Chapter- 4

Knowledge Representation

- The objective of knowledge representation is to express the knowledge about the world in a computer-tractable form
- Key aspects of knowledge representation languages are:
  i. Syntax: describes how sentences are formed in the language.
  ii. Semantics: describes the meaning of sentences, what is it the sentence refers to in the real world.
  iii. Computational aspect: describes how sentences and objects are manipulated in concordance with semantically conventions.
- A *formal language* is required o represent knowledge in a computer tractable form and r*easoning* processes are required to manipulate this knowledge to deduce non-obvious facts.

Knowledge Representation using Logic

- Logic is defined as a formal language for expressing knowledge and ways of reasoning.
- Logic makes statements about the world which are true (or false).
- Compared to natural languages (expressive but context sensitive) and programming languages (good for concrete data structures but not expressive) logic combines the advantages of natural languages and formal languages.
- Logic is:
  - concise
  - unambiguous
  - context insensitive
  - expressive
  - effective for inferences
- A logic is defined by the following:
  1. ***Syntax*** - describes the possible configurations that constitute sentences.
  2. ***Semantics*** - determines what facts in the world the sentences refer to i.e. the interpretation. Each sentence makes a claim about the world.
  3. ***Proof theory*** - set of rules for generating new sentences that are necessarily true given that the old sentences are true. The relationship between sentences is

called **entailment**. The semantics link these sentences (representation) to facts of the world. The proof can be used to determine new facts which follow from the old.

4. ***A set of sentences*** – A sentence is constructed from a set of primitives according to syntax rules.

5. ***A set of interpretations*** – An interpretation gives a semantic to primitives. It associates primitives with values.

- ***The valuation (meaning) function*** – Assigns a value (typically the truth value) to a given sentence under some interpretation. sentence × interpretation → {True , False }

- Knowledge representation using logic shows:
  - i. How logic can be used to form representations of the world?
  - ii. How a process of inference can be used to derive new representations about the world?
  - iii. How these can be used by an intelligent agent to deduce what to do.

- **Types of logic:** Different types of logics are: Propositional logic and First-order logic (First –order Predicate Logic).

Propositional logic

- A propositional logic is a declarative sentence which can be either true or false but not both or either.
- Propositional logic is a mathematical model that allows us to reason about the truth or falsehood of logical expression.
- In propositional logic, there are atomic sentences and compound sentences built up from atomic sentences using logical connectives.

A set of connectives:

$\wedge$	AND
$\vee$	OR
$\neg$	NOT
=>	implies
<=>	mutual implication

Sentences in the propositional logic:

i. ***Atomic sentences:*** – Constructed from constants and propositional symbols – True, False are (atomic) sentences, Light in the room is on, It rains outside are (atomic) sentences.

ii. ***Composite sentences:*** – Constructed from valid sentences via connectives eg: (A ∧ B) (A ∨ B) (A ⇒ B) (A ⇔ B) (A ∨ B) ∧ (A ∨ ¬ B)

- A sentence (well formed formula) is defined as follows:
  - A symbol is a sentence
  - If S is a sentence, then S is a sentence
  - If S is a sentence, then (S) is a sentence
  - If S and T are sentences, then (S ¬ T), (S ∨ T), (S ∧ T), and (S ↔ T) are sentences

A sentence results from a finite number of applications of the above rules

Truth Table in Propositional Logic

A	B	┐A	A V B	A ^ B	A => B	A ®B
T	T	F	T	T	T	T
T	F	F	T	F	F	F
F	T	T	T	F	T	F
F	F	T	F	F	T	T

Find Truth Table of following :

- A => (B => A)
- (A => B) => A
- A => ((B ^ C) V ┐A)
- $((A \wedge B) \rightarrow C) \Leftrightarrow (A \rightarrow (B \rightarrow C))$.

Tautology: Tautology is notation in formal logic which is always true (valid).
Contradiction (Unsatisfiable): notation in formal logic which is always false.
Satisfiable: If at least one sentence is true in the set.

