] \wedge [\neg Biker(x_3) \vee (Harley(f(x_3)) \vee BMW(f(x_3)))]$

FINAL CNF : Produces two clauses
$S_{21} : [\neg Biker(x_3) \vee Rides(x_3, f(x_3))]$
$S_{22} : [\neg Biker(x_3) \vee Harley(f(x_3)) \vee BMW(f(x_3))]$

Considering $S_3 : \forall x, y Rides(x, y) \wedge BMW(y) \implies Yuppie(x)$

Step-1 : Eliminate Implications
$S_3 : \forall x, y \neg(Rides(x, y) \wedge BMW(y)) \vee Yuppie(x)$

Step-2 : Move $\neg$ inwards
$S_3 : \forall x, y \neg Rides(x, y) \vee \neg BMW(y) \vee Yuppie(x)$

Step-3 : Standardize variables
$S_3 : \forall x_5, x_6 \neg Rides(x_5, x_6) \vee \neg BMW(x_6) \vee Yuppie(x_5)$

Step-4 : Skolemize
$S_3 : \forall x_5, x_6 \neg Rides(x_5, x_6) \vee \neg BMW(x_6) \vee Yuppie(x_5)$

Step-5 : Drop universal quantifiers
$S_3 : \neg Rides(x_5, x_6) \vee \neg BMW(x_6) \vee Yuppie(x_5)$

Step-6 : Distribute $\wedge$ over $\vee$
$S_3 : \neg Rides(x_5, x_6) \vee \neg BMW(x_6) \vee Yuppie(x_5)$

FINAL CNF :
$S_3 : \neg Rides(x_5, x_6) \vee \neg BMW(x_6) \vee Yuppie(x_5)$

Considering $S_4 : \forall x Yuppie(x) \implies Lawyer(x)$

Step-1 : Eliminate Implications
$S_4 : \forall x \neg Yuppie(x) \vee Lawyer(x)$

Step-2 : Move $\neg$ inwards
$S_4 : \forall x \neg Yuppie(x) \vee Lawyer(x)$

Step-3 : Standardize variables
$S_4 : \forall x_7 \neg Yuppie(x_7) \vee Lawyer(x_7)$

Step-4 : Skolemize
$S_4 : \forall x_7 \neg Yuppie(x_7) \vee Lawyer(x_7)$

Step-5 : Drop universal quantifiers
$S_4 : \neg Yuppie(x_7) \vee Lawyer(x_7)$

Step-6 : Distribute $\wedge$ over $\vee$
$S_4 : \neg Yuppie(x_7) \vee Lawyer(x_7)$

FINAL CNF :
$S_4 : \neg Yuppie(x_7) \vee Lawyer(x_7)$

Considering $S_5 : \forall x, y Nice(x) \wedge Rough(y) \implies \neg Date(x, y)$

Step-1 : Eliminate Implications
$S_5 : \forall x, y \neg[Nice(x) \wedge Rough(y)] \vee \neg Date(x, y)$

Step-2 : Move $\neg$ inwards
$S_5 : \forall x, y \neg Nice(x) \vee \neg Rough(y) \vee \neg Date(x, y)$

Step-3 : Standardize variables
$S_5 : \forall x_8, x_9 \neg Nice(x_8) \vee \neg Rough(x_9) \vee \neg Date(x_8, x_9)$

Step-4 : Skolemize
$S_5 : \forall x_8, x_9 \neg Nice(x_8) \vee \neg Rough(x_9) \vee \neg Date(x_8, x_9)$

Step-5 : Drop universal quantifiers
$S_5 : \neg Nice(x_8) \vee \neg Rough(x_9) \vee \neg Date(x_8, x_9)$

Step-6 : Distribute $\wedge$ over $\vee$
$S_5 : \neg Nice(x_8) \vee \neg Rough(x_9) \vee \neg Date(x_8, x_9)$

FINAL CNF :
$S_5 : \neg Nice(x_8) \vee \neg Rough(x_9) \vee \neg Date(x_8, x_9)$

Considering $S_6 : Nice(Mary) \wedge Biker(John)$

FINAL CNF Produces two clauses
$S_{61} : Nice(Mary)$
$S_{62} : Biker(John)$

Considering the conclusion $\alpha : \neg Lawyer(John) \implies \neg Date(Mary, John)$
Adding $\neg\alpha$ to the knowledge base inorder to prove resolution refutation gives us.

