First Order Logic

By the end of this module student will be able to-

- Understand First order logic
- Follow Syntax and Semantics of First-Order Logic
- Write predicates in FOL
- Apply knowledge engineering process to a given problem

Natural languages are highly context oriented, noncom-positional and yes, highly ambiguous. The syntax of natural language follows part of speech and the sentences talk about subjects, nouns, adjectives, objects, verbs, relations amongst objects, etc. When these natural language sentences are represented using the knowledge representation languages, it must serve three purposes:

1. The represented knowledge needs to be free from natural language lacunas like context-dependency and ambiguity
2. The represented knowledge must preserve the objects and their relations defined in original natural language statements.
3. Apart from mere facts, the represented knowledge must have ability to represent universal truths in the problem domain.

The knowledge representation schemes emphasize on studying how knowledge can be represented and how effectively it can resemble the representation of knowledge in human brain. In reality, the knowledge is pretty complex and hence the knowledge representation system should have well defined syntax and semantics to represent complexities in the language. Human knowledge is represented in natural language and one of the prime activities of the human intelligence is reasoning. This demands the artificially intelligent knowledge representation mechanisms to support the reasoning process in a scientific way.

AI uses various knowledge representation schemes like logic, Semantic Networks, Frames, Conceptual dependency and Scripts. Out of these techniques, logic is a formal language and is classified in two main categories- propositional logic and predicate logic. The logic can be defined as a scientific study of the process of reasoning.

Characteristics of Knowledge Representation

The knowledge representation languages should have following characteristics-

a. The representation scheme should have a set of well-defined syntax and semantics to represent various kinds of knowledge of varying complexities.
b. The knowledge representation scheme should be fairly expressive so as to facilitate the inference mechanism in its reasoning process.
c. The representation must be efficient and should use only limited resources without compromising on the expressive power.

First Order Logic Model

The propositional logic lacks expressiveness of the language and typically it works in circumstances where the outcome is Boolean, i.e. either true or false. However the real life situations are more complex and tricky to be expressed in just true or false manner. In such scenario, the First Order Logic (FOL) is a suitable solution. It extends expressiveness of propositional logic, lifts some of the inference methods from there like modus ponens and resolution and gives solution to the asked query.

The First-order logic assumes that the world consists of objects with certain relations among them that do or do not hold. The model of first order logic contains-

a. Objects- The objects are the entities in the task environment.
   e.g. person, course , animals, nose, car
b. Relations- The objects in task environment can be related in various ways. Depending on these relation, the relation tuple may have one or more elements as part of itself.
   e.g. Female(Raavi), Born(name, birthyear), Buy(name car, price), Person(name, age, gender, occupation, city, contact_no), etc
c. Functions – When the object is related to exactly one object in a particular way, then that relationship is represented with functions, rather than a relation tuple.
   e.g every person has a mother. So the model may have unary mother function. Raavi→ Raavi's mother.

Syntax and Semantics of First-Order Logic(FOL)

The basic element of first order logic are the symbols. The symbols are used in three ways-

a. Constant symbols representing objects
b. Symbols those represent relationships, also called as predicates
c. Symbols those represent functions.
d. Connectives

Each predicate and function have certain number of arguments and it is called **arity** of that predicate or function. When the symbols representing objects and the ones representing predicates put together, they form a sentence. The sentences in FOL are either atomic sentences or complex sentences. Though the sentences represented by FOL don't signify any tense in the represented statement.

e.g. Marcus is a man is represented as Man(Marcus) wherein reading the represented statement, one cannot determine if Marcus *is* a man, *was* a man or *will be* a man.

The FOL statement are read in English as,

Relation(subject, object1,object2,..,object) e.g. Mother(Seeta, Reema) ➔ Read as Seeta is mother of Reema, and NOT as Seeta's mother is Reema. The FOL statements don't suggest tense, but while converting the statements into simple English, some tense is added.

Connectives used in FOL

The connectives are part of FOL syntax. It uses five connective symbols as-

a. ~: Not or negation
b. ∩ : And or conjunction
c. ∪: OR or inclusive disjunction
d. → : Implication
e. ↔ : Equivalence or if and only if; also called as bidirectional implication.

Atomic sentence

Typically, the atomic sentence state facts. The atomic sentence is constructed by using a predicate symbol followed by a list of arguments enclosed in parenthesis.

sentence	FOL representation
Marcus is a man	Man(Marcus)
Marcus was born in 40 AD	Born(Marcus, 40AD)
Marcus was a Pompeian	Pompeian(Marcus))
Ramesh is Snehal's grandfather	Grandfather(Ramesh, Snehal)
Seema is a 20 year old female student from Mumbai with contact number 02212345678)	Student(Seema, 20, f, Mumbai, 02212345678)

If the relation denoted by the predicate and its object symbol(s) hold true under any given interpretation, then the atomic sentence also holds true.

Complex Sentences

The complex sentences are created by connecting atomic sentences using logical connectives.

sentence	FOL representation
Bill is rich and famous	Rich(Bill) ∩ Famous(Bill)
Either maid stole the jewellery or she milked the cow	Stole(Maid, Jewellery) ∪ Milked(Maid, Cow)
Marcus is a man and so he is mortal	Man(Marcus) → Mortal(Marcus)
Brass doesn't glitter	~Glitter(Brass)

Quantifiers

When a certain property is true with a set of objects, the sentences are strengthened with quantifiers to express certain truth about those objects in the world. Such sentences use variables instead of object names. FOL uses two types of quantifiers- universal quantifiers and existential quantifiers.

Universal Quantifier

The universal quantifier is used to represent a certain fact that is always true **for all objects** in the given world. The universal quantifier is denoted with symbol ∀ (read as For All) and is accompanied by a variable to state the fact about those objects. The statement: $\forall x\ P(x)$, implies the **universal quantification** of the atomic expression P(x). In English, it means,: "*For all x, P(x)* is true ". Here $\forall x$ means the expression is stated for all the objects x in the world.

sentence	**FOL representation using universal quantifier**
All men are mortal	$\forall x$ Man(x) → Mortal(x)
Every child likes toys	$\forall c$ Child(c) → Likes(c, toys)
Barber shaves all men who do not shave themselves	$\forall m$ Man(m) ∩ ~Shave(m,m) → Shave(Barber,m)
All cells adjacent to Wumpus have stench	$\forall x,y$ Wumpus(x,y) ↔ Stench(x+1,y) ∪ Stench(x-1,y) ∪ Stench(x,y+1) ∪ Stench(x, y-1)

Here, both x and c are called variables. The statements those use universal quantifier, always **state some implied** fact. The universal quantifier is always associated with a variable. The variable denotes condition on LHS and the implied fact resides on the RHS of the statement. Typically, the objects associated with all, everyone, not all, anything, everything, etc kinds of descriptive words are represented with universal quantifiers. The variable associated with universal quantifier appears on both LHS and RHS to indicate relationship between both the sides of implications.

Existential Quantifier

The existential quantifier is used to represent a certain fact that is true **for some objects** in the given world. The existential quantifier is denoted with symbol ∃ (read as there exists) and is accompanied by variable to state the fact about those objects. The statement : $\exists x\ P(x)$, represents the **existential quantification** of *P(x)*. In English it means, "There exists an *x,* such that *P(x) is true*" . $\exists x$ states that the stamen P(x) is true for at least one object *x* in the world.

sentence	FOL representation using universal quantifier
Some Indians like dosa.	∃ x Indian(x) ∩ like (x,dosa)
Some Dogs are brave	∃ d Dog(d) ∩ brave(d)
Sameer likes some action movies	∃ m Action_Movie(m) ∩ likes(Sameer,m)

Here, both x and d are called variables. The statements those use existential quantifiers, always have **AND connector** in the statement. Typically, the objects associated with some, a, few, no, no one, etc kind of descriptive words are represented with existential quantifiers.

Nested Quantifiers

Some complex statements can be represented only with multiple identifiers.

sentence	FOL representation using universal quantifier
Once dead, everyone is dead for all later times.	∀x ∀t1, ∀t2 Dead(x,t1) ∩ Greater_Than(t2,t1) → Dead(x, t2) OR ∀x, t1, t2 Dead(x,t1) ∩ Greater_Than(t2,t1) → Dead(x, t2)
Everyone is loyal to someone	∀x ∃y Loyalto(x,y)
All brothers are siblings	∀x,y Brother(x,y) → Sibling(x,y)
Children of same parent are siblings	∀x,y,z Parent(x,y) ∩ Parent(x,z) ∩ x < > y* → Sibling(y,z)

*- Not Equal to.

To convert such statements into FOL, the natural language statement should be read from left to right, identify objects, their relationships and accordingly they should be converted into the FOL.

Relationship between Universal and Existential Quantifiers

The universal and existential quantifiers are connected with each other through negation.

Statement	Logically Equivalent Statement
∀x ~P(x)	~ ∃x P(x)
e.g. Mortals do not live for more than 150 years.	No mortal lives for more than 150 years
~ ∀x P(x)	∃x ~P(x)
Not everyone lies.	There are some people who do not lie.
∀x P(x)	~ ∃x ~P(x)
e.g. Everyone likes sweets	There exists no one who does not like sweets.
∃x P(x)	∀x ~P(x)
There is someone who likes sweets	Everyone doesn't like sweets

Writing Predicates in FOL

Writing correct predicates is a skilful job but isn't very difficult one, too. To convert the English statements into the FOL, one must take care of following guidelines-

1. Identify objects, their relationships and determine arity of every relationship. Read the sentence(s) carefully. If there are more than one sentences, try to figure out common part so that generalisation is possible or some inference rules can be applied to derive new statements. This eventually helps to prove the given goal.
2. If a sentence has all, every, any none, each, etc, then the FOL representation will have universal quantifier.
3. If a statement has one, some, no, a, etc, then the FOL representation will have existential quantifier.
4. The universal quantifier always use implication to connect two or more literals and the existential quantifier always use the AND (∩) logical connector to connect the literals.
5. The implication statement has preconditions on LHS. When all the predicates on LHS are true/tautology, the RHS is true. The variable(s) associated with universal quantifier have presence on both sides of implication.

Consider the following individual statements. Observe how the number of predicates in statement and their arity changes.

Statement	**Logically Equivalent Statement**
Cat like fish	Like(cat,fish)
Tom, the cat, likes fish	Cat(Tom) ∩ Like(Tom,fish)
Tom, the cat, likes tuna fish	Cat(Tom) ∩ Fish(Tuna) ∩ Like(Tom,Tuna)
All cats like fish	∀x Cat(x) → like(x,fish)
All cats like tuna fish	∀x cat(x) ∩ Fish(Tuna) → like(x,Tuna)
Ram likes Dosa	Like(Ram, Dosa)
All Indians like dosa	∀x Indian(x)→ likes(x,dosa)
Some Indians like dosa	∃x Indian(x) ∩ likes(x,dosa)
No Indians like dosa	~∃x Indian(x) ∩ likes(x,dosa)
Every Indian likes some dosa	∀x ∃y Indian(x) ∩ Dosa(y) → like(x,y)
Ram, an Indian likes masala dosa	Indian(Ram) ∩ Dosa(Masala) ∩ likes(Ram, Masala)

Convert English Statements into FOL

Let's apply these rules to convert the English statements into FOL.

Example-

1. Marcus is a man.
2. Apple is food.

   The above sentences have Marcus & Apple as objects and man & food are their relationships. Being a man or a food item isn't dependent on anything. We take the relationships as predicate names and objects as the predicate elements to give the predicates-

   Man(Marcus).

   Food(Apple).

3. Marcus was born in 40 A.D.
   This statement has year of birth as relation which binds together two objects- Marcus and year of birth. The relationship of birth can only be answered by minimum two entities- who was born? When he/she was born? If the sentence has more information, additional questions would be where he/she was born, etc. For the given statement, the predicate must have at least two objects as,

   Born(Marcus, 40) ➔ if rest of the statements from KB state all the years in AD.
   Born(Marcus, 40 AD) ➔ if rest of the statements from KB state some years in AD and some in BC.

4. Sonia is Raj's mother.
   A person doesn't become mother unless she has a son or a daughter. So the relationship of mother must have two objects- who is the mother and who's mother she is.

   Mother(Sonia,Raj). ➔ correct
   Mother(Raj, Sonia) ➔ Incorrect

5. Kavya likes to play tennis and football.
   This statement has one relationship- like and two objects- play tennis and play football. The like relationship can be associated with various things like other hobbies, movies, food items, cities, other activities etc. But the basic like relationship needs two answers- who likes it and what is being liked. Therefore, it can have only two objects. To represent the above sentence in

   FOL it could be considered as-
   Kavya likes to play tennis and Kavya likes to play football.