Important Logical Equivalences: First List

The following logical equivalences apply to *any* statements; the p's, q's and r' s can stand for atomic statements or compound statements.

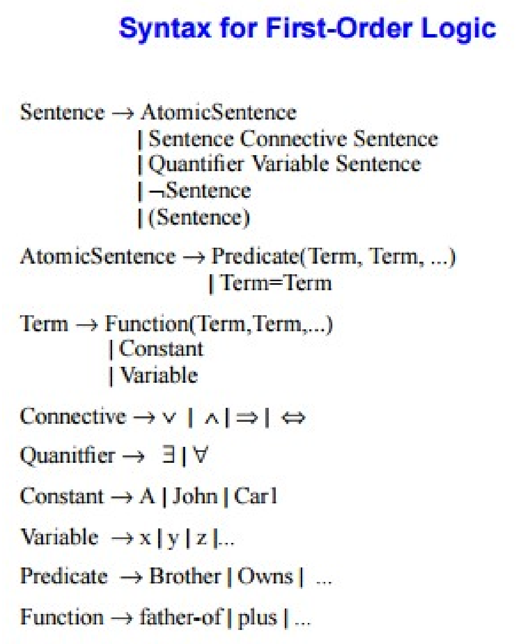
$\sim(\sim p) \equiv p$	**the Double Negative Law**
$\sim(p \vee q) \equiv (\sim p) \wedge (\sim q)$ $\sim(p \wedge q) \equiv (\sim p) \vee (\sim q)$	**De Morgan's Laws**
$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	**the Distributive Laws**

Inference rules in propositional logic:

Rule of inference	Tautology	Name
$p \to q$ $\underline{p \quad\quad}$ $\therefore q$	$[p \wedge (p \to q)] \to q$	Modus ponens
$\neg q$ $\underline{p \to q}$ $\therefore \neg p$	$[\neg q \wedge (p \to q)] \to \neg p$	Modus tollen
$p \to q$ $\underline{q \to r}$ $\therefore p \to r$	$[(p \to q) \wedge (q \to r)] \to (p \to r)$	Hypothetical syllogism
$p \vee q$ $\underline{\neg p \quad}$ $\therefore q$	$((p \vee q) \wedge \neg p) \to q$	Disjunctive syllogism
$\underline{p \quad\quad}$ $\therefore p \vee q$	$p \to (p \vee q)$	Addition
$\underline{p \wedge q}$ $\therefore p$	$(p \wedge q) \to p$	Simplification
p $\underline{q \quad\quad}$ $\therefore p \wedge q$	$((p) \wedge (q)) \to (p \wedge q)$	Conjunction
$p \vee q$ $\underline{\neg p \vee r}$ $\therefore q \vee r$	$[(p \vee q) \wedge (\neg p \vee r)] \to (p \vee r)$	Resolution