Step-1 : Eliminate Implications
$\neg\alpha : \neg[\neg\neg Lawyer(John) \vee \neg Date(Mary, John)]$

Step-2 : Move $\neg$ inwards
$\neg\alpha : \neg Lawyer(John) \wedge Date(Mary, John)$

Step-3 : Standardize variables
$\neg\alpha : \neg Lawyer(John) \wedge Date(Mary, John)$

Step-4 : Skolemize
$\neg\alpha : \neg Lawyer(John) \wedge Date(Mary, John)$

Step-5 : Drop universal quantifiers
$\neg\alpha : \neg Lawyer(John) \wedge Date(Mary, John)$

Step-6 : Distribute $\wedge$ over $\vee$
$\neg\alpha : \neg Lawyer(John) \wedge Date(Mary, John)$

FINAL CNF : Produces two clauses
$\neg\alpha : \neg Lawyer(John) \wedge Date(Mary, John)$
$S_{71} : \neg Lawyer(John)$
$S_{72} : Date(Mary, John)$

4.3

(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.

SOLUTION

Following are the clauses present in the Knowledge Base after converting to clause form.

1. $S_1 : \neg Rides(x_1, x_2) \vee \neg Harley(x_2) \vee Rough(x_1)$
2. $S_{21} : \neg Biker(x_3) \vee Rides(x_3, f(x_3))$
3. $S_{22} : \neg Biker(x_3) \vee Harley(f(x_3)) \vee BMW(f(x_3))$
4. $S_3 : \neg Rides(x_5, x_6) \vee \neg BMW(x_6) \vee Yuppie(x_5)$
5. $S_4 : \neg Yuppie(x_7) \vee Lawyer(x_7)$
6. $S_5 : \neg Nice(x_8) \vee \neg Rough(x_9) \vee \neg Date(x_8, x_9)$
7. $S_{61} : Nice(Mary)$
8. $S_{62} : Biker(John)$
9. $S_{71} : \neg Lawyer(John)$
10. $S_{72} : Date(Mary, John)$

RESOLUTION PROCESS:

Step:1

$S_5 : \neg Nice(x_8) \vee \neg Rough(x_9) \vee \neg Date(x_8, x_9)$	$S_{61} : Nice(Mary)$
Predicate : Nice	Substitution : $x_8/Mary$
RESULT	$X_1 : \neg Rough(x_9) \vee \neg Date(Mary, x_9)$

Step:2

$X_1 : \neg Rough(x_9) \vee \neg Date(Mary, x_9)$	$S_{72} : Date(Mary, John)$
Predicate : Date	Substitution : $x_9/John$
RESULT	$X_2 : \neg Rough(John)$

Step:3

$S_4 : \neg Yuppie(x_7) \vee Lawyer(x_7)$	$S_{71} : \neg Lawyer(John)$
Predicate : Lawyer	$Substitution : x_7/John$
RESULT	$X_3 : \neg Yuppie(John)$

Step:4

$X_3 : \neg Yuppie(John)$	$S_3 : \neg Rides(x_5, x_6) \vee \neg BMW(x_6) \vee Yuppie(x_5)$
Predicate : Yuppie	$Substitution : x_5/John$
RESULT	$X_4 : \neg Rides(John, x_6) \vee \neg BMW(x_6)$