   The FOL equivalent statement will be-
   Likes(Kavya, Tennis) $\cap$ Likes(Kavya, Football)➔ Correct
   Likes(Kavya, Tennis, Football) ➔ Incorrect

6. Meera eats all kinds of food.
   This statement talks about eating relationship and objects like who eats & what is being eaten. The eat predicate needs minimum two objects to convey the message. But the sentence says more than just eating. It talks about all the objects those come in the category of food. Hence the universal quantifier gets associated with food as the predicate. Also, being a food item is not dependent on any other entity so it gets represented just with one object.

   $\forall$f Food(f) → Eats(Meera,f) ➔ correct

   $\forall$f Eats(Meera,f) → Food(f) ➔ Incorrect
   The first sentence says, for all f that is food, Meera eats it, while the second sentence says for all f that Meera eats, is food. The second statement isn't really true in the world.
7. Arya eats everything that Sansa eats.

This statement also talks about collection of things being eaten by Sansa. If Sansa eats anything, Arya eats the same.

$\forall f$ Eats(Sansa, f) → Eats(Arya,f) ➔ correct

Note- The given statement doesn't mention if they are eating only food, so do not create a predicate for the same.

$\forall f$ food(f) $\cap$ Eats(Sansa, f) → Eats(Arya,f) ➔ Incorrect

8. A barber shaves all the men in city who do not shave themselves.

   This sentence states when a barber shaves all men in the city. One must write all conditions on LHS and their implication on RHS. The relations in statement- Barber, Man, Shave, where Barber and Men are unary relationships while the shaving activity needs two elements.

   $\exists b\ \forall m$ Barber(b) $\cap$ Man(m) $\cap$ ~Shave(m,m) → Shave(b,m)

9. Anything everyone eats and isn't killed by it is food.

   Relationships in statement are- eat, killed_by and food. From the previous example, eats needs two objects to convey the meaning and food is unary relationship. The Relation killed by as well needs two objects- who is killed and by whom. Universal quantifiers are associated with anything and everyone.

   $\forall x,f$ Eats(x,f) $\cap$ ~Killed_by(x,f) → food(f) ➔ Correct

   $\forall x,f$ Eats(x,f) $\cap$ ~Killed(x,f) → food(f) ➔ Incorrect

   The second statement says x hasn't killed f which isn't in accordance with the given fact.

10. The law states that it is crime for an American to sell weapons to hostile nations.

    Relationships in the statement are- being American, Selling, Weapons, hostility and being criminal. American, weapon and hostile_nation are unary relationships. Selling may have various objects such as who sold it, to whom it was sold, what was sold, what were the selling price, why it was sold etc. finally, the law holds good for all Americans, for all hostile nations and for all kinds of weapons.

    $\forall a, h, w$ American(a) $\cap$ Weapon(w) $\cap$ Hostile_Nation(h) $\cap$ Sells(a, w, h) ➔ Criminal(a).

Common Mistakes

1. Predicate name and arguments have same name.

   Sometimes the objects and relationships get intermixed and they exchange places with each other.

   e.g. All cats like tuna fish, gets represented as,

   $\forall c\ Cat(c) \cap Fish(tuna) \rightarrow like(cat, tuna)$

2. Words in given statement are changed.

   This is a very common mistake performed while representing English sentences into FOL.

   Example- Kirti eats peanuts and is still alive, gets incorrectly converted into FOL as,

   $Eats(Kirti,Peanuts) \cap \sim dead(Kirti)$

   The original statement talks about being alive and the word dead doesn't appear in it at all. The FOL should be accurate representation of the message conveyed in original statement.

   Example –

   Marcus tried to assassinate Caesar, gets incorrectly converted into FOL as,

   Assassinate(Marcus, Caesar)

   There is big difference between trying to assassinate and actual assassination, which is totally ignored if the words given in sentence are changed.

3. Subject of statement is changed during FOL conversion.

   This is another common mistake performed in FOL representation.

   E.g. Neel is brother of Neha gets represented as s

   Brother(Neha, Neel)

The Knowledge Engineering Process

Knowledge engineering is the general process of knowledgebase creation. A knowledge engineer, who is good at technical aspects of AI but may not necessarily have domain knowledge, is responsible for successful implementation of knowledge base creation process.

All the projects for which some KB is built are not the same. Although the projects differ with each other in terms of content, scope, complexity, difficulty, etc, they follow these steps as part of knowledge engineering process.

1. Identify the task

   This step is analogous to PEAS identification step in designing intelligent agent's solutions. Identification of task limits scope of the work, defines the exact problem, inputs and outputs. It also helps in finalizing the kinds of queries that will be handled by the KB.

2. Assemble the relevant knowledge

   This step is analogous to realising "How the world evolves" in agent design. The knowledge engineer may not be the domain expert for the given problem. In such cases he/she sits with domain experts to get more knowledge about the same. This step is also called knowledge acquisition step. By completion of this step, the knowledge engineer must know definitions, interpretations of things in problem domain, consequences of chosen actions, etc.

3. Decide on vocabulary of predicates, functions and constants

   This is the first step towards the actual implementation. It transforms the important domain specific concepts into logical concepts. The decisions are taken on programming style to be used, representations of objects and predicates etc. Output of this step is Ontology definition for the domain.

4. Encode domain specific general knowledge

   Once the vocabulary is defined, it is used to encode the in general i.e. domain specific knowledge. The rules and basic facts are encoded as a starting step of KB creation. At this stage reveals if there are any errors in vocabulary definition. In such scenario, the knowledge engineer returns to step 3 to fix and revise through process.

5. Encode specific knowledge of problem specific instance

   The completion of encoding the domain specific knowledge asserts that vocabulary definition is correct and almost complete. At such a stage problem specific knowledge can be encoded correctly. If the ontology is well defined, this step will be effortless. The knowledge engineer and team writes simple atomic instances about the facts in the problem specific instance.

6. Run queries to inference procedure to get the answers

   This is the step where one can verify the correctness of the domain specific and problem specific knowledge. The inference rules are executed to give answers to the posed queries.

7. Debug the knowledgebase.

The incorrect answers reveals that inference procedure isn't sound. Lack of answers to the queries uncover missing axioms in KB. In such cases, the KB needs to be debugged to locate missing axioms, weak axioms by checking where chain of reasoning stops in unexpected manner.

Repeat the steps 6 and 7 till the perfect results are not obtained.

Knowledge Engineering for Crime Investigation Agent

Problem- Design a crime investigation agent that goes through incident narration(s) or a question-answer system to give list of suspects and if possible, culprit to the crime.

1. Identify the task(s)

- From the problem statement, the tasks are identified as-
  - Find the motive
  - Find all the suspects
  - List all culprit
- Scope of work is based on answers to the questions

2. Assemble relevant knowledge

This step is involved with collecting general knowledge about the problem domain. The crime investigation general knowledge answers questions like what are the typical crime motives, how the science of crime investigation works, how the world around the victims and culprits evolves, etc. some of the points from this relevant knowledge can be listed as-

- Basic definitions of – crime, suspect, evidence & proof, legal evidence, alibi etc.
- Types of crimes – robbery, shoplifting, terrorism, kidnapping, blackmail, harassment, organized crime, etc.
- Evidence- fingerprints, DNA, belongings, weapons, leaving some trail behind, mobile locations, call record data, alibi, etc
- Different consequences of different crimes- shoplifting is minor crime while murder and conspiracies are very serious.
- Science of crime investigation
- Human virtues responsible for crime – hate, jealousy, professional crime, psychological issues, money, revenge, challenge, blackmail, corporate rivalry, etc.
- People- social status, economic status, presence or absence of respect or credibility in society with money and/or education, work culture of different professions, availability of tools due to professions, knowledge about Non-tool entities which can act as tool in crime (a very hard object can work like a hammer), interpersonal relations- like relationships, break-ups, affairs, too close friendship wherein on can do anything for the other, etc.

Every crime story gives a new information, so the list of relevant knowledge about the same may never be complete.

3. Decide on vocabulary of predicates, functions and constants

   At this stage, the knowledge engineer decides on how exactly to represent the knowledge and needs to choose programming language too. Let's choose PROLOG language for the given problem. Some of the vocabulary for the crime investigation agent can be given as-

   ◦ person(name, age, sex, occupation)
   ◦ had_affair(name, name)
   ◦ killed_with(name, object)
   ◦ killed(name)
   ◦ killer(name)
   ◦ motive(vice)
   ◦ smeared_in(name, substance
   ◦ owns(name, object)
   ◦ operates_identically(object, object)
   ◦ owns_probably(name, object)
   ◦ suspect(name)

4. Encode domain specific knowledge

   Some domain specific knowledge for the crime investigation can be encoded in PROLOG as,

   ◦ operates_identically(wooden_leg, club).
   ◦ operates_identically(bar, club).
   ◦ operates_identically(pair_of_scissors, knife).
   ◦ operates_identically(football_boot, club).
   ◦ owns_probably(X, football_boot) :- person(X, _, _, football_player).
   ◦ owns_probably(X, pair_of_scissors) :-person(X, _, _, hairdresser).
   ◦ owns_probably(X, Object) :- owns(X, Object).
   ◦ suspect(X) :-motive(money) , person(X, _, _, pickpocket). ➔ motive money
   ◦ suspect(X) :- motive(jealousy), person(X, _, female, _) had_affair(X, Man), has_affair(VictimFemale, Man). ➔ motive jealousy
   ◦ suspect(X) :- killed_with(Victim, Weapon), operates_identically(Object, Weapon), owns_probably(X, Object).
   ◦ killer(Killer) :- person(Killer, _, _, _) , killed(Killed), Killed <> Killer, suspect(Killer), smeared_in(Killer, X), smeared_in(Killed, X). ➔homicide
   ◦ killer(Killer) :- person(Killer, _, _, _) , killed(Killed), Killed = Killer. ➔ Suicide

5. Encode problem specific instance

Let's assume an incident involving some people from different professions, their past and present relationships, ownership of some objects due to their profession and who is killed. This part can be encoded in PROLOG following the vocabulary defined in step 3 as,

- person(tyrion, 55, m, carpenter).
- person(jorah, 25, m, football_player).
- person(jamie, 25, m, butcher).
- person(john, 25, m, pickpocket).

- had_affair(ygritte, john).
- had_affair(shae, tyrion).
- had_affair(rose, john).

- killed_with(ygritte, club).
- killed(ygritte).

- smeared_in(tyrion, blood).
- smeared_in(ygritte, blood).
- smeared_in(jorah, mud).
- smeared_in(john, chocolate).
- smeared_in(shae, chocolate).

- owns(tyrion, wooden_leg).
- owns(john, pistol).

6. Run the queries to get the answers

   One can run the queries as suspect(X) or killer(X) to find out list of suspects and the culprit. The query will be executed by combining both the parts of information, problem specific and domain specific to give the answer.

7. Debug the knowledge base

If the answers are not correct, iterate through step 4 and 5 to detect missing facts and missing rules to detect the place where chain of reasoning stops in unexpected manner.

Summary

- Knowledge representation and reasoning is an important part of artificial intelligence.
- Knowledge representation languages should be declarative, compositional, expressive, context-independent, and unambiguous.
- AI uses various knowledge representation schemes like propositional and predicate logic, semantic networks, frames, conceptual dependencies, and scripts.

- The knowledge representation languages have set of well-defined Syntax and semantics to represent various kinds of knowledge. They are expressive enough to facilitate inference mechanism in the reasoning process.
- First order logic is improvement over propositional and is more expressive than propositional logic.
- First order logic assumes the world in terms of objects and their relationships.
- The FOL sentences represent facts of the task environment.
- The FOL atomic sentences are formed by using a predicate symbol followed by a list of arguments enclosed in parenthesis.
- The FOL uses Universal and existential quantifiers to convey more information about the objects in sentences.
- In FOL, the natural language sentences are converted into knowledge representation format by identifying objects, their relationships and then by determining arity of every relationship.
- Some FOL sentences use Universal or existential or sometimes both of the quantifiers, depending on the information being conveyed by the statement.
- The universal quantifier uses implication to connect two sides of well-formed formula, where the literals on left side indicate conditions and the ones on right hand side indicate consequences
- The existence in quantifier uses AND as a logical connector to connect the literals in the statement.
- In higher order logic a predicate can have other predicates or functions as arguments which in turn can have predicate or function quantifiers associated with them.
- Knowledge engineering is seven step process of creating knowledge base for the given problem. The knowledge engineer goes through a well defined seven step process during construction of knowledge base.

Test Your Knowledge

Multiple choice Questions.

