Knowledge Representation

Knowledge is a theoretical or practical understanding of a subject or a domain. Knowledge is also the sum of what is currently known.

Knowledge is the sum of what is known: the body of truth, information, and principles acquired by mankind. Or, "Knowledge is what I know, Information is what we know."

There are many other definitions such as:

- Knowledge is "information combined with experience, context, interpretation, and reflection. It is a high-value form of information that is ready to apply to decisions and actions." (T. Davenport et al., 1998)

- Knowledge is human expertise stored in a person‘s mind, gained through experience, and interaction with the person‘s environment." (Sunasee and Sewery, 2002)

- Knowledge is information evaluated and organized by the human mind so that it can be used purposefully, e.g., conclusions or explanations." (Rousa, 2002)

The intended role of knowledge representation in artificial intelligence is to reduce problems of intelligent action to search problems. The existing languages used for knowledge representation all have a well-defined syntax, and sentences in these languages have agreed-upon meanings. But there is no general understanding of exactly what knowledge we bring to bear to solve specific problems, and there is little known about how any particular bit of knowledge is to be encoded in any particular language.

Logic

Logic is the discipline that deals with the methods of reasoning. On an elementary level, logic provides rules and techniques for determining whether a given argument is valid. Logical reasoning is used in mathematics to prove theorems, in computer science to verify correctness of a program and to prove theorems, in natural and physical sciences to draw conclusion from experiments, in social science and in our daily life to solve multitude of problems. We are constantly using logical reasoning.

Logic is concise, unambiguous, expressive, context insensitive and effective.

Syntax, Semantics and Proof Theory

Syntax: In logic, syntax is anything having to do with formal languages or formal systems without regard to any interpretation or meaning given to them. Syntax is concerned with the rules used for constructing, or transforming the symbols and words of a language. The notion of syntax is clear enough in ordinary arithmetic: "x + y = 4" is a well-formed sentence, whereas "x4y+ =" is not.

Semantics: The semantics defines the truth of each sentence with respect to each possible world. A logic must also define the semantics or meaning of sentences. For example, the semantics for arithmetic specifies that the sentence "x + y =4" is true in a world where x is 2 and y is 2, but false in a world where x is 1 and y is 1. In standard logics, every sentence must be either true or false in each possible world—there is no "in between.

Proof Theory: Proof theory is a major branch of mathematical logic that represents proofs as formal mathematical objects, facilitating their analysis by mathematical techniques. Proofs are typically presented as inductively-defined data structures such as plain lists, boxed lists, or trees, which are constructed according to the axioms and rules of inference of the logical system. These are rules for generating new sentences that are true, given that old sentence, i.e. the sentence used to derive them are true.

There are three types of Logic: Propositional Logic, Predicate logic and Fuzzy Logic.

Propositional logic

A proposition is a declarative sentence (that is, a sentence that declares a fact) that is either true or false, but not both. All the following declarative sentences are propositions.

1. Washington, D.C., is the capital of the United States of America.

2. Toronto is the capital of Canada.

3. 1 + 1 = 2.

4. 2 + 2 = 3.

Propositions 1 and 3 are true, whereas 2 and 4 are false.

Consider the following sentences.

1. What time is it?

2. Read this carefully.

3. x + 1 = 2.

4. x + y = z.

Sentences 1 and 2 are not propositions because they are not declarative sentences. Sentences 3 and 4 are not propositions because they are neither true nor false. Note that each of sentences 3 and 4 can be turned into a proposition if we assign values to the variables.

Logical Connectives

A Logical Connective is a symbol which is used to connect two or more propositional or predicate logics in such a manner that resultant logic depends only on the input logics and the meaning of the connective used. Generally there are five connectives which are −

OR (∨), AND (∧), Negation/ NOT (¬), Implication / if-then (→) and If and only if (↔)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Figure Truth tables for the five logical connectives. To use the table to compute, for example, the value of $P \vee Q$ when P is true and Q is false, first look on the left for the row where P is *true* and Q is *false* (the third row). Then look in that row under the $P \vee Q$ column to see the result: *true*.

We use letters to denote propositional variables (or sentential variables), that is, variables that represent propositions, just as letters are used to denote numerical variables. The conventional letters used for propositional variables are p, q, r, s, … . The truth value of a proposition is true, denoted by T, if it is a true proposition, and the truth value of a proposition is false, denoted by F, if it is a false proposition. Propositions that cannot be expressed in terms of simpler propositions are called atomic propositions.

Tautology: A formula of propositional logic is a tautology if the formula itself is always true, regardless of which valuation is used for the propositional variables.

Contradiction: A formula of propositional logic is a contradiction if the formula itself is always false, regardless of which valuation is used for the propositional variables.

Example: Show that $(p \rightarrow q) \vee (q \rightarrow p)$ is a tautology.