Step:5

$S_{22} : \neg Biker(x_3) \vee Harley(f(x_3)) \vee BMW(f(x_3))$	$S_{62} : Biker(John)$
Predicate : Biker	$Substitution : x_3/John$
RESULT	$X_5 : Harley(f(John)) \vee BMW(f(John))$

Step:6

$S_{21} : \neg Biker(x_3) \vee Rides(x_3, f(x_3))$	$S_{62} : Biker(John)$
Predicate : Biker	$Substitution : x_3/John$
RESULT	$X_6 : Rides(John, f(John))$

Step:7

$X_6 : Rides(John, f(John))$	$X_4 : \neg Rides(John, x_6) \vee \neg BMW(x_6)$
Predicate : Rides	$Substitution : x_6/f(John)$
RESULT	$X_7 : \neg BMW(f(John))$

Step:8

$X_7 : \neg BMW(f(John))$	$X_5 : Harley(f(John)) \vee BMW(f(John))$
Predicate : BMW	$Substitution : \{\}$
RESULT	$X_8 : Harley(f(John))$

Step:9

$X_8 : Harley(f(John))$	$S_1 : \neg Rides(x_1, x_2) \vee \neg Harley(x_2) \vee Rough(x_1)$
Predicate : Harley	$Substitution : x_2 : f(John)$
RESULT	$X_9 : \neg Rides(x_1, f(John)) \vee Rough(x_1)$

Step:10

$X_6 : Rides(John, f(John))$	$X_9 : \neg Rides(x_1, f(John)) \vee Rough(x_1)$
Predicate : Rides	$Substitution : x_1 : John$
RESULT	$X_{10} : Rough(John)$

Step:11

$X_{10} : Rough(John)$	$X_2 : \neg Rough(John)$
Predicate : Rides	$Substitution : \{\}$
RESULT	$X_{11} : NIL$

4.4

(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

4.4.1 ASSUMPTIONS AND GOAL:

Following are the assumptions made

1. $S_1 : \neg Rides(x_1, x_2) \vee \neg Harley(x_2) \vee Rough(x_1)$
2. $S_{21} : \neg Biker(x_3) \vee Rides(x_3, f(x_3))$
3. $S_{22} : \neg Biker(x_3) \vee Harley(f(x_3)) \vee BMW(f(x_3))$
4. $S_3 : \neg Rides(x_5, x_6) \vee \neg BMW(x_6) \vee Yuppie(x_5)$
5. $S_4 : \neg Yuppie(x_7) \vee Lawyer(x_7)$
6. $S_5 : \neg Nice(x_8) \vee \neg Rough(x_9) \vee \neg Date(x_8, x_9)$
7. $S_{61} : Nice(Mary)$
8. $S_{62} : Biker(John)$

Following are the goals considered

1. $S_7 : Lawyer(John) \vee -Date(Mary, John)$

4.4.2 INPUT AND OUTPUT OF PROVER9

INPUT

```
formulas(assumptions).   % synonym for formulas(sos).
  -Rides(x,y) | -Harley(y) | Rough(x).
  -Biker(x) | Rides(x,f(x)).
  -Biker(x) | Harley(f(x)) | BMW(f(x)).
  -Rides(x,y) | -BMW(y) | Yuppie(x).
  -Yuppie(x) | Lawyer(x).
  -Nice(x) | -Rough(y) | -Date(x,y).
  Nice(Mary).
  Biker(John).
end_of_list.

formulas(goals).         % to be negated and placed in the sos list
  Lawyer(John) | -Date(Mary, John).
end_of_list.
```

OUTPUT

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 12132 was started by ROHITH PEDDI on DESKTOP-T2S851L,
Mon Nov  9 22:03:58 2020
The command was "/cygdrive/c/Program Files (x86)/Prover9-Mace4/bin-win32/prover9".
============================== end of head ===========================

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

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

% -------- Comments from original proof --------
% Proof 1 at 0.01 (+ 0.06) seconds.
% Length of proof is 23.
% Level of proof is 6.
% Maximum clause weight is 0.
% Given clauses 0.