1. Which of the knowledge representation schemes are used by AI?
   a. logic
   b. Semantic Networks
   c. Frames
   d. Conceptual dependency
   e. Scripts
   f. **All of the above**

2. FOL is more expressive than propositional logic. State true or false
   a**. True**
   b. False

3. Generalized modus ponens is lifted version of modus ponens in propositional logic. State true of false.
   a. **True**
   b. False

4. Which of the following statements are equivalent?

I. For all x ~P(x) II. ~ There exists x P(x)
III. There exists x P(x) IV. There exists x ~P(x)
a. **I and II**
b. I and II, III and IV
c. III and IV
d. II and IV
e. None of the above

5. What is the correct FOL equivalent representation for - "Some real numbers are rational"
a. There exists x, Real(x) OR Rational(x)
b. For all x, Real(x) —> Rational(x)
c. **There exists x, Real(x) AND Rational(x)**
d. There exists x, Real(x) —> Rational(x)

6. State true of false- There exist three types of quantifiers, Universal Quantification, Existential Quantification and Hybrid Quantification
a) True
b**) False**

7. The knowledge engineering process involves
a. Creation of knowledge base right from content decision, representation, implementation and execution.
b. Knowledge acquisition from domain experts
c. Deciding programming languages and implementation
d. Converting the natural language sentences using some knowledge representation scheme.

8. State true or false.
Typically, the answers to the queries in knowledge engineering process are always correct in the very first attempt and hence the engineer doesn't have to debug the KB to fix errors.
a. True
b. **False**

10. The knowledge engineering involves
a. **Domain specific as well as problem specific knowledge**
b. Only domain specific knowledge
c. Only problem specific knowledge
d. None of the above.

Theoretical questions

1. Differentiate between predicate logic and first order logic.
2. State and explain characteristics of knowledge representation languages.
3. Explain Syntax and semantics of first order logic.
4. Represent the following sentences in first order logic.
   a. All that glitters is not gold.
   b. No mortal can live for more than 150 years.
   c. Either curiosity or Jack killed the cat, whose name was Kitty.

d. All the Pompeian died in the volcano erupted in 79 AD.
e. Anything anyone eats and isn't killed by it, is food.

5. Apply the knowledge engineering process to medical diagnosis problem. list all the steps in knowledge engineering and explain contents of each of them for the given problem.

INFERENCE IN FIRST-ORDER LOGIC

After studying this module, students will be able to-

- Understand the process of unification and lifting
- Apply inference rules like modus ponens, modus tollens, And-elimination, bidirectional-elimination
- Solve reasoning problem using forward chaining
- Solve reasoning problem using Backward chaining
- Solve reasoning problem using resolution

First order logic(FOL) is expressive and stronger than propositional logic. It can represent almost every statement in the word. Another strength of FOL lays in generalization of certain facts that are typically true in the domain areas. The inference procedures incredibly use these FOL features to answer the queries.

Special Inference Methods for first-Order Sentences

FOL inferences can be done by lifting some inference methods from propositional logic like modus ponens, modus tollens, and-elimination, bidirectional-elimination, de-Morgan's law, laws of equivalence, satisfiability and validity, etc. Apart from these methods, FOL uses some more inference procedures like Universal Instantiation, Existential Instantiation, Skolemization, Generalized Modus Ponens, Unification, Lifting, etc.

Universal Instantiation

The **Universal Instantiation rule states that** any sentence can be inferred by substituting a **ground term** for the variable in the given statement.

Mathematically, it is denoted with the function Substitution(θ, a) indicating that statement a is operated on by substitution θ.

$$\frac{\forall v\ a}{\mathrm{Substitution}(\{\mathrm{v/g}\}, a)}$$

This is read as, in statement a containing universal quantifier for the variable v, all of the appearances of v have been substituted by ground literal g.

e.g. Consider the following axioms.

a. All the trained collies are good dogs.

b. Tom is a trained collie.

$\forall$x Trained(x) $\cap$ Collie(x) $\rightarrow$ Is_good_dog(x)

This can be used to infer –

$\forall$x Trained(Tom) $\cap$ Collie(Tom) → Is_good_dog(Tom)

As the universal quantifier states a fact that is always true in the world, the statement that is inferred with substitution of the ground literal is also true. Universal Instantiation can be applied multiple times to yield many different conclusions.

Existential Instantiation

The statements with existential quantifiers are true for some objects in the world. So the **Existential Instantiation rule** states that a sentence can be inferred by substituting a term for the variable in the given statement that doesn't appear elsewhere in knowledge base. Fundamentally, the existential sentence states that there exists some object that satisfies a condition, and the existential instantiation process just giving a name to that object.

Mathematically, it is denoted with the function Substitution(θ, α) indicating that statement a is operated on by substitution θ.

$$\frac{\exists v\ \alpha}{\text{Substitution}(\{v/k\}, \alpha)}$$

This is read as, in statement α containing existential quantifier for the variable v, all of the appearances of v have been substituted by a constant symbol k, that does not appear elsewhere in the given knowledge base.

e.g. Consider the following axiom.

Some child has toy.

$\exists$c child(c) $\cap$ has(c,toy)

This can be used to infer –

child(chintu) $\cap$ has(chintu,toy)

This statement would be true as far as chintu doesn't appear anywhere else in the knowledge base. Unlike universal Instantiation, the existential Instantiation can be applied once, and then the existentially quantified sentence can be discarded.

Generalised Modus Ponens

In real life the knowledge bases do not contain everything straightforward. The substitution operation in universal and existential instantiation works only if all the statements are directly given. In the earlier example, Tom is inferred to be a good dog by finding some x such that x is trained and x is a collie then infer that x is a good dog. i.e. if the sentences are already present the knowledgebase, then the substitution implies the conclusion.

If the KB contains following statements-

1. All collies are trained dogs.
   $\forall$ c Collie(c) ➔ Trained_dog(c)
2. All the trained collies are good dogs
   $\forall$x Trained_dog(x) $\cap$ Collie(x) ➔ Is_good_dog(x)
3. Fred is a collie.
   Collie(Fred)

This KB doesn't state if Fred was trained, still we can prove that Fred is a good dog by using the substitutions {c/Fred, x/Fred} in Trained(x) and Collie(c) and we can infer the implication conclusion as Is_good_dog(Fred).

This process of capturing inference process as a single inference rule is called generalized modus ponens. Mathematically its defined as, for atomic sentences a_i, a_i' and p, if there exist some substitution such that *Substitution(*θ*,* a_i'*)= Substitution(*θ*,* a_i*),* for all *i.*

$$\frac{a1', a2', \ldots, an', \quad (a1 \cap a2 \cap \ldots \cap an \rightarrow p)}{\mathrm{Substitution}(\theta, p)}$$

This rule uses n+1 premises to infer the conclusion. The first n are the atomic sentences on the left hand side of the implication and there is one premise on the right hand side of it. The inference is drawn by applying substitution θ to conclude statement p.

Generalized Modus Ponens is a sound inference rule. The generalized Modus Ponens is a **lifted** version of Modus Ponens, raised from propositional logic to the first order logic.

UNIFICATION AND LIFTING

The substitutions in FOL are possible only when those two or more expressions being compared are a match for the same. The predicates must be equal in terms of their predicate names, their number of arguments, their data types, etc. in FOL this process is called unification. Any substitution that renders two or more expressions equal, is called unifier for the expression. The unification function assigns bindings to variables where these binding are either a constant, a functional expression or another variable.

The fundamental idea behind unification is simple. To unify any two literals, their predicate names are compared first. If that's a match, the process continues further otherwise the literals cannot be unified.

e.g. Master(Sam, Fred) and Erupted(Volcano, 79AD) cannot be unified for the obvious reasons.

If the predicate symbols of any two literals match then their arguments are checked one by one pairwise, if first one matches the next pair gets compared and soon.

Literal 1	Literal 2	Unification?	Substitution
L1(x,y)	L2(A,A)	Yes	{x/A}
L1(x,x)	L2(A,B)	Fail	
L1(x,y)	L2(A,B)	Yes	{X/A, y/B)
L1(x,y)	L2(A,A)	Fail	
L1(f(x),y)	L2(f(A),B)	Yes	{X/A, y/B)

Algorithm List Unify(Literal L1, Literal L2)

{// The algorithm Unify matches two literals L1 and L2.
//Algorithm returns a List named SUBSTITUTION containing all the substitutions used to unify L1 and L2.
//An empty SUBSTITUTION List indicates that match was found without any substitution.
//On failure, the SUBSTITUTION List returns a single value FAIL.

1. If L1 or L2 are both variables or constants, then-
   a. If L1 and L2 are identical, then return NIL.
   b. Else if L1 is a variable, then if L1 occurs in L2 then return {FAIL}, else return (L2/L1).
   c. Else if L2 is a variable, then if L2 occurs in L1 then return {FAIL} , else return (L1/L2).
   d. Else return {FAIL}.
2. If the initial predicate symbols in L1 and L2 are not identical, then return {FAIL}.
3. If LI and L2 have a different number of arguments, then return {FAIL}.
4. Set SUBSTITUTION to NIL.
5. For i ← 1 to number of arguments in L1 :
   a. Call Unify with the ith argument of L1 and the ith argument of L2, putting result in S.
   b. If S contains FAIL then return {FAIL}.
   c. If S is not equal to NIL then:
      i. Apply S to the remainder of both L1 and L2.
      ii. SUBSTITUTION: = APPEND(S, SUBSTITUTION).
6. Return SUBSTITUTION.

The forward and backward chaining as well as resolution uses unification process for reasoning.

Representing Reasoning Process Using And-Or Graph

And-OR graph is a hierarchical structure with goal as the root and subgoals as the internal nodes. The leaf nodes contain the information needed to solve the problem. There might be multiple ways to achieve a single goal or subgoal which are represented by degree of that goal/subgoal node. The goal/subgoal could be achieved with one rule or combinations of the rules. s

e.g. A person can own the car by stealing it or by buying it with earned money.

This sentence can be represented in FOL as,

Steal(Person,Car) ∪ [Earn(Person,Money) ∩ Buy(Person,TV)] → Own(Person,Car)

This can be represented using And-OR graph as-

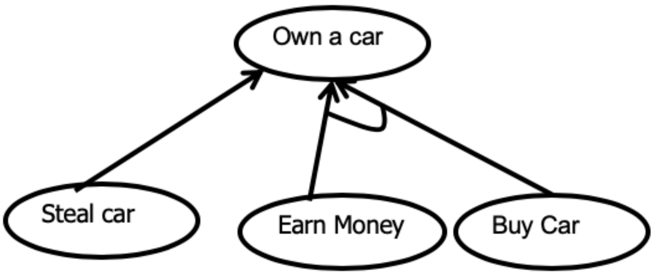


Fig: And-OR graph

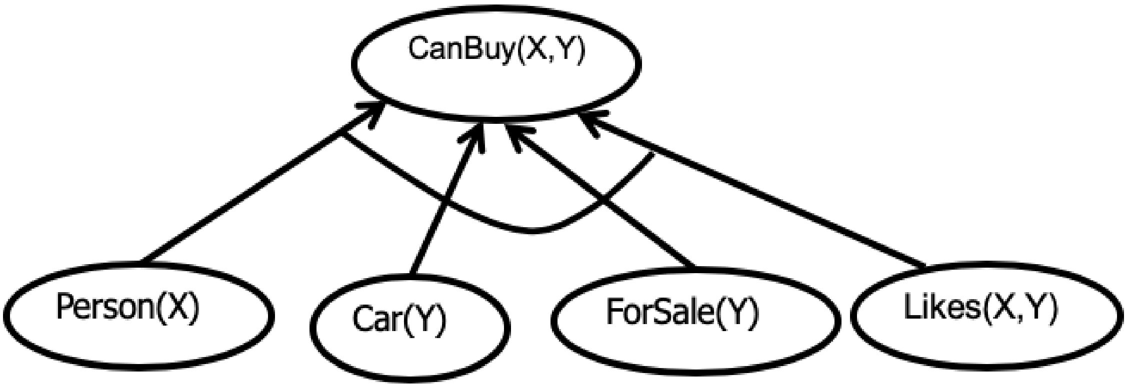


Fig: And-OR graph with an example CanBuy rule

Both forward reasoning and backward reasoning processes can be detailed using And-OR graph by showing intermediate steps at every level. Typically it is preferred to solve backward chaining examples using And-OR graph.

Forward-Chaining Algorithm

Forward chaining is one of the most popular inference methods used for reasoning. Forward chaining is most appropriate reasoning strategy for the exploration kind environments wherein one keeps on receiving new pieces of information as they progress through the solution. Forward chaining can derive all the possible sentences in the KB, i.e. it is a complete inference method. Also, it preserves the truth in the generated statements, i.e. it is also holds soundness property of inference methods.

The forward chaining starts with some facts and applies rules to find all possible conclusions. Therefore, it is also known as Data Driven Approach.

Basically, the standard procedure of forward-chaining operates by repeating the following sequence of operations until the solution to the query isn't found.

1. Inspect the rules to locate the one who's If part is satisfied by the current contents of KB.

2. Execute the rule, add the facts from Then part to the KB.

Examples - Reasoning with Forward Chaining

1. Translate the following set of sentences into FOL. Use forward chaining to prove Raja is angry.
    a. Rimi is hungry
    b. if Rimi is hungry, she barks.
    c. if Rimi is barking then Raja is angry.