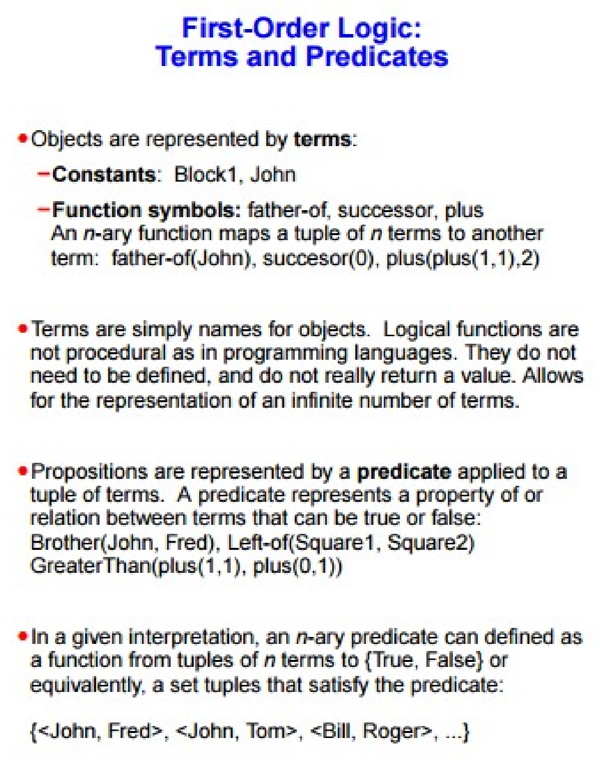
Predicate Logic

- Predicate logic is an extension of propositional logic with more expressive power.
- In propositional logic, each possible atomic fact requires a separate unique propositional symbol.
- If there are n people and m locations, representing the fact that some person moved from one location to another requires (nm^2) separate symbols.
- Predicate logic includes a richer ontology: -objects (terms) -properties (unary predicates on terms) -relations (n-ary predicates on terms) -functions (mappings from terms to other terms)
- Allows more flexible and compact representation of knowledge: Move(x, y, z) for person x moved from location y to z.
- Predicate Symbols are used to denote a property of objects or a relation between objects



Syntax for First-Order Logic

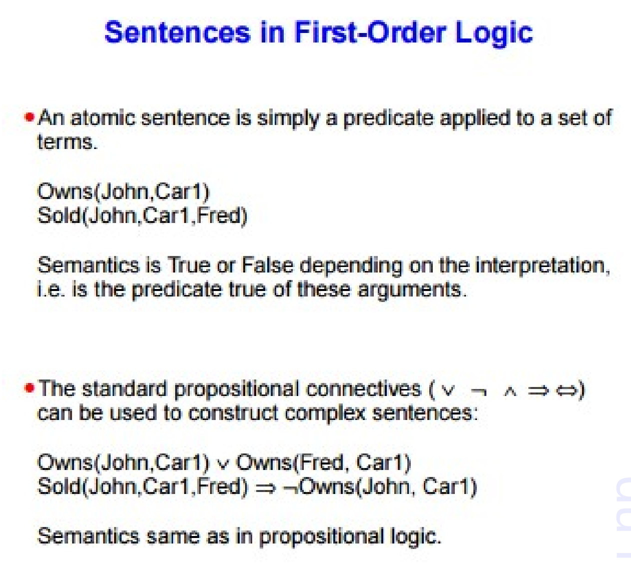
```
Sentence → AtomicSentence
           | Sentence Connective Sentence
           | Quantifier Variable Sentence
           | ¬Sentence
           | (Sentence)

AtomicSentence → Predicate(Term, Term, ...)
               | Term=Term

Term → Function(Term,Term,...)
     | Constant
     | Variable

Connective → ∨ | ∧ | ⇒ | ⇔

Quanitfier →  ∃ | ∀

Constant → A | John | Car1

Variable  → x | y | z |...

Predicate  → Brother | Owns |  ...

Function → father-of | plus | ...
```



First-Order Logic:
Terms and Predicates

- Objects are represented by **terms**:
  - **Constants**: Block1, John
  - **Function symbols:** father-of, successor, plus
    An *n*-ary function maps a tuple of *n* terms to another term: father-of(John), succesor(0), plus(plus(1,1),2)
- Terms are simply names for objects. Logical functions are not procedural as in programming languages. They do not need to be defined, and do not really return a value. Allows for the representation of an infinite number of terms.
- Propositions are represented by a **predicate** applied to a tuple of terms. A predicate represents a property of or relation between terms that can be true or false:
  Brother(John, Fred), Left-of(Square1, Square2)
  GreaterThan(plus(1,1), plus(0,1))
- In a given interpretation, an *n*-ary predicate can defined as a function from tuples of *n* terms to {True, False} or equivalently, a set tuples that satisfy the predicate:

  {<John, Fred>, <John, Tom>, <Bill, Roger>, ...}



Sentences in First-Order Logic

- An atomic sentence is simply a predicate applied to a set of terms.