1 Lawyer(John) | -Date(Mary,John) # label(non_clause) # label(goal).  [goal].
2 -Biker(x) | Rides(x,f(x)).  [assumption].
3 -Rides(x,y) | -Harley(y) | Rough(x).  [assumption].
```

```
4 -Rides(x,y) | -BMW(y) | Yuppie(x).  [assumption].
5 Biker(John).  [assumption].
6 -Biker(x) | Harley(f(x)) | BMW(f(x)).  [assumption].
7 -Biker(x) | -Harley(f(x)) | Rough(x).  [resolve(2,b,3,a)].
8 -BMW(f(x)) | Yuppie(x) | -Biker(x).  [resolve(4,a,2,b)].
9 -BMW(f(John)) | Yuppie(John).  [resolve(8,c,5,a)].
10 -Yuppie(x) | Lawyer(x).  [assumption].
11 Nice(Mary).  [assumption].
12 -Nice(x) | -Rough(y) | -Date(x,y).  [assumption].
13 -BMW(f(John)) | Lawyer(John).  [resolve(9,b,10,a)].
14 -Lawyer(John).  [deny(1)].
15 -Rough(x) | -Date(Mary,x).  [resolve(11,a,12,a)].
16 Date(Mary,John).  [deny(1)].
17 -Harley(f(John)) | Rough(John).  [resolve(7,a,5,a)].
18 Harley(f(John)) | BMW(f(John)).  [resolve(5,a,6,a)].
19 Rough(John) | BMW(f(John)).  [resolve(17,a,18,a)].
20 -BMW(f(John)).  [resolve(13,b,14,a)].
21 Rough(John).  [resolve(19,b,20,a)].
22 -Rough(John).  [resolve(15,b,16,a)].
23 $F.  [resolve(21,a,22,a)].

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

4.4.3 CONCLUSION:

1. Prover9 proved the query using resolution technique
2. Knowledge base entails the given query

5 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

5.0.1 ASSUMPTIONS AND GOAL

Following are the assumptions considered :

1. $Dances\_on\_tightropes(x) \vee Eats\_pennybuns(x) \vee Old(x)$
2. $\neg Pig(x) \vee \neg Liable\_to\_giddiness(x) \vee Treated\_with\_respect(x)$
3. $\neg Wise(x) \vee \neg Balloonist(x) \vee Has\_umbrella(x)$
4. $\neg Looks\_ridiculous(x) \vee \neg Eats\_pennybuns(x) \vee \neg Eats\_lunch\_in\_public(x)$
5. $\neg Balloonist(x) \vee \neg Young(x) \vee Liable\_to\_giddiness(x)$
6. $\neg Fat(x) \vee \neg Looks\_ridiculous(x) \vee Dances\_on\_tightropes(x) \vee Eats\_lunch\_in\_public(x)$
7. $\neg Liable\_to\_giddiness(x) \vee \neg Wise(x) \vee \neg Dances\_on\_tightropes(x)$
8. $\neg Pig(x) \vee \neg Has\_umbrella(x) \vee Looks\_ridiculous(x)$
9. $Dances\_on\_tightropes(x) \vee \neg Treated\_with\_respect(x) \vee Fat(x)$
10. $Young(x) \vee Old(x)$
11. $\neg Young(x) \vee \neg Old(x)$