    FOL representation-
    a. Rimi is hungry
       Hungry(Rimi)
    b. if Rimi is hungry, she barks.
       Hungry(Rimi)➔ Bark(Rimi)
    c. if Rimi is barking then Raja is angry.
       Bark(Rimi)➔ Angry(Raja)

    Forward Chaining-
    From (a), (b) and modus ponens,
    d. Bark(Rimi)
       From (d), (c) and modus ponens,
    e. Angry(Raja)
    Hence Proved.

    And-OR graph- As the KB does not contain any variables, there is no substitution.

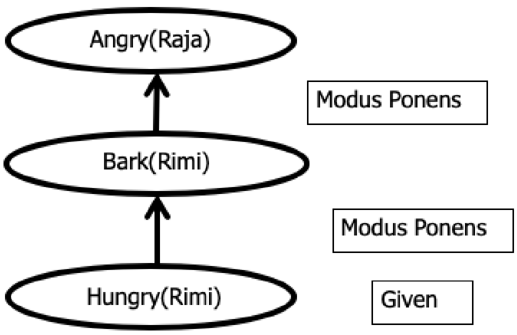


Fig: And-OR graph for Dog barking problem

2. Translate the following set of sentences into FOL. Use forward chaining to prove Fred is at museum.

a. Fred is a collie.
b. Sam is Fred's master.
c. The day is Saturday
d. It is cold on Saturday.
e. Fred is trained.
f. Trained collies are good dogs.
g. If a dog is a good dog and has a master, then he will be with his master.
h. If the day is Saturday and the day is cold, the Sam is at the museum.

FOL Representation-

a. Fred is a collie.
   Collie(Fred)
b. Sam is Fred's master.
   Master(Sam, Fred)
c. The day is Saturday
   Day(Saturday)
d. It is cold on Saturday.
   Cold(Saturday)
e. Fred is trained.
   Trained(Collie)
f. Trained collies are good dogs.
   ∀ x Trained(x) ∩ Collie(x) ➔ GoodDog(x)
g. Good dogs who have a master, then they are always with their master.
   ∀ x,y,z GoodDog(x) ∩ Master(y,x) ∩ At(y,z) ➔ At(x,z)
h. If the day is Saturday and the day is cold, the Sam is at the museum.
   Day(Saturday) ∩ Cold(Saturday) ➔ At(Sam, Museaum)

Forward Chaining-

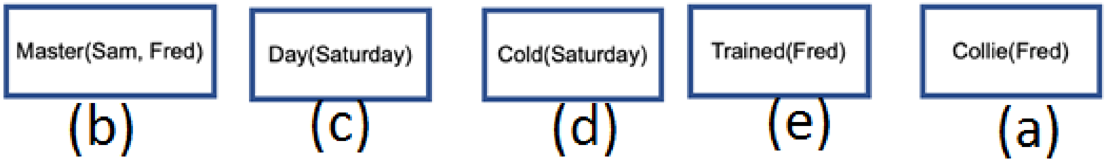
From (a), (e), (f) and Generalized modus ponens, {x/Fred}

i. GoodDog(Fred)
   From (c), (d), (h) and Generalized modus ponens,
j. At(Sam, Museaum)
   From (i), (b),(j), (g) and Generalized modus ponens, {x/Fred, y/Sam, z/Museaum}
   At(Fred, Museaum)

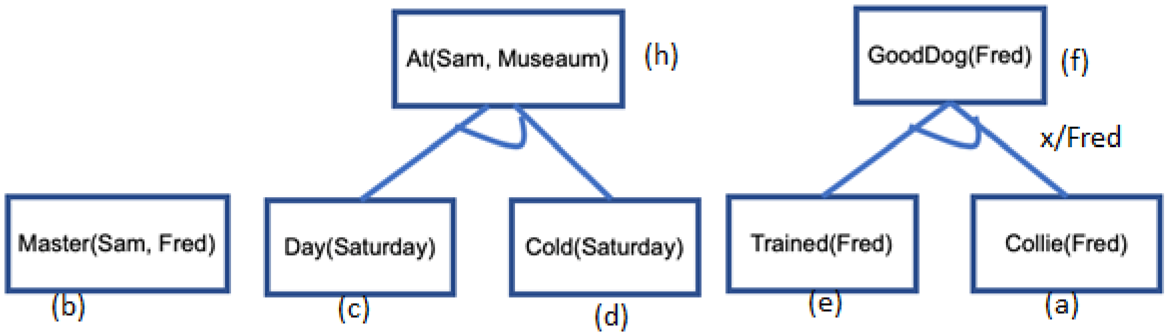
Hence proved.

And-OR graph solution-

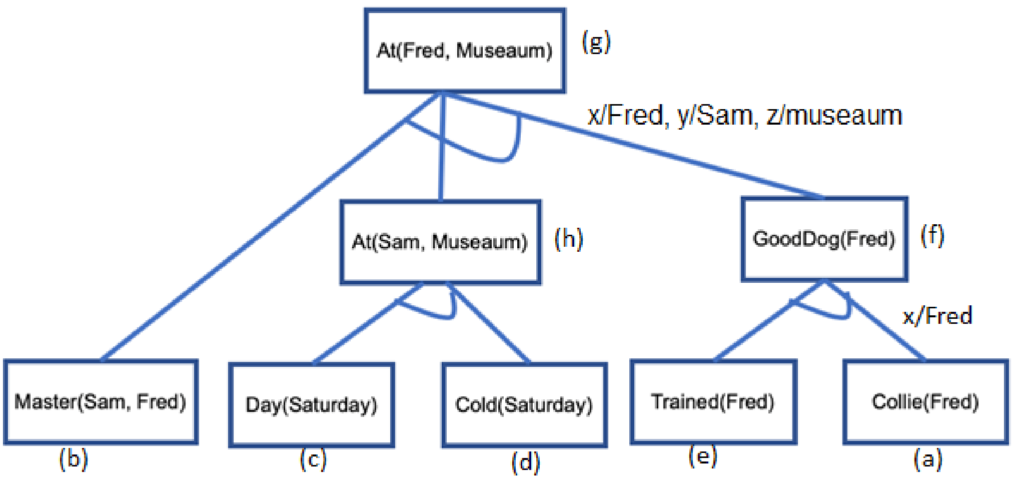
Step 1-


Master(Sam, Fred)
(b)
Day(Saturday)
(c)
Cold(Saturday)
(d)
Trained(Fred)
(e)
Collie(Fred)
(a)

Step 2-



Step 3-



3. Translate the following set of sentences into FOL. Use forward chaining to prove Colonel West is a criminal.

   The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

   FOL Representation-

   a. $\forall$ x,y,z American(X) $\cap$ Weapon(y) $\cap$ HostileNation(z) $\cap$ Sell(x,y,z)$\rightarrow$ Criminal(x)
   b. Country(Nono)
   c. Enemy(Nono, America)
   d. $\exists$ m Has(Nono,m) $\cap$ Missile(m)
   e. $\forall$ m1 Missile(m1) $\cap$ Has(Nono,m1)$\rightarrow$ Sell(West, m1, Nono)
   f. American(West)

The given statements lack some knowledge that is essential to prove the given query. The additional knowledge with their FOL representation go as-

g. Countries those are enemy of America are hostile countries.
   ∀ c Country(c) ∩ Enemy(c, America) → HostileNation(c)
h. All missiles are weapons.
   ∀ m2 Missile(m2) → Weapon(m2)

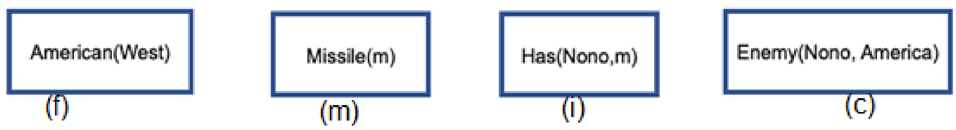
Forward Chaining

From (d) and AND-Elimination rule,

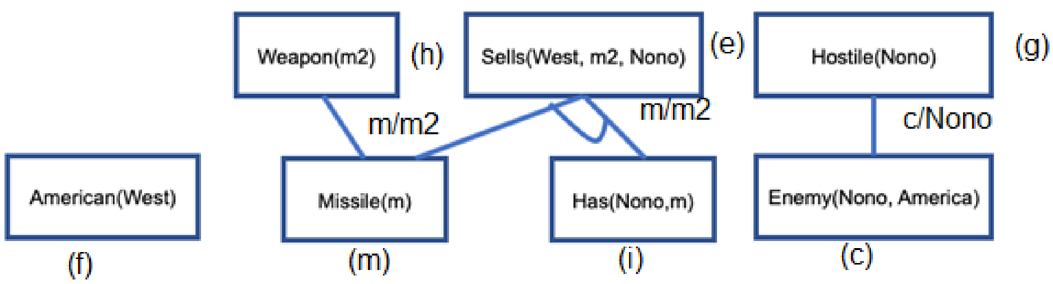
i. Has(Nono, m)
j. Missile(m)
   From (j), (h) and Generalized Modus Ponens, {x/m}
k. Weapon(m)
   From (b), (c), (g) and Generalized Modus Ponens, {c/Nono}
l. HostileNation(Nono)
   From (i),(j),(k),(e) and and Generalized Modus Ponens, {m1/m2}
m. Sell(West, m2,Nono)
   From (f), (k), (l), (m), and Generalized Modus Ponens, {x/ West, y/m2, z/Nono}
n. Criminal(West)

Hence Proved

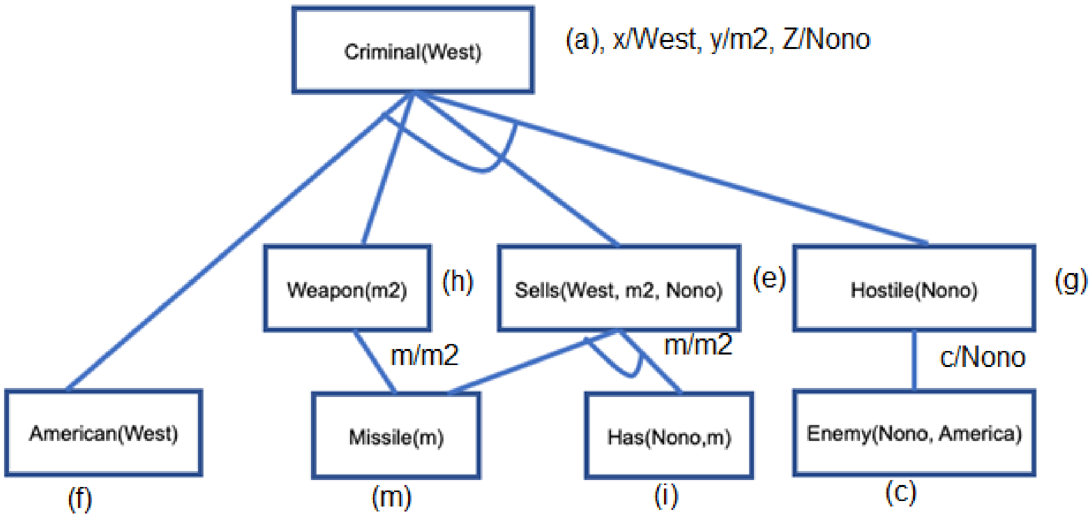
Step 1-



Step 2-



Step 3-



4. Translate the following set of sentences into FOL. Use forward chaining to prove if Marcus was loyal to Caesar.
   a. Marcus was a man.
   b. Marcus was a Pompeian.
   c. All Pompeian were Romans.
   d. Caesar was a ruler.
   e. All Romans were either loyal to Caesar or hated him.
   f. Everyone is loyal to someone.
   g. People only try to assassinate rulers they aren't loyal to.
   h. Marcus tried to assassinate Caesar.