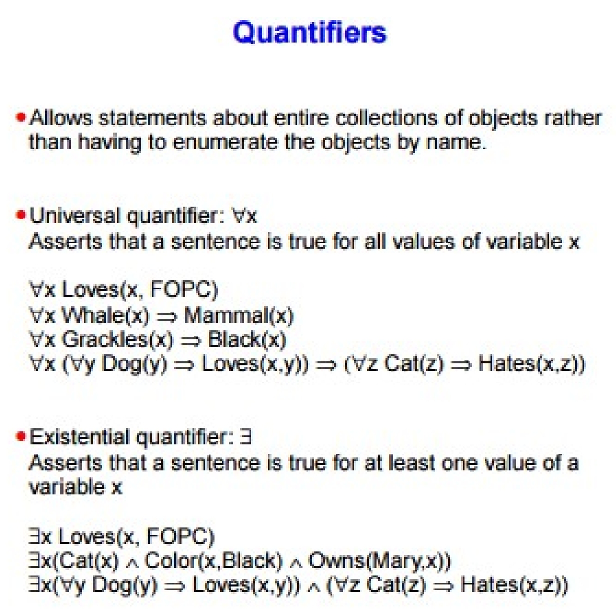
  $Owns(John,Car1)$
  $Sold(John,Car1,Fred)$

  Semantics is True or False depending on the interpretation, i.e. is the predicate true of these arguments.

- The standard propositional connectives ($\vee$ $\neg$ $\wedge$ $\Rightarrow$ $\Leftrightarrow$) can be used to construct complex sentences:

  $Owns(John,Car1) \vee Owns(Fred, Car1)$
  $Sold(John,Car1,Fred) \Rightarrow \neg Owns(John, Car1)$

  Semantics same as in propositional logic.



Quantifiers

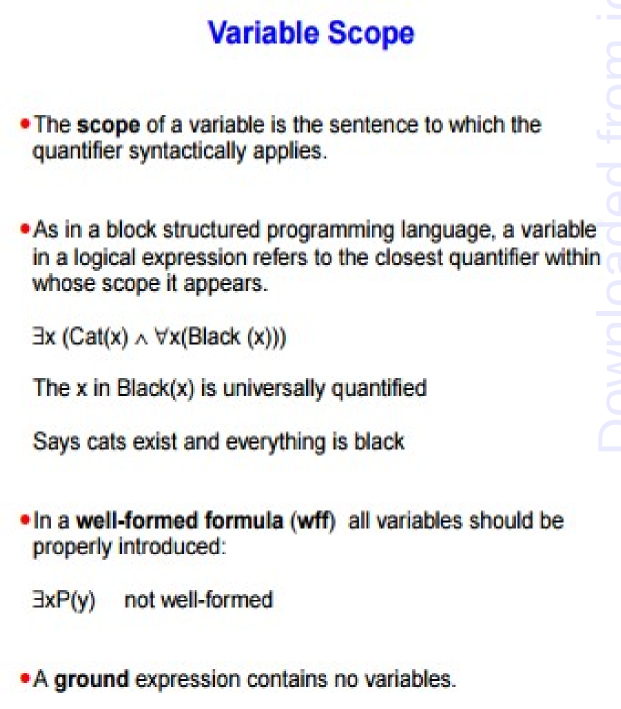
- Allows statements about entire collections of objects rather than having to enumerate the objects by name.