Following are the goals considered

1. $\neg Wise(x) \vee \neg Young(x) \vee \neg Pig(x) \vee \neg Balloonist(x)$

5.0.2 INPUT AND OUTPUT

INPUT

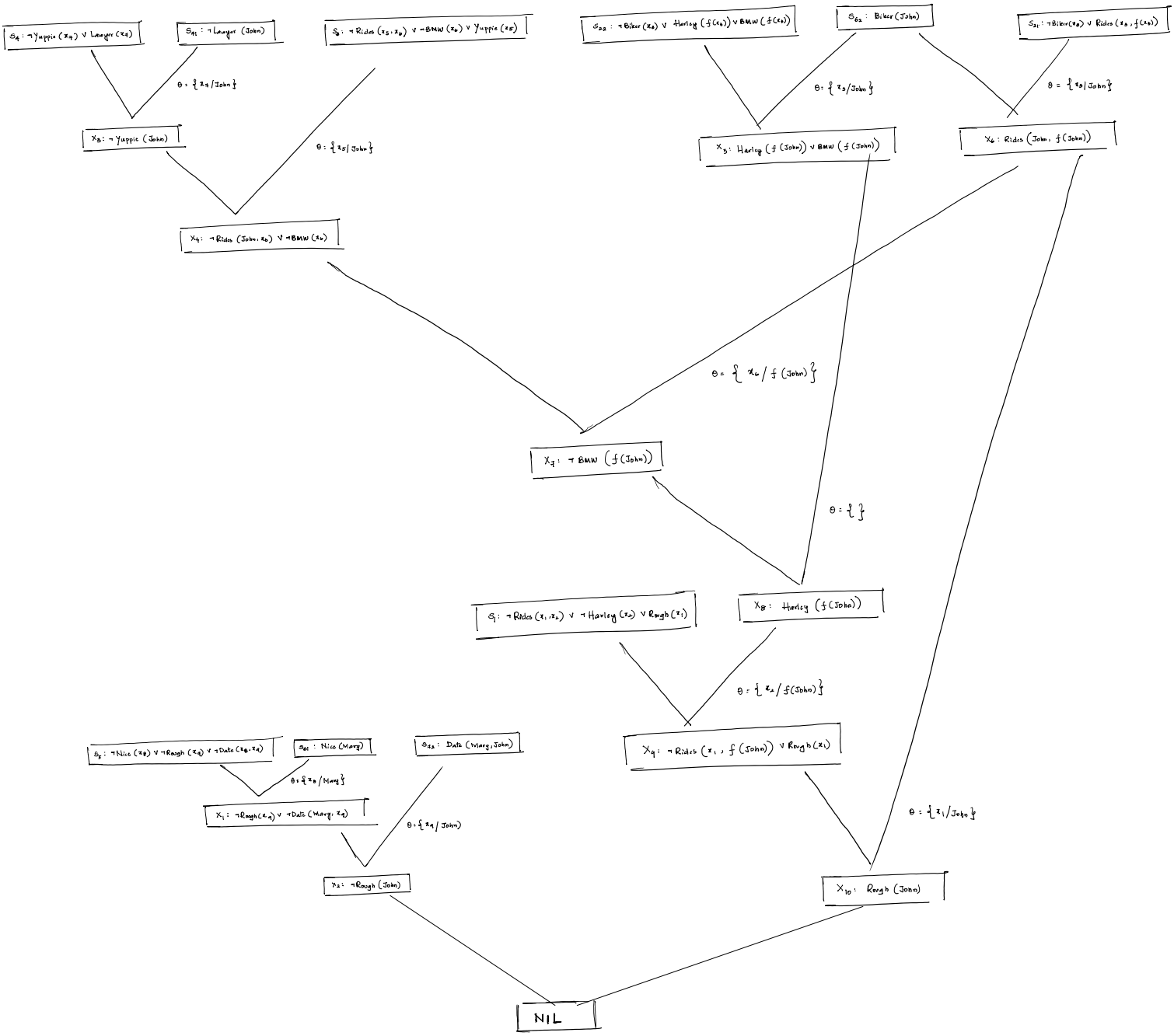
```
formulas(assumptions).   % synonym for formulas(sos).
  Dances_on_tightropes(x) | Eats_pennybuns(x) | Old(x).
  -Pig(x)|-Liable_to_giddiness(x) | Treated_with_respect(x).
  -Wise(x) | -Balloonist(x) | Has_umbrella(x).
  -Looks_ridiculous(x) | -Eats_pennybuns(x) | -Eats_lunch_in_public(x).
  -Balloonist(x) | -Young(x) | Liable_to_giddiness(x).
  -Fat(x) | -Looks_ridiculous(x) | Dances_on_tightropes(x) | Eats_lunch_in_public(x).
  -Liable_to_giddiness(x) | -Wise(x) | -Dances_on_tightropes(x).
  -Pig(x) | -Has_umbrella(x) | Looks_ridiculous(x).
  Dances_on_tightropes(x) | -Treated_with_respect(x) | Fat(x).
  Young(x) | Old(x).
  -Young(x) | -Old(x).
end_of_list.

formulas(goals).         % to be negated and placed in the sos list
  -Wise(x) | -Young(x) | -Pig(x) | -Balloonist(x).
end_of_list.
```