FOL Representation -

a. Marcus was a man.
   Man(Marcus)
b. Marcus was a Pompeian.
   Pompeian(Marcus)
c. All Pompeian were Romans.
   $\forall$x: Pompeian(x) $\rightarrow$ Roman(x)
d. Caesar was a ruler.
   Ruler(Caesar)
e. All Romans were either loyal to Caesar or hated him.
   $\forall$x: Roman(x) $\rightarrow$ LoyalTo(x,Caesar) $\cup$ Hate(x,Caesar)
   Note- The above representation is inclusive-OR which is supported by English Language.
   There is another possible interpretation of the sentence considering the Exclusive-OR representation meaning as-

$\forall$x: Roman(x) $\rightarrow$ [(LoyalTo(x,Caesar) $\cup$ Hate(x,Caesar)) $\cap$ ~ (loyalto(x, Caesar) $\cap$ hate(x, Caesar))]

f. Everyone is loyal to someone.
   $\forall x: \exists y: LoyalTo(x,y)$
g. People only try to assassinate rulers they aren't loyal to. This is again an ambiguous statement which may infer multiple meanings. Is it 'the only thing people do in such cases is try to assassinate' or it should be taken that 'people just try to assassinate but don't actually assassinate rulers'?
   Anyway, let's consider one of the meanings and represent the sentence as-
   $\forall x: \forall y: Person(x) \cap Ruler(y) \cap TryAssassinate(x,y) \rightarrow \neg LoyalTo(x,y)]$
h. Marcus tried to assassinate Caesar.
   TryAssassinate(Marcus, Caesar)

   Additional Knowledge needed to answer the query-
i. All men are people.
   $\forall x: Man(x) \rightarrow Person(x)$

Forward chaining –

From (a), (i), Generalized Modus Ponens, {x/ Marcus}

j. Person(Marcus)

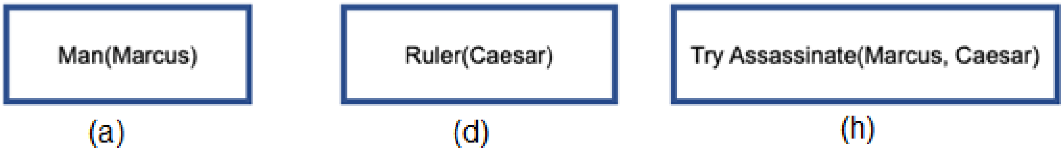
From (j), (d), (h), (g), Generalized Modus Ponens, {x/ Marcus, y/ Caesar}

k. ~LoyalTo(Marcus, Caesar)

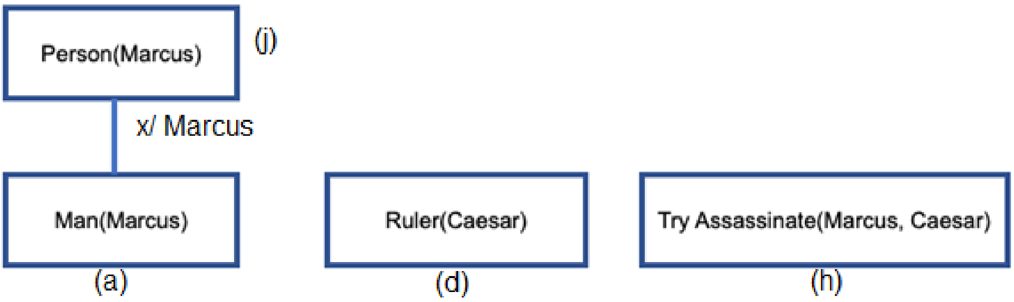
Hence Proved.

And-OR Graph Solution

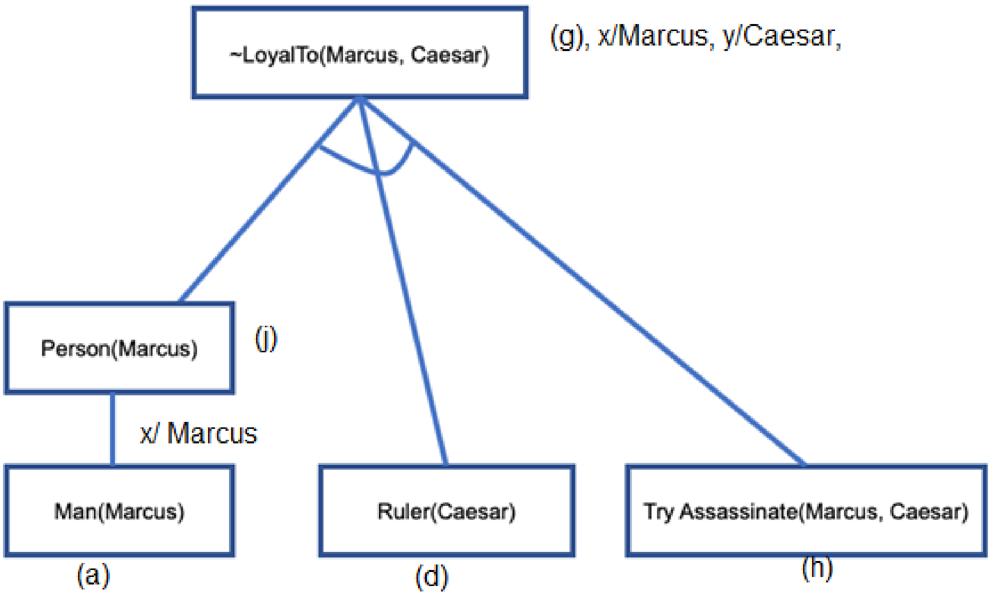
Step 1-



Step 2-



Step 3-



Backward Chaining

The forward chaining being the data driven approach, sometimes processes unnecessary facts and thereby wastes processing cycles and time. On the contrary, the backward chaining is goal oriented approach and works only on the relevant facts and rules. It scans through the list of inference rules until it finds the one who's THEN clause matches a desired goal. If the IF clause of that inference rule is not known to be true, then it is added to the list of goals. This way the approach starts with the desired goal and works recursively backwards to find supporting facts.

Steps for backward chaining-
The backward chaining sort of maintains a stack of goals. Initially the stack contains the conclusion to be proved. This stack is processed until it is empty to solve the query as-
1. Scan conclusions of all the rules to locate the ones who's conclusions match the topmost goal on the stack.
2. For every such rule found- Evaluate the condition(s) in the IF clause of the rule one at a time: If the condition is currently unknown, push it to goal-stack and

recursively invoke the system; elseif the condition is known to be satisfied, add the rule to the KB.
3. If all the conditions in the selected rule are satisfied, then pop the goal off the stack. Repeat step 2 and 3 till the stack is not empty or there are no more new rules to be matched.

The process terminates with success when the goal stack is empty. It terminates with failure if the algorithm runs out of rules to try in Step 2.

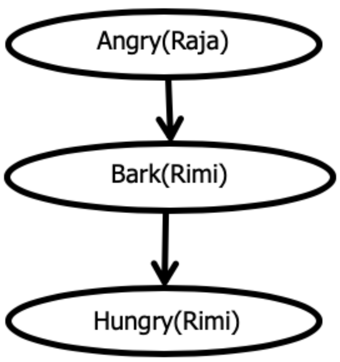
Examples - Reasoning with Backward Chaining

1. Translate the following set of sentences into FOL. Use backwad chaining to prove Raja is angry.
    a. Rimi is hungry
    b. if Rimi is hungry, she barks.
    c. if Rimi is barking then Raja is angry.

    FOL representation-
    a. Rimi is hungry.
       Hungry(Rimi)
    b. if Rimi is hungry, she barks.
       Hungry(Rimi) → Bark(Rimi)
    c. if Rimi is barking then Raja is angry.
       Bark(Rimi) → Angry(Raja)

    Backward Chaining-
       From (c),
    d. Bark(Rimi)
       From (b),
    e. Hungry(Rimi) → Given
    Hence Proved.

    And-OR graph- As the KB does not contain any variables, there is no substitution.



Hence Proved.

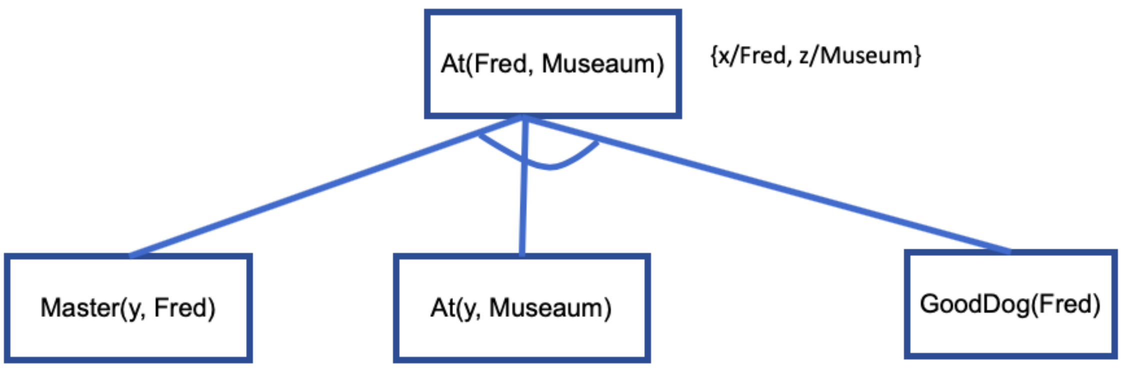
2. Translate the following set of sentences into FOL. Use backward chaining to prove Fred is at museum.
    a. Fred is a collie.
    b. Sam is Fred's master.
    c. The day is Saturday
    d. It is cold on Saturday.
    e. Fred is trained.
    f. Trained collies are good dogs.
    g. If a dog is a good dog and has a master, then he will be with his master.
    h. If the day is Saturday and the day is cold, the Sam is at the museum.

FOL Representation-

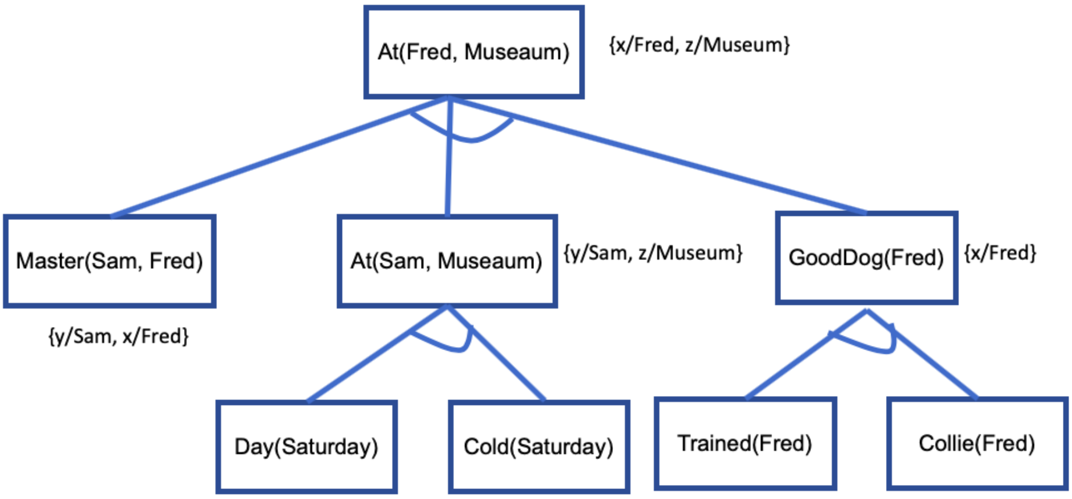
a. Fred is a collie.
   Collie(Fred)
b. Sam is Fred's master.
   Master(Sam, Fred)
c. The day is Saturday
   Day(Saturday)
d. It is cold on Saturday.
   Cold(Saturday)
e. Fred is trained.
   Trained(Collie)
f. Trained collies are good dogs.
   $\forall$ x Trained(x) $\cap$ Collie(x) → GoodDog(x)
g. Good dogs who have a master, then they are always with their master.
   $\forall$ x,y,z GoodDog(x) $\cap$ Master(y,x) $\cap$ At(y,z) → At(x,z)
h. If the day is Saturday and the day is cold, the Sam is at the museum.
   Day(Saturday) $\cap$ Cold(Saturday) → At(Sam, Museaum)

Backward Chaining solution with And-OR graph -

Step 1-



Step 2-



Hence Proved.