- Universal quantifier: $\forall x$
  Asserts that a sentence is true for all values of variable x

  $\forall x\ Loves(x, FOPC)$
  $\forall x\ Whale(x) \Rightarrow Mammal(x)$
  $\forall x\ Grackles(x) \Rightarrow Black(x)$
  $\forall x\ (\forall y\ Dog(y) \Rightarrow Loves(x,y)) \Rightarrow (\forall z\ Cat(z) \Rightarrow Hates(x,z))$

- Existential quantifier: $\exists$
  Asserts that a sentence is true for at least one value of a variable x

  $\exists x\ Loves(x, FOPC)$
  $\exists x(Cat(x) \wedge Color(x,Black) \wedge Owns(Mary,x))$
  $\exists x(\forall y\ Dog(y) \Rightarrow Loves(x,y)) \wedge (\forall z\ Cat(z) \Rightarrow Hates(x,z))$



Variable Scope

- The **scope** of a variable is the sentence to which the quantifier syntactically applies.

- As in a block structured programming language, a variable in a logical expression refers to the closest quantifier within whose scope it appears.

  $\exists x\ (Cat(x) \wedge \forall x(Black\ (x)))$

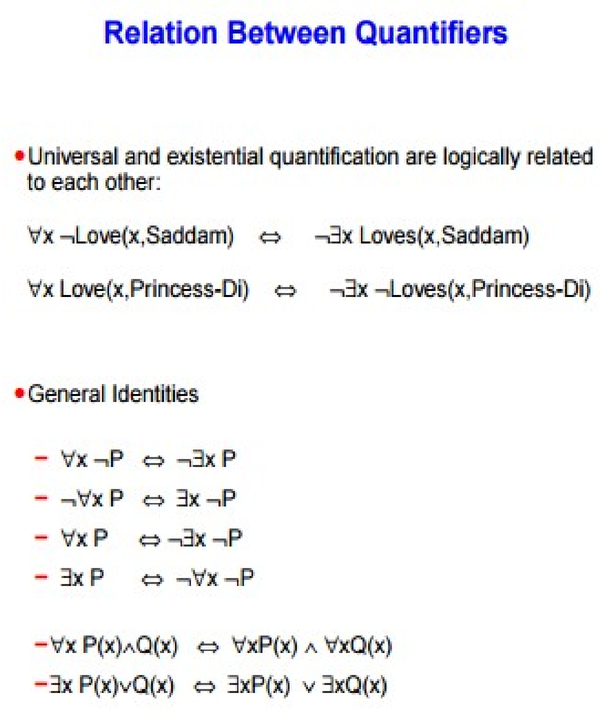
  The x in Black(x) is universally quantified

  Says cats exist and everything is black

- In a **well-formed formula** (**wff**) all variables should be properly introduced:

  $\exists xP(y)$ not well-formed

- A **ground** expression contains no variables.



Relation Between Quantifiers

- Universal and existential quantification are logically related to each other:

  $\forall x\ \neg Love(x,Saddam) \Leftrightarrow \neg\exists x\ Loves(x,Saddam)$

  $\forall x\ Love(x,Princess\text{-}Di) \Leftrightarrow \neg\exists x\ \neg Loves(x,Princess\text{-}Di)$

- General Identities

  - $\forall x\ \neg P \Leftrightarrow \neg\exists x\ P$
  - $\neg\forall x\ P \Leftrightarrow \exists x\ \neg P$
  - $\forall x\ P \Leftrightarrow \neg\exists x\ \neg P$
  - $\exists x\ P \Leftrightarrow \neg\forall x\ \neg P$

  - $\forall x\ P(x) \wedge Q(x) \Leftrightarrow \forall x P(x) \wedge \forall x Q(x)$
  - $\exists x\ P(x) \vee Q(x) \Leftrightarrow \exists x P(x) \vee \exists x Q(x)$

**** Refer Class Note for Fopl conversation, Inference rules in propositional logic, CNF conversion process and Proof by Resolution (RRS)**

Chapter-4

Rule-Based Systems

- Rule-based systems (RBS) provide automatic problem solving tools for capturing the human expertise and decision making.
- RBS are means for codifying the problem solving of human experts.
- Experts typically express most of their problem solving techniquses in terms of antecedent-consequent rules.
- The main properties of rule-based systems are:
  - They incorporate practical human knowledge in if-then rules;
  - Their skill increases proportionally to the enlargement of their knowledge;
  - They can solve a wide range of potentially complex problems by selecting relevant rules and then combining the results;
  - They adaptively determine the best sequence of rules to examine;
  - They explain their conclusions by retracting their lines of reasoning