OUTPUT

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 23656 was started by ROHITH PEDDI on DESKTOP-T2S851L,
Mon Nov  9 22:14:08 2020
The command was "/cygdrive/c/Program Files (x86)/Prover9-Mace4/bin-win32/prover9".
============================== end of head ===========================

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

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

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.06) seconds.
% Length of proof is 42.
% Level of proof is 8.
% Maximum clause weight is 2.
% Given clauses 0.

1 -Wise(x) | -Young(x) | -Pig(x) | -Balloonist(x) # label(non_clause) # label(goal).  [goal].
2 -Liable_to_giddiness(x) | -Wise(x) | -Dances_on_tightropes(x).  [assumption].
3 Dances_on_tightropes(x) | Eats_pennybuns(x) | Old(x).  [assumption].
4 -Fat(x) | -Looks_ridiculous(x) | Dances_on_tightropes(x) | Eats_lunch_in_public(x).  [
    assumption].
5 Dances_on_tightropes(x) | -Treated_with_respect(x) | Fat(x).  [assumption].
6 Pig(c1).  [deny(1)].
7 -Pig(x) | -Liable_to_giddiness(x) | Treated_with_respect(x).  [assumption].
8 -Pig(x) | -Has_umbrella(x) | Looks_ridiculous(x).  [assumption].
9 Wise(c1).  [deny(1)].
10 -Wise(x) | -Balloonist(x) | Has_umbrella(x).  [assumption].
11 -Liable_to_giddiness(x) | -Wise(x) | Eats_pennybuns(x) | Old(x).  [resolve(2,c,3,a)].
12 -Liable_to_giddiness(x) | -Wise(x) | -Fat(x) | -Looks_ridiculous(x) | Eats_lunch_in_public(
    x).  [resolve(2,c,4,c)].
13 -Treated_with_respect(x) | Fat(x) | -Liable_to_giddiness(x) | -Wise(x).  [resolve(5,a,2,c)
    ].
14 -Has_umbrella(c1) | Looks_ridiculous(c1).  [resolve(6,a,8,a)].
15 -Looks_ridiculous(x) | -Eats_pennybuns(x) | -Eats_lunch_in_public(x).  [assumption].
16 -Liable_to_giddiness(c1) | -Fat(c1) | -Looks_ridiculous(c1) | Eats_lunch_in_public(c1).  [
    resolve(12,b,9,a)].
17 Balloonist(c1).  [deny(1)].
18 -Balloonist(x) | -Young(x) | Liable_to_giddiness(x).  [assumption].
19 -Balloonist(c1) | Has_umbrella(c1).  [resolve(9,a,10,a)].
20 -Young(x) | -Old(x).  [assumption].
21 Young(x) | Old(x).  [assumption].
22 Young(c1).  [deny(1)].
23 -Young(c1) | Liable_to_giddiness(c1).  [resolve(17,a,18,a)].
24 Liable_to_giddiness(c1) | Old(c1).  [resolve(23,a,21,a)].
25 -Liable_to_giddiness(c1) | Treated_with_respect(c1).  [resolve(6,a,7,a)].
26 -Liable_to_giddiness(c1) | Eats_pennybuns(c1) | Old(c1).  [resolve(11,b,9,a)].
27 -Treated_with_respect(c1) | Fat(c1) | -Liable_to_giddiness(c1).  [resolve(13,d,9,a)].
28 -Liable_to_giddiness(c1) | -Fat(c1) | Eats_lunch_in_public(c1) | -Has_umbrella(c1).  [
    resolve(16,c,14,b)].
30 Has_umbrella(c1).  [resolve(19,a,17,a)].
31 -Has_umbrella(c1) | -Eats_pennybuns(c1) | -Eats_lunch_in_public(c1).  [resolve(14,b,15,a)].
32 Old(c1) | -Fat(c1) | Eats_lunch_in_public(c1) | -Has_umbrella(c1).  [resolve(24,a,28,a)].
34 Old(c1) | -Treated_with_respect(c1) | Fat(c1).  [resolve(24,a,27,c)].
35 Old(c1) | Treated_with_respect(c1).  [resolve(24,a,25,a)].
38 -Eats_pennybuns(c1) | -Eats_lunch_in_public(c1).  [resolve(30,a,31,a)].
39 Old(c1) | Eats_pennybuns(c1) | Old(c1).  [resolve(24,a,26,a)].
40 Old(c1) | Fat(c1) | Old(c1).  [resolve(34,b,35,b)].
41 Old(c1) | -Fat(c1) | Eats_lunch_in_public(c1).  [resolve(32,d,30,a)].
44 Old(c1) | Old(c1) | Old(c1) | Eats_lunch_in_public(c1).  [resolve(40,b,41,b)].
45 -Eats_lunch_in_public(c1) | Old(c1) | Old(c1).  [resolve(38,a,39,b)].
47 -Old(c1).  [resolve(22,a,20,a)].
48 Old(c1) | Old(c1) | Old(c1) | Old(c1) | Old(c1).  [resolve(44,d,45,a)].
49 $F.  [copy(48),merge(b),merge(c),merge(d),merge(e),unit_del(a,47)].