Truth Table for $(p \rightarrow q) \vee (q \rightarrow p)$				
p	q	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \vee (q \rightarrow p)$
T	T	T	T	T
T	F	F	T	T
F	T	T	F	T
F	F	T	T	T

Contingency: A formula of propositional logic is a contingency if the formula is neither always true nor always false.

Atomic sentences: The atomic sentences consist of a single proposition symbol. Each such symbol stands for a proposition that can be true or false.

Composite sentence: Composite sentences are formed from atomic sentences and express relationships among the constituent sentences.

Logical Equivalences

In logic and mathematics, statements p and q are said to be logically equivalent if they are provable from each other under a set of axioms, or have the same truth value in every model. The logical equivalence of p and q is sometimes expressed as $p \equiv q$, $p :: q$, E_{pq} or $p \Leftrightarrow q$, depending on the notation being used. However, these symbols are also used for material equivalence, so proper interpretation would depend on the context. Logical equivalence is different from material equivalence, although the two concepts are intrinsically related.

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Figure: Standard logical equivalences. The symbols α, β, and γ stand for arbitrary sentences of propositional logic.

Knowledge base:

A knowledge base is a collection of sentences $\alpha_1,\alpha_2,...,\alpha_k$ that we know are true, i.e. $\alpha_1 \wedge \alpha_2 \wedge ... \wedge \alpha_k$. The sentences in the knowledge base allows us to compactly specify the allowable states of the world. By adding new sentences, the number of possible worlds consistent with our knowledge base could decrease (if we gain new information), go to zero (if the new sentence is inconsistent with the existing knowledge base), or remain the same (if the new sentence is entailed by existing sentences).

KB Evaluation: The result of the KB evaluation is the conjunction of the results of the evaluations of all the sentences in KB

Example KB

KB:
$A \vee B$
$\neg C \vee A$

A	B	C	KB
F	F	F	F
F	F	T	F
F	T	F	T
F	T	T	F
T	F	F	T
T	F	T	T
T	T	F	T
T	T	T	T

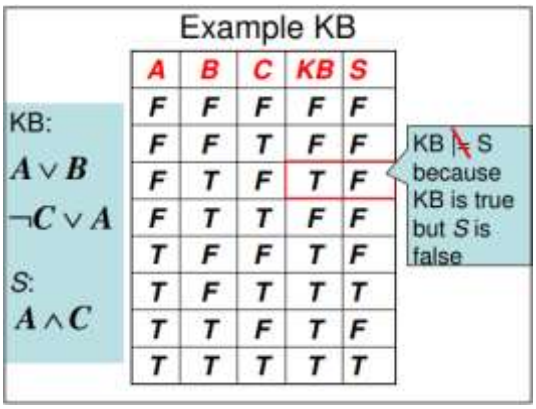
Entailment:

Entailment means that one thing follows from another:

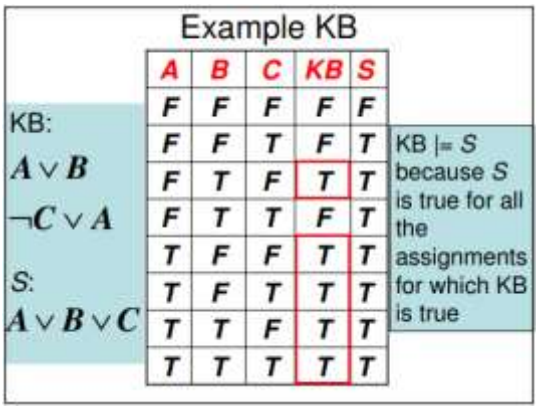
KB ⊨ α

Knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true

Note: We do not care about those models that evaluate KB to False. The result of evaluating S for these models is irrelevant. We can determine whether S ⊨ P by finding Truth Table for S and P, if any row of Truth Table where all formulae in S is true.



A	B	C	KB	S
F	F	F	F	F
F	F	T	F	F
F	T	F	T	F
F	T	T	F	F
T	F	F	T	F
T	F	T	T	T
T	T	F	T	F
T	T	T	T	T



A	B	C	KB	S
F	F	F	F	F
F	F	T	F	T
F	T	F	T	T
F	T	T	F	T
T	F	F	T	T
T	F	T	T	T
T	T	F	T	T
T	T	T	T	T

Representing Implications

There are many reasons to translate English sentences into expressions involving propositional variables and logical connectives. In particular, English (and every other human language) is often ambiguous. Translating sentences into compound statements and other types of logical expressions, removes the ambiguity. Note that this may involve making a set of reasonable assumptions based on the intended meaning of the sentence. Moreover, once we have translated sentences from English into logical expressions, we can analyze these logical expressions to determine their truth values, we can manipulate them, and we can use rules of inference to reason about them.

“If you are under 4 feet tall unless you are older than 16 years old then you cannot ride the roller coaster.”