Forward chaining

- Forward chaining is one of the two main methods of reasoning when using inference rules
- Described logically as repeated application of modus ponens.
- Forward chaining is a popular implementation strategy for expert systems, business and production rule systems.
- Forward chaining starts with the available data and uses inference rules to extract more data until a goal is reached.
- An inference engine using forward chaining searches the inference rules until it finds one where the antecedent (If clause) is known to be true. When such a rule is found, the engine can conclude, or infer, the consequent (Then clause), resulting in the addition of new information to its data.
- Inference engines will iterate through this process until a goal is reached.
- Because the data determines which rules are selected and used, this method is called data-driven.
- The forward chaining approach is often employed by expert systems, such as CLIPS.
- One of the advantages of forward-chaining over backward-chaining is that the reception of new data can trigger new inferences, which makes the engine better suited to dynamic situations in which conditions are likely to change.

Example
The rule base contains the following four rules:
- If X croaks and eats flies - Then X is a frog
- If X chirps and sings - Then X is a canary
- If X is a frog - Then X is green
- If X is a canary - Then X is yellow

Goal: conclude the color of a pet named Fritz, given that he croaks and eats flies,

Inference Mechanism:

- This rule base would be searched and the first rule would be selected, because its antecedent (If Fritz croaks and eats flies) matches our data. Now the consequents (Then X is a frog) is added to the data.
- The rule base is again searched and this time the third rule is selected, because its antecedent (If Fritz is a frog) matches our data that was just confirmed. Now the new consequent (Then Fritz is green) is added to our data.
- Nothing more can be inferred from this information, but we have now accomplished our goal of determining the color of Fritz.

Backward chaining

- Backward chaining (or backward reasoning) is an inference method that can be described as working backward from the goal(s).
- It is used in automated theorem provers, proof assistants and other artificial intelligence applications.
- In game theory, its application to subgames in order to find a solution to the game is called backward induction.
- In chess, it is called retrograde analysis, and it is used to generate tablebases for chess endgames for computer chess.
- Backward chaining is implemented in logic programming by SLD resolution
- Rules are based on the modus ponens inference rule.
- It is one of the two most commonly used methods of reasoning with inference rules and logical implications.
- Backward chaining systems usually employ a depth-first search strategy.
- Backward chaining starts with a list of goals (or a hypothesis) and works backwards from the consequent to the antecedent to see if there is data available that will support any of these consequents.
- An inference engine using backward chaining would search the inference rules until it finds one which has a consequent (Then clause) that matches a desired goal.
- If the antecedent (If clause) of that rule is not known to be true, then it is added to the list of goals.
- Because the list of goals determines which rules are selected and used, this method is called goal-driven, in contrast to data-driven forward-chaining inference. The backward chaining approach is often employed by expert systems.

Example

The rule base contains the following four rules:

- If X croaks and eats flies - Then X is a frog
- If X chirps and sings - Then X is a canary
- If X is a frog - Then X is green
- If X is a canary - Then X is yellow

Goal: conclude the color of a pet named Fritz, given that he croaks and eats flies,

Inference Mechanism:

- This rule base would be searched and the third and fourth rules would be selected, because their consequents (Then Fritz is green, Then Fritz is yellow) match the goal (to determine Fritz's color).
- It is not yet known that Fritz is a frog, so both the antecedents (If Fritz is a frog, If Fritz is a canary) are added to the goal list.
- The rule base is again searched and this time the first two rules are selected, because their consequents (Then X is a frog, Then X is a canary) match the new goals that were just added to the list.
- The antecedent (If Fritz croaks and eats flies) is known to be true and therefore it can be concluded that Fritz is a frog, and not a canary.
- The goal of determining Fritz's color is now achieved (Fritz is green if he is a frog, and yellow if he is a canary, but he is a frog since he croaks and eats flies; therefore, Fritz is green).