3. Translate the following set of sentences into FOL. Use backward chaining to prove Colonel West is a criminal.

   The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

   FOL Representation-

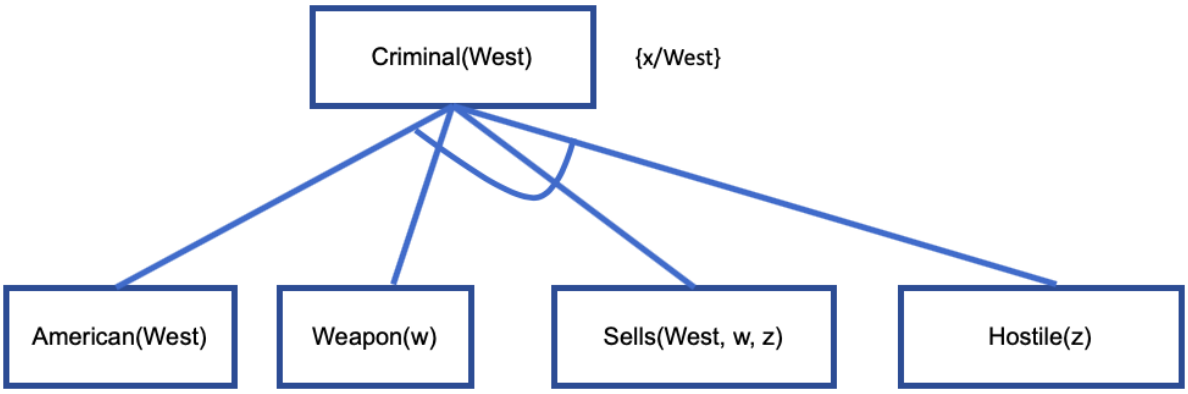
   a. ∀ x,y,z American(X) ∩ Weapon(y) ∩ HostileNation(z) ∩ Sell(x,y,z) → Criminal(x)
   b. Country(Nono)
   c. Enemy(Nono, America)
   d. ∃ m Has(Nono,m) ∩ Missile(m)
   e. ∀ m1 Missile(m1) ∩ Has(Nono,m1) → Sell(West, m1, Nono)
   f. American(West)

      The given statements lack some knowledge that is essential to prove the given query. The additional knowledge with their FOL representation go as-

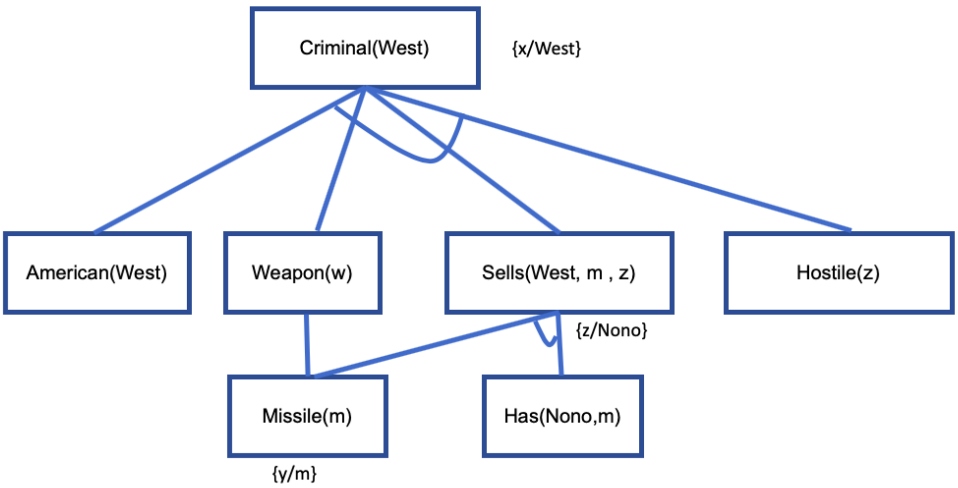
   g. Countries those are enemy of America are hostile countries.
      ∀ c Country(c) ∩ Enemy(c, America) → HostileNation(c)
   h. All missiles are weapons.
      ∀ m2 Missile(m2) → Weapon(m2)

Backward Chaining for this problem can best be illustrated with And-OR graph

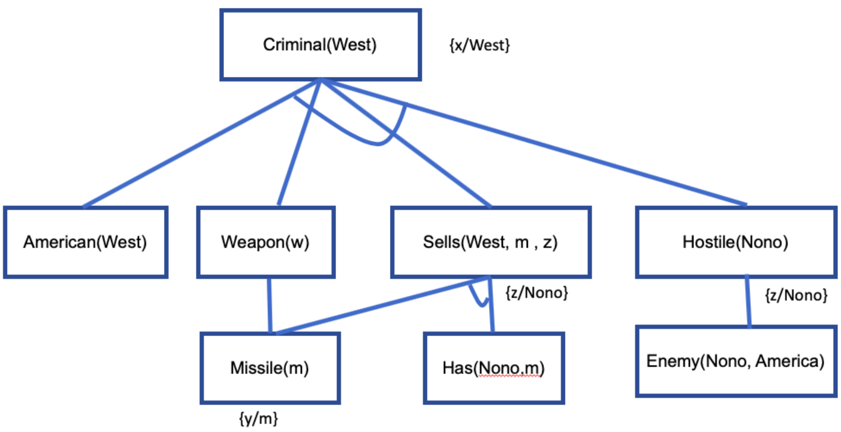
Step 1-



Step 2-



Step 3-



Hence Proved.

4. Translate the following set of sentences into FOL. Use backward chaining to prove if Marcus was loyal to Caesar.
   a. Marcus was a man.
   b. Marcus was a Pompeian.
   c. All Pompeian were Romans.
   d. Caesar was a ruler.
   e. All Romans were either loyal to Caesar or hated him.
   f. Everyone is loyal to someone.
   g. People only try to assassinate rulers they aren't loyal to.
   h. Marcus tried to assassinate Caesar.

FOL Representation -

a. Marcus was a man.
   Man(Marcus)
b. Marcus was a Pompeian.
   Pompeian(Marcus)
c. All Pompeian were Romans.
   $\forall$x: Pompeian(x) $\rightarrow$ Roman(x)
d. Caesar was a ruler.
   Ruler(Caesar)
e. All Romans were either loyal to Caesar or hated him.
   $\forall$x: Roman(x) $\rightarrow$ LoyalTo(x,Caesar) $\cup$ Hate(x,Caesar)

   Note- The above representation is inclusive-OR which is supported by English Language.

   There is another possible interpretation of the sentence considering the Exclusive-OR representation meaning as-

   $\forall$x: Roman(x) $\rightarrow$ [(LoyalTo(x,Caesar) $\cup$ Hate(x,Caesar)) $\cap$ ~(loyalto(x, Caesar) $\cap$ hate(x, Caesar))]

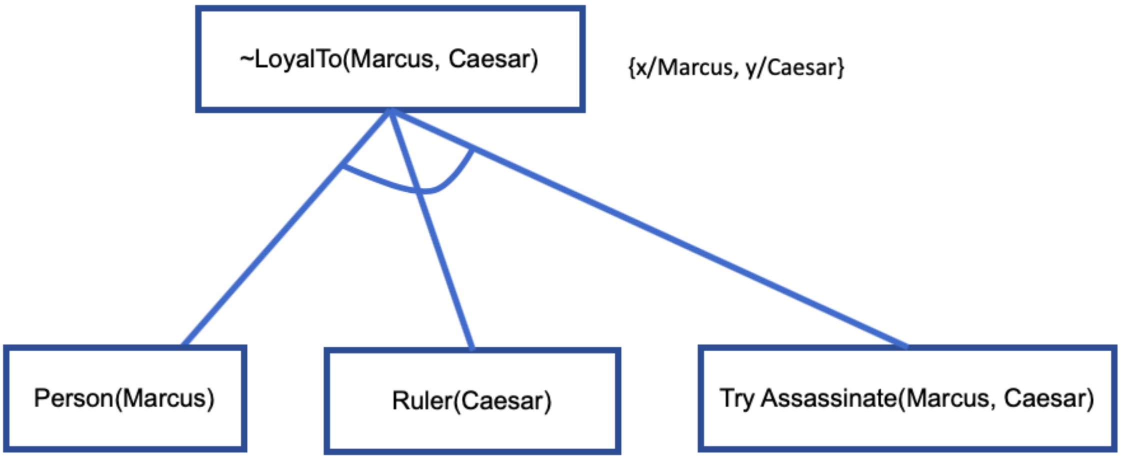
f. Everyone is loyal to someone.
   $\forall$x: $\exists$y: LoyalTo(x,y)
g. People only try to assassinate rulers they aren't loyal to.
   This is again an ambiguous statement which may infer multiple meanings. Is it 'the only thing people do in such cases is try to assassinate' or it should be taken that 'people just try to assassinate but don't actually assassinate rulers'?
   Anyway, let's consider one of the meanings and represent the sentence as-
   $\forall$x: $\forall$y: Person(x) $\cap$ Ruler(y) $\cap$ TryAssassinate(x,y) $\rightarrow\neg$LoyalTo(x,y)]
h. Marcus tried to assassinate Caesar.
   TryAssassinate(Marcus, Caesar)

Additional Knowledge needed to answer the query-

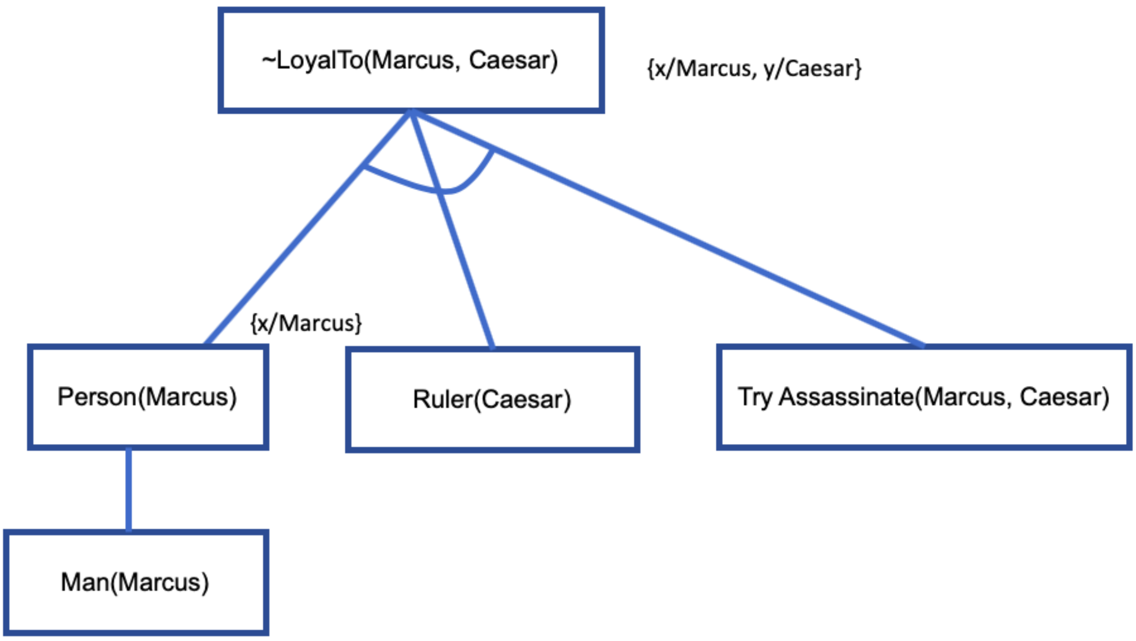
i. All men are people.
   $\forall x$: Man(x) $\rightarrow$ Person(x)

Backward chaining solution using And-OR graph –

Step 1-



Step 2-



Hence Proved.

Resolution

Resolution is an inference rule that uses proof by negation for proving the proposition. It is based on the concept that (KB ∩ a) can be proved true if (KB ∩ ~a) is contradiction. Resolution can be applied to propositional as well as first order logic. The process of resolution required the KB to be represented in Conjunctive Normal Form(CNF) which in simple words is conjunction of all disjuncts. As the inference answers the query by using negation of the goal, resolution is also called as proof by refutation and proof by contradiction.

Resolution is a complete inference process and it uses horn clauses, i.e. a sentence contains at the most only one positive literal.

Conversion to Conjunctive Normal Form

As a first step towards conversion into Conjunctive Normal Form, the given knowledge base is first presented into the FOL. These FOL statements are processed with following rules to convert them into CNF.

1. Eliminate implications.

Eliminate bidirectional implication by applying Bidirectional-Elimination followed by And-Elimination. Single directional implication is eliminated by using its logical equivalent as A→ B ≡ ~ A ∪ B.

Example Statement- Dead(x, t) ↔ ~Alive(x, t)

➔ Apply Bidirectional-Elimination to gain

(Dead(x, t) → ~Alive(x, t)) ∩ (~Alive(x, t) → Dead(x, t))

➔ Apply And-Elimination as,

a. Dead(x, t) → ~Alive(x, t)
b. ~Alive(x, t) → Dead(x, t)

➔ Remove implication by converting the statement into logically equivalent implication free Solution CNF form as-

a. ~Dead(x, t) ∪ ~Alive(x, t)
b. ~(~Alive(x, t)) ∪ Dead(x, t)

i.e. Alive(x, t) ∪ Dead(x, t)

2. Apply ~ to every single term.

Move the negation inwards by following the common rules for negated connectives and rules for negated quantifiers. The rules for negated quantifiers are –

a. $\sim \forall x\ p(x)$ is equivalent to $\exists x \sim p(x)$
b. $\sim \exists x\ p(x)$ is equivalent to $\forall x \sim p(x)$

Example 1- 'Every boy or girl is a child'

$\forall$ x Boy(x) $\cup$ Girl(x) → Child(x)

➔ Remove implication $\forall$ ~(Boy(x) $\cup$ Girl(x)) $\cup$ Child(x)

➔ Apply negation to every single term $\forall$ [(~Boy(x) $\cap$ ~Girl(x)) $\cup$ Child(x)]

Example 2- No Indian likes dosa.