Solution: Let q, r, and s represent “You can ride the roller coaster,” “You are under 4 feet tall,” and “You are older than 16 years old,” respectively. Then the sentence can be translated to

(r ∧ ¬s) → ¬q.

Clausal Form

Propositional Resolution works only on expressions in clausal form. Before the rule can be applied, the premises and conclusions must be converted to this form. Fortunately, as we shall see, there is a simple procedure for making this conversion.

A literal is either an atomic sentence or a negation of an atomic sentence. For example, if p is a logical constant, the following sentences are both literals.

- p
- ¬p

A clausal sentence is either a literal or a disjunction of literals. If p and q are logical constants, then the following are clausal sentences.

- p
- ¬p
- ¬p ∨ q

A clause is the set of literals in a clausal sentence. For example, the following sets are the clauses corresponding to the clausal sentences above.

- {p}
- {¬p}
- {¬p, q}

Note that the empty set {} is also a clause. It is equivalent to an empty disjunction and, therefore, is unsatisfiable. As we shall see, it is a particularly important special case.

Conjunctive Normal Form(CNF)

A statement is in conjunctive normal form if it is a conjunction (sequence of ANDs) consisting of one or more conjuncts, each of which is a disjunction (OR) of one or more literals (i.e., statement letters and negations of statement letters). Examples of conjunctive normal forms include

- A
- (A v B) ∧ (¬A v C)
- A v B

- A ∧ (B v C)

Algorithm

- Eliminate ↔rewriting P↔Q as (P→Q)∧(Q→P)
- Eliminate →rewriting P→Q as ¬P∨Q
- Use De Morgan's laws to push ¬ inwards:
  - ✓ rewrite ¬(P∧Q) as ¬P∨¬Q
  - ✓ rewrite ¬(P∨Q) as ¬P∧¬Q
- Eliminate double negations: rewrite ¬¬P as P
- Use the distributive laws to get CNF:
  - ✓ rewrite (P∧Q)∨R as (P∨R)∧(Q∨R)
- Flatten nested clauses:
  - ✓ (P∧Q) ∧ R as P∧Q ∧ R
  - ✓ (P∨Q)∨R as P∨Q∨R

Eg.

$B \Leftrightarrow (A \vee C)$

- Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow ß$ with $(\alpha \Rightarrow ß) \wedge (ß \Rightarrow \alpha)$.

$$\equiv (B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B)$$

- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow ß$ with $\neg \alpha \vee ß$.

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

- Move $\neg$ inwards using de Morgan's rules and double-negation:

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

- Apply distributivity law ($\wedge$ over $\vee$) and flatten:

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

Question: Convert to CNF:

a) $(\neg B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B) \wedge (\neg B)$

b) $\neg((\neg A \vee B) \wedge C \wedge (A \Rightarrow B))$

c) $(B \Leftrightarrow (A \vee C)) \wedge \neg B$

Inference Rules

Consider the following argument involving propositions (which, by definition, is a sequence of propositions):

"If you have a current password, then you can log onto the network."

"You have a current password."

Therefore,

"You can log onto the network."

We would like to determine whether this is a valid argument. That is, we would like to determine whether the conclusion "You can log onto the network" must be true when the premises "If you have a current password, then you can log onto the network" and "You have a current password" are both true.

Before we discuss the validity of this particular argument, we will look at its form. Use p to represent "You have a current password" and q to represent "You can log onto the network." Then, the argument has the form

$$\begin{array}{l} p \rightarrow q \\ \underline{\quad p \quad} \\ \therefore q \end{array}$$

where $\therefore$ is the symbol that denotes "therefore."

We can always use a truth table to show that an argument form is valid. We do this by showing that whenever the premises are true, the conclusion must also be true. However, this can be a tedious approach. Instead, we can first establish the validity of some relatively simple argument forms, called rules of inference. These rules of inference can be used as building blocks to construct

more complicated valid argument forms. We will now introduce the most important rules of inference in propositional logic.

Rule of Inference	*Tautology*	*Name*
p $p \rightarrow q$ $\therefore q$	$(p \wedge (p \rightarrow q)) \rightarrow q$	Modus ponens
$\neg q$ $p \rightarrow q$ $\therefore \neg p$	$(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$	Modus tollens
$p \rightarrow q$ $q \rightarrow r$ $\therefore p \rightarrow r$	$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$	Hypothetical syllogism
$p \vee q$ $\neg p$ $\therefore q$	$((p \vee q) \wedge \neg p) \rightarrow q$	Disjunctive syllogism
p $\therefore p \vee q$	$p \rightarrow (p \vee q)$	Addition
$p \wedge q$ $\therefore p$	$(p \wedge q) \rightarrow p$	Simplification
p q $\therefore p \wedge q$	$((p) \wedge (q)) \rightarrow (p \wedge q)$	Conjunction
$p \vee q$ $\neg p \vee r$ $\therefore q \vee r$	$((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$	Resolution

Using Rules of Inference to Build Arguments

Show that the premises "It is not sunny this afternoon and it is colder than yesterday," "We will go swimming only if it is sunny," "If we do not go swimming, then we will take a canoe trip," and "If we take a canoe trip, then we will be home by sunset" lead to the conclusion "We will be home by sunset."