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

5.0.3 CONCLUSION:

1. Prover9 proved the query using resolution technique
2. Knowledge base entails the given query


S4 : ¬Yuppie (x4) ∨ Lawyer (x4)
S41 : ¬Lawyer (John)
S5 : ¬Rides (x5, x5) ∨ ¬BMW (x5) ∨ Yuppie (x5)
S22 : ¬Biker (x2) ∨ Harley (f(x2)) ∨ BMW (f(x2))
S61 : Biker (John)
S21 : ¬Biker(x2) ∨ Rides (x2, f(x2))
θ = { x4 / John }
θ = { x5 / John }
θ = { x2 / John }
θ = { x2 / John }
X3 : ¬Yuppie (John)
X5 : Harley (f (John)) ∨ BMW (f (John))
X6 : Rides (John, f (John))
X4 : ¬Rides (John, x6) ∨ ¬BMW (x6)
θ = { x6 / f (John) }
X7 : ¬ BMW (f (John))
θ = { }
S1 : ¬Rides (x1, x2) ∨ ¬Harley (x2) ∨ Rough (x1)
X8 : Harley (f (John))
θ = { x2 / f (John) }
S8 : ¬Nice (x8) ∨ ¬Rough (x8) ∨ ¬Date (x8, x8)
S81 : Nice (Mary)
S82 : Date (Mary, John)
X9 : ¬Rides (x1 , f (John)) ∨ Rough (x1)
θ = { x8 / Mary }
X1 : ¬Rough(x4) ∨ ¬Date (Mary, x4)
θ = { x4 / John }
θ = { x1 / John }
X2 : ¬Rough (John)
X10 : Rough (John)
NIL