~ $\exists$ x Indian(x) $\cap$ Likes(x,Dosa)

➔ $\forall$x ~(Indian(x) $\cap$ likes(x,Dosa))

➔ $\forall$x ~Indian(x) $\cup$ ~likes(x,Dosa)

3. Standardize variables
   To avoid confusion during substitutions, every statement should use a different variable. Though the initial KB might have used different variable in each sentence, application of earlier rules may create multiple sentences out of one, making same variable appear in more than one sentence.
   e.g. One of the following sentences needs a new variable as-

   a. ~Dead(x, t) $\cup$ ~Alive(x, t)
   b. Alive(x, t) $\cup$ Dead(x, t) ➔ Alive(x_1, t) $\cup$ Dead(x_1, t)

4. Eliminate existential quantifier.
   This step is also known as Skolemization. The simplest way of Skolemization is to replace the variable with a new constant by following rule that $\exists$ x p(x) $\equiv$ p(M) where M is a new constant.
   e.g.
   a. $\exists$ x Indian(x) is converted as ➔ Indian(Rahul)
   b. Everyone is loyal to someone
   ➔ $\forall$x $\exists$y LoyalTo(x,y)
   ➔ $\forall$x LoyalTo(x,f(x))

5. Drop universal quantifier

   By this step all remaining variables are universally quantified, i.e. statement is always true in the given world. So the universal quantifiers can be easily dropped.

   e.g. $\forall$x LoyalTo(x,f(x)) gets converted as LoyalTo(x,f(x))

6. Distribute $\cup$ over $\cap$.

Distribute nested conjunctions and disjunctions. If needed, standardise the variables again so that every statement in KB has a different variable.
e.g. Earlier statement from step 2 gets processed as,

$\forall$ [(~Boy(x) $\cap$ ~Girl(x)) $\cup$ Child(x)]

➔ Drop universal quantifier [(~Boy(x) $\cap$ ~Girl(x)) $\cup$ Child(x)]

➔ Distribute $\cup$ over $\cap$ as ((~Boy(x) $\cup$ Child(x)) $\cap$ (~Girl(x)) $\cup$ Child(x))
➔ Apply And-Elimination and change variable in one of the statements as,
a. ~Boy(x) $\cup$ Child(x)
b. ~Girl(x)) $\cup$ Child(x) ➔ ~Girl(x_1)) $\cup$ Child(x_1)

Resolution inference rule

The resolution rule for first-order logic works exactly like it works for propositional logic in a kind of lifted fashion. Two clauses that are containing no common variables can be resolved if they contain complementary literals.

e.g. TryAssassinate(x,y) is a complimentary literal for ~TryAssassinate(Marcus,Caesar) when applied with the unification process with {x/ Marcus, y/Caesar} as substitution.

The resolution inference can be mathematically defined as,

$$\frac{[(l1 \cup l2 \cup \ldots \cup lk), (m1 \cup m2 \cup \ldots mn)]}{SUBSTITUTION(\theta, l1 \cup \ldots \cup li-1 \cup li+1 \cup \ldots \cup lk \cup m1 \cup \ldots \cup mj-1 \cup mj+1 \cup \ldots \cup mn)}$$

Assuming li and mj are complementary literals, they cancel each other and rest of the literals from both the statements.

Examples - Reasoning with Resolution

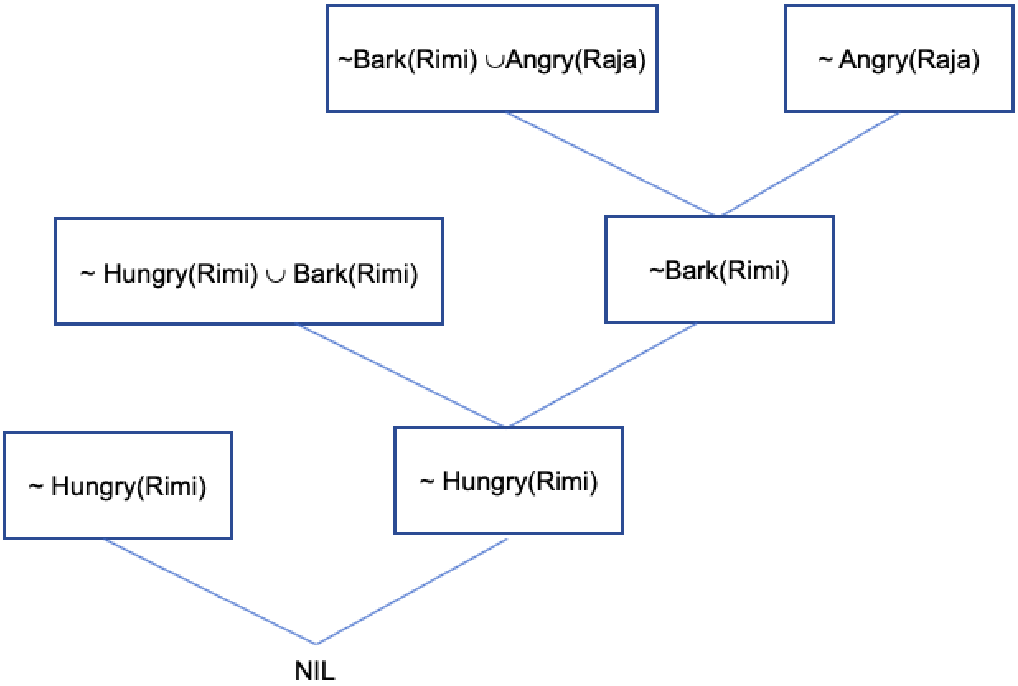
1. Translate the following set of sentences into CNF. Use resolution to prove Raja is angry.
   a. Rimi is hungry
   b. if Rimi is hungry, she barks.
   c. if Rimi is barking then Raja is angry.

   CNF representation-
   f. Rimi is hungry.
   FOL➔ Hungry(Rimi)
   CNF➔ Hungry(Rimi)
   g. if Rimi is hungry, she barks.
   FOL➔Hungry(Rimi)➔ Bark(Rimi)
   CNF➔ ~ Hungry(Rimi) $\cup$ Bark(Rimi)
   h. if Rimi is barking then Raja is angry.

FOL→ Bark(Rimi) → Angry(Raja)
CNF→ ~Bark(Rimi) ∪ Angry(Raja)

Resolution-

a. Negate the goal - ~ Angry(Raja)
b. Prove (KB ∩ ~Goal) is a contradiction.



The contradicting statements left no clause in KB, i.e. (KB ∩ ~Goal) is a contradiction. Hence negation of goal is false, which proves that goal statement was correct.

2. Translate the following set of sentences into CNF. Use resolution to prove Fred is at museum.
    a. Fred is a collie.
    b. Sam is Fred's master.
    c. The day is Saturday
    d. It is cold on Saturday.
    e. Fred is trained.
    f. Trained collies are good dogs.
    g. If a dog is a good dog and has a master, then he will be with his master.
    h. If the day is Saturday and the day is cold, the Sam is at the museum.

CNF Representation-

a. Fred is a collie.
FOL➔ Collie(Fred)
CNF➔ Collie(Fred)

b. Sam is Fred's master.
FOL➔Master(Sam, Fred)
CNF➔ Master(Sam, Fred)

c. The day is Saturday
FOL➔ Day(Saturday)
CNF➔ Day(Saturday)

d. It is cold on Saturday.
FOL➔ Cold(Saturday)
CNF➔ Cold(Saturday)

e. Fred is trained.
FOL➔ Trained(Collie)
CNF➔ Trained(Collie)

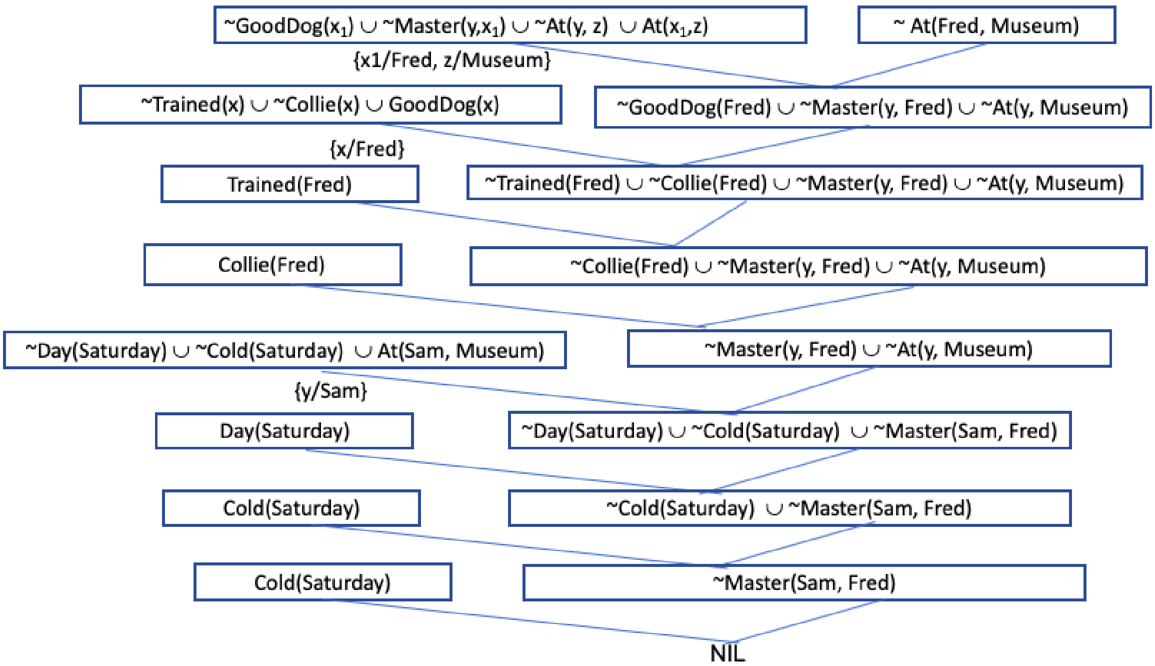
f. Trained collies are good dogs.
FOL➔ $\forall$ x Trained(x) $\cap$ Collie(x) → GoodDog(x)
CNF➔$\forall$ x ~(Trained(x) $\cap$ Collie(x)) $\cup$ GoodDog(x)
➔ $\forall$ x ~Trained(x) $\cup$ ~Collie(x) $\cup$ GoodDog(x)
➔ ~Trained(x) $\cup$ ~Collie(x) $\cup$ GoodDog(x)

g. Good dogs who have a master, then they are always with their master.
FOL➔ $\forall$ x,y,z GoodDog(x) $\cap$ Master(y,x) $\cap$ At(y,z) → At(x,z)
CNF➔$\forall$ x,y,z ~(GoodDog(x) $\cap$ Master(y,x) $\cap$ At(y,z)) $\cup$ At(x,z)
➔$\forall$ x,y,z ~GoodDog(x) $\cup$ ~Master(y,x) $\cup$ ~At(y,z) $\cup$ At(x,z)
➔ ~GoodDog(x_1) $\cup$ ~Master(y,x_1) $\cup$ ~At(y,z) $\cup$ At(x_1,z)

h. If the day is Saturday and the day is cold, the Sam is at the museum.
FOL➔ Day(Saturday) $\cap$ Cold(Saturday) → At(Sam, Museaum)
CNF➔ ~(Day(Saturday) $\cap$ Cold(Saturday)) $\cup$ At(Sam, Museaum)
➔ ~Day(Saturday) $\cup$ ~Cold(Saturday) $\cup$ At(Sam, Museaum)

Resolution-

a. Negate the goal - ~ At(Fred, Museum)
b. Prove (KB $\cap$ ~Goal) is a contradiction.



The contradicting statements left no clause in KB, i.e. (KB ∩ ~Goal) is a contradiction. Hence negation of goal is false, which proves that goal statement was correct.