Solution: Let p be the proposition "It is sunny this afternoon," q the proposition "It is colder than yesterday," r the proposition "We will go swimming," s the proposition "We will take a canoe trip," and t the proposition "We will be home by sunset." Then the premises become

$\neg p \wedge q$,

$r \rightarrow p$,

$\neg r \rightarrow s$, and

$s \rightarrow t$.

The conclusion is simply t.

We need to give a valid argument with premises $\neg p \wedge q$, $r \rightarrow p$, $\neg r \rightarrow s$, and $s \rightarrow t$ and conclusion t. We construct an argument to show that our premises lead to the desired conclusion as follows.

1. $\neg p \wedge q$ //Premise
2. $\neg p$ //Simplification using (1)
3. $r \rightarrow p$ //Premise
4. $\neg r$ //Modus tollens using (2) and (3)
5. $\neg r \rightarrow s$ //Premise
6. s //Modus ponens using (4) and (5)
7. $s \rightarrow t$ //Premise
8. t //Modus ponens using (6) and (7)

Note that we could have used a truth table to show that whenever each of the four hypotheses is true, the conclusion is also true. However, because we are working with five propositional variables, p, q, r, s, and t, such a truth table would have 32 rows.

Using Resolution to Prove Statement

Computer programs have been developed to automate the task of reasoning and proving theorems.

Many of these programs make use of a rule of inference known as resolution. This rule of inference is based on the tautology:

$$((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$$

Resolution example

Example 1.

Consider the knowledge base given as below:

$KB = (B \Leftrightarrow (A \vee C)) \wedge \neg B$

Prove that $\neg A$ can be inferred from above KB by using resolution.

Solution:

Convert KB into CNF

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

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

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

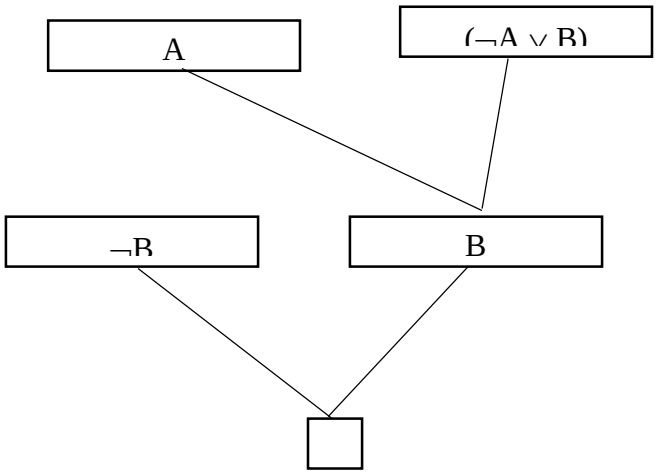
$$\equiv (\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B) \wedge \neg B$$

Add negation of sentence to be inferred from KB into KB

Now KB contains following sentences all in CNF

1. $(\neg B \vee A \vee C)$ //Premise
2. $(\neg A \vee B)$ //Premise
3. $(\neg C \vee B)$ //Premise
4. $\neg B$ //Premise
5. A //Negation of conclusion

Now use Resolution algorithm



Since we end up with empty clause the sentence to be inferred is proved.

Example 2.

Use resolution to prove that the following argument is valid:

1. $p \Rightarrow (q \vee r)$

2. $\sim r$

$\therefore q$

Solution:

Step 1: Once again, we first convert the premises to clause form.

1) $\sim p \vee (q \vee r)$

2) $\sim r$

Step 2: Negate the conclusion.

3) $\sim q$

Step 3: Convert the negation of the conclusion to clause form:

3) ~q // it was already in clause form

Step 4: Search for a contradiction in this list of clauses: 1), 2), and 3)

We try using resolution between 1) and 3) and we obtain:

4) ~p ∨ r

Then using resolution between 4) and 2):

5) ~p

It soon becomes apparent that "we are spinning our wheels", i.e., there is no contradiction present. Once we have searched (everywhere) to find a contradiction and are sure that none is present, then we can safely assume that the argument is not valid. If p had been given as a premise as well, then this argument would indeed be valid.

Example 3

Show that the premises "If you send me an e-mail message, then I will finish writing the program," "If you do not send me an e-mail message, then I will go to sleep early," and "If I go to sleep early, then I will wake up feeling refreshed" lead to the conclusion "If I do not finish writing the program, then I will wake up feeling refreshed."