Statistical Reasoning

Probability & Bayes Theorem

- An important goal for many problems solving systems is to collect evidence as the system goes along and to modify its behavior on the basis of evidence.
- To model this behavior we need a statistical theory of evidence.
- Bayesian statistics is such a theory. The fundamental notion of Bayesian statistics is that of condition probability.
- Read the expression as the probability of Hypothesis H given that we have observed evidence E.
- To compute this, we need to take into account the prior probability of H (the probability that we would assign to H if we had no evidence) & the extent to which E provides evidence of H. To do this we need to define a universe that contains an exhaustive, mutually exclusive set of H_i, among which we are trying to discriminate
- P(Hi\E)=The probability that hypothesis Hi is true given evidence E, P(E\Hi) (The probability that we will observe evidence E given that hypothesis Hi is true.)
- P(Hi) = The apriori probability that hypothesis Hi is true in the absence of any specific evidence. These probabilities are called prior probabilities.
- Bayes theorem states then that
  P(Hi\E)=P(E\Hi)* P(Hi)/P(E)

Bayesian Networks (Bayes N/W, Belief N/W, Directed Acyclic Graph)

- It is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph.
- Nodes represent the random variables and the edges represent conditional dependencies.

- Each node is associated with a probability function.
- Disadvantage:
  i. Loop can occur sometimes
  ii. The number of paths to explore exponentially with each node.

Eg: Refer Class note

Entailment (Logical consequence)

- Entailment is the relationship between current sentences and new sentences derived from the current sentences.
- Entailment is one of the most fundamental concepts in logic.
- It is the relationship between statements that holds true when one logically "follows from" one or more others.
- A valid logical argument is one in which the conclusions follow from its premises, and its conclusions are consequences of its premises.
- Entailment reflects the relation of one fact in the world following from the others.
- Knowledge base (KB) entails sentence if and only if is true in all worlds where KB is true.
- ***Sound and complete inference***: If a knowledgebase is true then any sentences α derived from knowledgebase by a sound inference is also true. Completeness is a condition if it can derive any sentences that are entailed.
- An inference algorithm that derives only entailed sentences id called sound or truth preserving.

Horn Clause

- Horn clause is a logical formula of a particular rule-like form which gives it useful properties for use in logic programming.
- A Horn clause is a clause (a disjunction of literals) with at most one positive literal.
- A Horn clause with exactly one positive literal is a ***definite clause***.
  Eg; ¬p ∨ ¬q ∨ ... ∨ ¬t ∨ u (assume that, if p and q and ... and t all holds, then also u holds)
- A definite clause with no negative literals is sometimes called a ***fact***.
  Eg: u (assume that u holds).
- Horn clause without a positive literal is sometimes called a ***goal clause*** i.e. the empty clause consisting of no literals is a goal clause.
  Eg: ¬p ∨ ¬q ∨ ... ∨ ¬t (show that p and q and ... and t all holds)

Well-Formed Formula (WFF)

Rules for constructing Wffs

A predicate name followed by a list of variables such as P(x, y), where P is a predicate name, and x and y are variables, is called an atomic formula.

Wffs are constructed using the following rules:

1. True and False are wffs.

2. Each propositional constant (i.e. specific proposition), and each propositional variable (i.e. a variable representing propositions) are wffs.

3. Each atomic formula (i.e. a specific predicate with variables) is a wff.

4. If A, B, and C are wffs, then so are $\neg$ A, (A $\wedge$ B), (A $\vee$ B), (A $\rightarrow$ B), and (A $\leftrightarrow$ B).

5. If x is a variable (representing objects of the universe of discourse), and A is a wff, then so are $\forall$ x A and $\exists$ x A .