3. Translate the following set of sentences into CNF. Use resolution to prove Colonel West is a criminal.

   The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

   CNF Representation-

   a. FOL➔ ∀ x,y,z American(X) ∩ Weapon(y) ∩ HostileNation(z) ∩ Sell(x,y,z) → Criminal(x)
      CNF➔ ~ American(X) ∪ ~Weapon(y) ∪ ~HostileNation(z) ∪ ~Sell(x,y,z) ∪ Criminal(x)

   b. FOL and CNF➔ Country(Nono)
   c. FOL and CNF➔ Enemy(Nono, America)
   d. FOL➔ ∃ m Has(Nono,m) ∩ Missile(m)
      CNF➔ d1. Has(Nono,m)
      ➔ d2. Missile(m3)

   e. FOL➔ ∀ m1 Missile(m1) ∩ Has(Nono,m1) → Sell(West, m1, Nono)
      CNF➔ ~Missile(m1) ∪ ~Has(Nono,m1) ∪ Sell(West, m1, Nono)

   f. FOL and CNF➔ American(West)

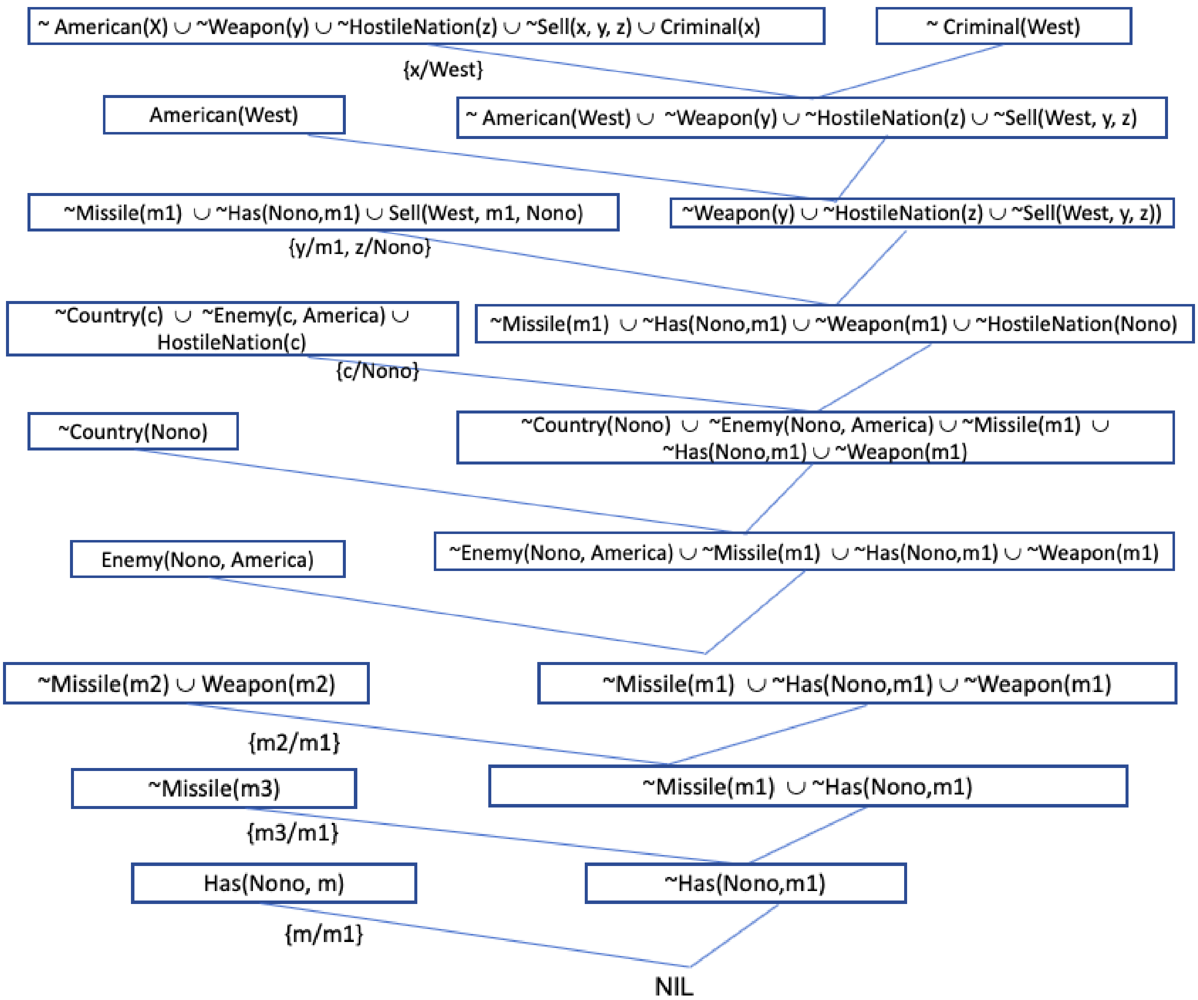
The given statements lack some knowledge that is essential to prove the given query. The additional knowledge with their FOL representation go as-

g. Countries those are enemy of America are hostile countries.
FOL➔∀ c Country(c) ∩ Enemy(c, America) → HostileNation(c)
CNF➔ ~Country(c) ∪ ~Enemy(c, America) ∪ HostileNation(c)

h. All missiles are weapons.
FOL➔ ∀ m2 Missile(m2) → Weapon(m2)
CNF➔ ~Missile(m2) ∪ Weapon(m2)

Resolution-

a. Negate the goal - ~ Criminal(West)
b. Prove (KB ∩ ~Goal) is a contradiction.



5. Translate the following set of sentences into CNF. Use resolution to prove if Marcus was loyal to Caesar.
    a. Marcus was a man.
    b. Marcus was a Pompeian.

c. All Pompeian were Romans.
d. Caesar was a ruler.
e. All Romans were either loyal to Caesar or hated him.
f. Everyone is loyal to someone.
g. People only try to assassinate rulers they aren't loyal to.
h. Marcus tried to assassinate Caesar.

FOL Representation -

a. Marcus was a man.
   FOL and CNF➔ Man(Marcus)
b. Marcus was a Pompeian.
   FOL and CNF➔ Pompeian(Marcus)
c. All Pompeian were Romans.
   FOL➔ $\forall$x: Pompeian(x) $\rightarrow$ Roman(x)
   CNF➔ ~ Pompeian(x_1) $\cup$ Roman(x_1)
d. Caesar was a ruler.
   FOL and CNF➔ Ruler(Caesar)
e. All Romans were either loyal to Caesar or hated him.
   FOL➔ $\forall$x: Roman(x) $\rightarrow$ LoyalTo(x,Caesar) $\cup$ Hate(x,Caesar)
   CNF➔ ~ Roman(x_2) $\cup$ LoyalTo(x_2,Caesar) $\cup$ Hate(x_2,Caesar)
f. Everyone is loyal to someone.
   FOL➔ $\forall$x: $\exists$y: LoyalTo(x,y)
   CNF➔ LoyalTo(x_3,f(x_3))

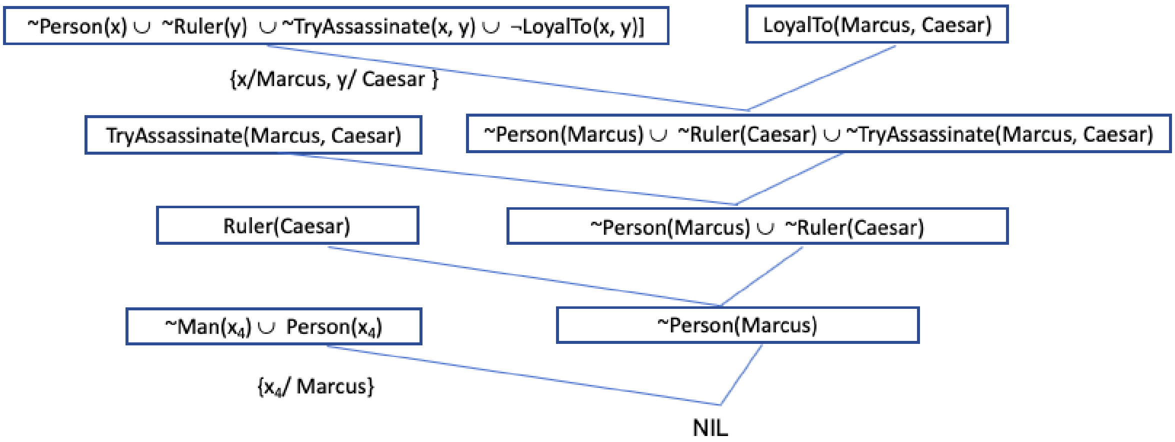
g. People only try to assassinate rulers they aren't loyal to.
   This is again an ambiguous statement which may infer multiple meanings. Is it 'the only thing people do in such cases is try to assassinate' or it should be taken that 'people just try to assassinate but don't actually assassinate rulers'?
   Anyway, let's consider one of the meanings and represent the sentence as-
   FOL➔ $\forall$x: $\forall$y: Person(x) $\cap$ Ruler(y) $\cap$ TryAssassinate(x,y) $\rightarrow\neg$LoyalTo(x,y)]
   CNF➔ ~Person(x) $\cup$ ~Ruler(y) $\cup$ ~TryAssassinate(x,y) $\cup$ $\neg$LoyalTo(x,y)]

h. Marcus tried to assassinate Caesar.
   FOL and CNF➔ TryAssassinate(Marcus, Caesar)

   Additional Knowledge needed to answer the query-
i. All men are people.
   FOL➔ $\forall$x: Man(x) $\rightarrow$ Person(x)
   CNF➔ ~Man(x_4) $\cup$ Person(x_4)

Resolution-

c. Negate the goal – LoyalTo(Marcus, Caesar)
d. Prove (KB $\cap$ ~Goal) is a contradiction.



Summary

- First order logic lifts some inference methods from propositional logic to draw the inferences. Apart from those rules, FOL uses universal instantiation and existential instantiation for reasoning.
- Unification is used to match two literals and infer new statement first by matching-unifying the literals and then by using some inference methods.
- The unification process is used by forward chaining, backward chaining and resolution.
- The process of reasoning can be explained step by step or by using And-OR graphs.
- Forward chaining is best suitable for exploration environments where the data driven approach is used for reasoning.
- Forward chaining is comparatively slow process and may process irrelevant literals in an attempt to answer the given query.
- Forward chaining matches the rules on the left-hand side of the implication and recursively prove the implied literals to prove the given conclusion.
- Backward chaining starts with goal and proves that knowledge base plus goal is not a contradiction.
- Backward chaining is goal driven approach and takes less processing cycles compared to forward chaining.
- Resolution is an inference rule that uses proof by negation to prove the conclusion. It works on the concept that knowledge base + negation of goal is a contradiction. i.e. negation of goal is false, in other words- the goal is true.
- Resolution process converts the natural language sentences into FOL and then in the Conjunctive Normal Form; to resolve the complimenting literals.
- The correct resolution process cancels out each other and leave no literals in the knowledge base proving it as a contradiction.

Test your knowledge

Multiple choice questions

1. ________ is the process that captures inference process as a single inference rule?
   a) Law of contrapositive
   b) Quantifiers
   c) **Generalized Modus Ponens**
   d) Variables and constants

2. Which of the following operation checks if any two logical statements are identical?

   a. Generalized modus ponens
   b. Equivalence, validity and satisfiability
   c. Unification
   d. Forward chaining

3. Forward chaining and backward chaining do not need logical expressions be converted into CNF. State true or false
   a. True
   b. False
4. State true or false- CNF contains disjuncts of conjuncts.
   a. True
   b. False
5. State true or false-In a backward chaining system, reasoning process starts with the initial set of facts and recursively use the rules to infer new conclusions until goal or contradiction isn’t proved.
   a. True
   b. False
6. State true or false-In a forward chaining system, reasoning process starts with the initial set of facts and recursively use the rules to infer new conclusions until goal or contradiction isn’t proved.
   a. True
   b. False
7. Forward chaining is _____ oriented, while backward chaining is _____ oriented.
   a. Goal, Data
   b. Data, Data
   c. Data, Goal
   d. None of the above
8. A Horn clause is a clause with _______ positive literal(s).
   a. Exactly one
   b. **At the most one**
   c. Zero
   d. All
9. Backward chaining uses-
   a) Conjuncts and disjuncts
   b) Negation

c) **List of successful substitutions**
d) None of the mentioned

10. Resolution is also known by
    a. Proof by negation
    b. Proof by refutation
    c. Proof by contradiction
    d. All of the above
    e. None of the above.

Theoretical questions.

Questions

1. Translate the following set of sentences into FOL. Use forward chaining to prove Butler got the cream
    a. If maid stole the jewelry, then butler was not guilty
    b. Either maid stole the jewelry or she milked the cow.
    c. If maid milked the cow, then butler got the cream.
    d. Butler was guilty
2. Translate the following set of sentences into FOL. Use backward chaining to prove Butler got the cream
    a. If maid stole the jewelry, then butler was not guilty
    b. Either maid stole the jewelry or she milked the cow.
    c. If maid milked the cow, then butler got the cream.
    d. Butler was guilty
3. Translate the following set of sentences into CNF. Use Resolution to prove Curiosity kill the cat.
    a. Everyone who loves all animals is loved by someone.
    b. Anyone who kills an animal is loved by no one.
    c. Jack loves all animals.
    d. Either Jack or Curiosity killed the cat, who is named Tuna.
4. Translate the following set of sentences into FOL. Use backward chaining to prove John does not have any mice.
    a. All hounds howl at night.
    b. Anyone who has any cats will not have any mice.
    c. Light sleepers do not have anything which howls at night.
    d. John has either a cat or a hound.
    e. John is a light sleeper.
    f.
5. Translate the following set of sentences into FOL. Use forward, backward chaining and resolution to prove every student who is a CS major and makes good grades is brilliant.
    a. Every student who makes good grades is brilliant or studies.
    b. Every student who is a CS major has some roommate.
    c. Every student who has any roommate who likes to party goes to Sixth Street.
    d. Anyone who goes to Sixth Street does not study.
    e. Every roommate of every CS major likes to party.