Solution:

Let

p: You send me an e-mail message

q: I will finish writing the program

r: I will go to sleep early

s: I will wake up feeling refreshed.

Then the premises are

1. $p \rightarrow q$
2. $\neg p \rightarrow r$
3. $r \rightarrow s$.

The desired conclusion is

4. $\neg q \rightarrow s$.

we now convert the premises to clause form.

1. $p \rightarrow q$
   $\neg p \vee q$
2. $\neg p \rightarrow r$
   $\neg(\neg p) \vee r$
   $p \vee r$
3. $r \rightarrow s$
   $\neg r \vee s$

We also need negation of conclusion. So,

4. $\neg(\neg q \rightarrow s)$
   $\neg(\neg(\neg q) \vee s)$
   $\neg(q \vee s)$
   $\neg q \wedge \neg s$

So the KB contains:

1. $\neg p \vee q$
2. $p \vee r$
3. $\neg r \vee s$
4. $\neg q$
5. $\neg s$

Now we use resolution to derive following clauses.

6. $\neg r$ // resolution between 5 and 3
7. p // resolution between 6 and 2
8. q // resolution between 7 and 1
9. $\square$ // resolution between 8 and 4

Since we end up in empty clause, the conclusion is valid.

Predicate Logic

Propositional logic cannot adequately express the meaning of all statements in mathematics and in natural language. For example, suppose that we know that "Every computer connected to the university network is functioning properly."

No rules of propositional logic allow us to conclude the truth of the statement "MATH3 is functioning properly," where MATH3 is one of the computers connected to the university network.

Likewise, we cannot use the rules of propositional logic to conclude from the statement "CS2 is under attack by an intruder," where CS2 is a computer on the university network, to conclude the truth of "There is a computer on the university network that is under attack by an intruder."

To perform these tasks, a more powerful type of logic called predicate logic is required.

Predicate

Statements involving variables, such as

"x > 3," "x = y + 3," "x + y = z,"

And "Computer x is under attack by an intruder," And "Computer x is functioning properly," are often found in mathematical assertions, in computer programs, and in system specifications. These statements are neither true nor false when the values of the variables are not specified. But there is a way by which propositions can be produced from such statements.

The statement "x is greater than 3" has two parts. The first part, the variable x, is the ***subject*** of the statement. The second part—the ***predicate***, "is greater than 3"—refers to a property that the subject of the statement can have.

We can denote the statement "x is greater than 3" by P(x), where P denotes the predicate "is greater than 3" and x is the variable. The statement P(x) is also said to be the value of the propositional function P at x. Once a value has been assigned to the variable x, the statement P(x) becomes a proposition and has a truth value. Consider following Examples.

EXAMPLE: Let P(x) denote the statement "x > 3." What are the truth values of P(4) and P(2)?

Solution: We obtain the statement P(4) by setting x = 4 in the statement "x > 3." Hence, P(4), which is the statement "4 > 3," is true. However, P(2), which is the statement "2 > 3," is false.

EXAMPLE: Let A(x) denote the statement "Computer x is under attack by an intruder." Suppose that of the computers on campus, only CS2 and MATH1 are currently under attack by intruders. What are truth values of A(CS1), A(CS2), and A(MATH1)?

Solution: We obtain the statement A(CS1) by setting x = CS1 in the statement "Computer x is under attack by an intruder." Because CS1 is not on the list of computers currently under attack, we conclude that A(CS1) is false. Similarly, because CS2 and MATH1 are on the list of computers under attack, we know that A(CS2) and A(MATH1) are true.

We can also have statements that involve more than one variable. For instance, consider the statement "x = y + 3."We can denote this statement by Q(x, y), where x and y are variables and Q is the predicate. When values are assigned to the variables x and y, the statement Q(x, y) has a truth value.

EXAMPLE: Let Q(x, y) denote the statement "x = y + 3." What are the truth values of the propositions Q(1, 2) and Q(3, 0)?

Solution: To obtain Q(1, 2), set x = 1 and y = 2 in the statement Q(x, y). Hence, Q(1, 2) is the statement "1 = 2 + 3," which is false. The statement Q(3, 0) is the proposition "3 = 0 + 3," which is true.

EXAMPLE: Let A(c, n) denote the statement "Computer c is connected to network n," where c is a variable representing a computer and n is a variable representing a network. Suppose that the computer MATH1 is connected to network CAMPUS2, but not to network CAMPUS1. What are the values of A(MATH1, CAMPUS1) and A(MATH1, CAMPUS2)?

Solution: Because MATH1 is not connected to the CAMPUS1 network, we see that A(MATH1, CAMPUS1) is false. However, because MATH1 is connected to the CAMPUS2 network, we see that A(MATH1, CAMPUS2) is true.

Similarly, we can let R(x, y, z) denote the statement "x + y = z." When values are assigned to the variables x, y, and z, this statement has a truth value.

A statement of the form $P(x_1, x_2,\ldots, x_n)$ is the value of the propositional function P at the n-tuple $(x_1, x_2,\ldots, x_n)$, and P is also called an n-place predicate or an n-ary predicate.

Quantifiers

When the variables in a propositional function are assigned values, the resulting statement becomes a proposition with a certain truth value. However, there is another important way, called quantification, to create a proposition from a propositional function. Quantification expresses the extent to which a predicate is true over a range of elements. In English, the words all, some, many, none, and few are used in quantifications. We will focus on two types of quantification here; universal quantification, which tells us that a predicate is true for every element under consideration, and existential quantification, which tells us that there is one or more element under consideration for which the predicate is true. The area of logic that deals with predicates and quantifiers is called the predicate calculus.

Universal Quantifier

Many mathematical statements assert that a property is true for all values of a variable in a particular domain, called the domain of discourse (or the universe of discourse), often just referred to as the domain. Such a statement is expressed using universal quantification. The universal quantification of P(x) for a particular domain is the proposition that asserts that P(x) is true for all values of x in this domain.

The notation ∀xP(x) denotes the universal quantification of P(x). Here ∀ is called the universal quantifier. We read ∀xP(x) as "for all x P(x)" or "for every x P(x)." An element for which P(x) is false is called a counterexample to ∀xP(x).

EXAMPLE: Let P(x) be the statement "x + 1 > x." What is the truth value of the quantification ∀xP(x), where the domain consists of all real numbers?

Solution: Because P(x) is true for all real numbers x, the quantification ∀xP(x) is true.

Existential Quantifier

Many mathematical statements assert that there is an element with a certain property. Such statements are expressed using existential quantification. With existential quantification, we form a proposition that is true if and only if P(x) is true for at least one value of x in the domain.

The existential quantification of P(x) is the proposition "There exists an element x in the domain such that P(x)." We use the notation ∃xP(x) for the existential quantification of P(x). Here ∃ is called the existential quantifier.

EXAMPLE: Let P(x) denote the statement "x > 3." What is the truth value of the quantification ∃xP(x), where the domain consists of all real numbers?

Solution: Because "x > 3" is sometimes true—for instance, when x = 4—the existential quantification of P(x), which is ∃xP(x), is true.

The statement ∃xP(x) is false if and only if there is no element x in the domain for which P(x) is true. That is, ∃xP(x) is false if and only if P(x) is false for every element of the domain.

Uniqueness Quantifier

Universal and existential quantifiers are the most important quantifiers in mathematics and computer science. However, there is no limitation on the number of different quantifiers we can define, such as "there are exactly two," "there are no more than three," "there are at least 100," and so on. Of these other quantifiers, the one that is most often seen is the uniqueness quantifier, denoted by ∃! or ∃1.

The notation ∃!xP(x) [or ∃1xP(x)] states "There exists a unique x such that P(x) is true." (Other phrases for uniqueness quantification include "there is exactly one" and "there is one and only one.")

Translating from English into Logical Expressions

Translating sentences in English (or other natural languages) into logical expressions is a crucial task in mathematics, logic programming, artificial intelligence, software engineering, and many other disciplines. In earlier section, propositional logic was used to express sentences in logical expressions. In propositional logic predicates and quantifiers are not needed. Translating from English to logical expressions becomes even more complex when quantifiers are needed. Furthermore, there can be many ways to translate a particular sentence. (As a consequence, there is no "cookbook" approach that can be followed step by step.)

EXAMPLE: Express the statement "Every student in this class has studied calculus" using predicates and quantifiers.

Solution: First, we rewrite the statement so that we can clearly identify the appropriate quantifiers to use. Doing so, we obtain: "For every student in this class, that student has studied calculus."

Next, we introduce a variable x so that our statement becomes "For every student x in this class, x has studied calculus." Now assume,

C(x): x has studied calculus.

If the domain for x consists of the students in the class, we can translate our statement as

$\forall x C(x)$.

However, there are other correct approaches; different domains of discourse and other predicates can be used. The approach we select depends on the subsequent reasoning we want to carry out. For example, we may be interested in a wider group of people than only those in this class. If we change the domain to consist of all people, we will need to express our statement as

"For every person x, if person x is a student in this class, then x has studied calculus."

If S(x) represents the statement that person x is in this class, we see that our statement can be expressed as $\forall x(S(x) \rightarrow C(x))$. [Our statement cannot be expressed as $\forall x(S(x) \wedge C(x))$ because this statement says that all people are students in this class and have studied calculus!]

Finally, when we are interested in the background of people in subjects besides calculus, we may prefer to use the two-variable quantifier Q(x, y) for the statement "Student x has studied subject y." Then we would replace C(x) by Q(x, calculus) in both approaches to obtain

$\forall x Q(x, \text{calculus})$ or

$\forall x(S(x) \rightarrow Q(x, \text{calculus}))$.

EXAMPLE: Consider these statements. The first two are called premises and the third is called the conclusion.

The entire set is called an argument.

"All lions are fierce."

"Some lions do not drink coffee."

"Some fierce creatures do not drink coffee."

Let, P(x): x is a lion,

Q(x): x is fierce, and

R(x): x drinks coffee.

Assuming that the domain consists of all creatures, express the statements in the argument using quantifiers and P(x), Q(x), and R(x).

Solution: We can express these statements as

$\forall x(P(x) \rightarrow Q(x))$.

$\exists x(P(x) \wedge \neg R(x))$.

$\exists x(Q(x) \wedge \neg R(x))$.

Notice that the second statement cannot be written as $\exists x(P(x) \rightarrow \neg R(x))$. The reason is that $P(x) \rightarrow \neg R(x)$ is true whenever x is not a lion, so that $\exists x(P(x) \rightarrow \neg R(x))$ is true as long as there is at least one creature that is not a lion, even if every lion drinks coffee. Similarly, the third statement cannot be written as

$\exists x(Q(x) \rightarrow \neg R(x))$.

EXAMPLE: Consider these statements, of which the first three are premises and the fourth is a valid conclusion.

"All hummingbirds are richly colored."

"No large birds live on honey."

"Birds that do not live on honey are dull in color."

"Hummingbirds are small."

Let P(x): x is a hummingbird,

Q(x): x is large,

R(x): x lives on honey, and

S(x): x is richly colored.

Assuming that the domain consists of all birds, express the statements in the argument using quantifiers and P(x), Q(x), R(x), and S(x).

Solution: We can express the statements in the argument as

$\forall x(P(x) \rightarrow S(x))$.

$\neg\exists x(Q(x) \wedge R(x))$.

$\forall x(\neg R(x) \rightarrow \neg S(x))$.

$\forall x(P(x) \rightarrow \neg Q(x))$.

(Note we have assumed that "small" is the same as "not large" and that "dull in color" is the same as "not richly colored."

Nested Quantifiers

A statement is said to have nested quantifiers if one quantifier is within the scope of another.

For example, $\forall x\exists y(x + y = 0)$.

Many mathematical statements involve multiple quantifications of propositional functions involving more than one variable. It is important to note that the order of the quantifiers is important, unless all the quantifiers are universal quantifiers or all are existential quantifiers.

Properties of nested quantifiers

$\forall x\ \forall y$ is the same as $\forall y\ \forall x$

$\exists x\ \exists y$ is the same as $\exists y\ \exists x$

$\exists x\ \forall y$ is not the same as $\forall y\ \exists x$

$\exists x\ \forall y$ Loves(x,y)

- "There is a person who loves everyone in the world"

$\forall y\ \exists x$ Loves(x,y)

- "Everyone in the world is loved by at least one person"

Translating from Nested Quantifiers into English and vice versa

Let L(x, y) be the statement "x loves y," where the domain for both x and y consists of all people in the world. Use quantifiers to express each of these statements.

a) Everybody loves Jerry.

$\forall x L(x, Jerry)$

b) Everybody loves somebody.

$\forall x \exists y L(x, y)$

c) There is somebody whom everybody loves.

$\exists y \forall x L(x, y)$

d) Nobody loves everybody.

$\forall x \exists y \neg L(x, y)$ or $\neg \exists x \forall y L(x, y)$

e) Everyone loves himself or herself

$\forall x L(x, x)$

Example: Express the statement "If a person is female and is a parent, then this person is someone's mother" as a logical expression involving predicates, quantifiers with a domain consisting of all people, and logical connectives.

Solution: The statement "If a person is female and is a parent, then this person is someone's mother" can be expressed as "For every person x, if person x is female and person x is a parent, then there exists a person y such that person x is the mother of person y." We introduce the propositional functions F(x) to represent "x is female," P(x) to represent "x is a parent," and M(x, y) to represent "x is the mother of y." The original statement can be represented as

$\forall x((F(x) \wedge P(x)) \rightarrow \exists y M(x, y))$.

Now, we can move $\exists y$ to the left so that it appears just after $\forall x$, because y does not appear in $F(x) \wedge P(x)$. We obtain the logically equivalent expression

$\forall x \exists y((F(x) \wedge P(x)) \rightarrow M(x, y))$.

Expressions with nested quantifiers expressing statements in English can be quite complicated. The first step in translating such an expression is to write out what the quantifiers and predicates in the expression mean. The next step is to express this meaning in a simpler sentence.

EXAMPLE Translate the statement

$$\forall x(C(x) \vee \exists y(C(y) \wedge F(x, y)))$$

into English, where C(x) is "x has a computer," F(x, y) is "x and y are friends," and the domain for both x and y consists of all students in your school.

Solution: The statement says that for every student x in your school, x has a computer or there is a student y such that y has a computer and x and y are friends. In other words, every student in your school has a computer or has a friend who has a computer.

EXAMPLE Translate the statement

$$\exists x \forall y \forall z((F(x, y) \wedge F(x, z) \wedge (y \neq z)) \rightarrow \neg F(y, z))$$

into English, where F(a, b) means a and b are friends and the domain for x, y, and z consists of all students in your school.

Solution: We first examine the expression $(F(x, y) \wedge F(x, z) \wedge (y \neq z)) \rightarrow \neg F(y, z)$. This expression says that if students x and y are friends, and students x and z are friends, and furthermore, if y and z are not the same student, then y and z are not friends. It follows that the original statement, which is triply quantified, says that there is a student x such that for all students y and all students z other than y, if x and y are friends and x and z are friends, then y and z are not friends. In other words, there is a student none of whose friends are also friends with each other.

Conversion to CNF

1. Eliminate implications and bi-implications as in propositional logic
2. Reduce the scope of each $\neg$ to a single term.

   a. $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$

   b. $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$

   c. $\neg \forall x: P \equiv \exists x: \neg P$

   d. $\neg \exists x: p \equiv \forall x: \neg P$

3. Eliminate double negations
4. Standardize or rename bound variables if necessary so each only occurs once

   e.g. ∀xP(x)∨∃xQ(x) becomes ∀xP(x)∨∃yQ(y)

5. Use equivalences to move quantifiers to the left

   e.g. ∀xP(x)∧Q becomes ∀x (P(x)∧Q) where x is not in Q

   e.g. ∀xP(x)∧∃yQ(y) becomes ∀x∃y(P(x)∧Q(y))

6. Skolemise (replace each existentially quantified variable by a new term). ∃xP(x) becomes P(a0) using a Skolem constant a0 since ∃x occurs at the outermost level ∀x∃yP(x, y) becomes P(x, f0(x)) using a Skolem function f0 since ∃y occurs within ∀x
7. The formula now has only universal quantifiers and all are at the left of the formula: drop them
8. Use distribution laws to get CNF and then clausal form

Example 1:

∀x [∀yP(x, y)→¬∀y(Q(x, y)→R(x, y))]

Solution:

1. ∀x [¬∀yP(x, y)∨¬∀y(¬Q(x, y)∨R(x, y))]

2, 3. ∀x [∃y¬P(x, y)∨∃y(Q(x, y)∧¬R(x, y))]

4. ∀x [∃y¬P(x, y)∨∃z (Q(x, z)∧¬R(x, z))]

5. ∀x∃y∃z [¬P(x, y)∨(Q(x, z)∧¬R(x, z))]

6. ∀x [¬P(x, f (x))∨(Q(x, g(x))∧¬R(x, g(x)))]

7. ¬P(x, f (x))∨(Q(x, g(x))∧¬R(x, g(x)))

8. (¬P(x, f (x))∨Q(x, g(x)))∧(¬P(x, f (x))∨¬R(x, g(x)))

8. {¬P(x, f (x))∨Q(x, g(x)), ¬P(x, f (x))∨¬R(x, g(x))}

Example 2

Everyone who loves all animals is loved by someone:
$\forall x\ [\forall y\ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ Loves(y,x)]$

1. Eliminate biconditionals and implications
$\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

2. Move $\neg$ inwards: $\neg\forall x\ p \equiv \exists x\ \neg p,\ \neg \exists x\ p \equiv \forall x\ \neg p$
$\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)]$
$\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$
$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

3. Standardize variables: each quantifier should use different one
$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z\ Loves(z,x)]$

4. Skolemize: general form of existential instantiation.
Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:
$\forall x\ [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$

5. Drop universal quantifiers:
$[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$

6. Distribute $\vee$ over $\wedge$:
$[Animal(F(x)) \vee Loves(G(x),x)] \wedge [\neg Loves(x,F(x)) \vee Loves(G(x),x)]$

Resolution in the Predicate Logic

Resolution provides a method for finding contradictions in a database of clauses. Resolution refutation proves a theorem by negating the statement that needs to be proved and then adding this negation to the set of axioms that are known (have been assumed) to be true. Resolution refutation proofs involve the following steps:

1. Put the premises (these are sometimes called axioms or hypotheses) into clause form.
2. Add the negation of what is to be proved (i.e., the negation of either the conclusion or goal), in clause form, to the set of premises.
3. Resolve these clauses together, producing new clauses that logically follow from them.
4. Produce a contradiction by generating what is referred to as the empty clause.
5. The substitutions used to produce the empty clause are precisely those under which the opposite of the negated goal is true.

Resolution is refutation complete. This means that a contradiction can always be generated whenever one exists. Resolution refutation proofs require that the premises and the negation of the conclusion be placed in a normal form called clause form (as was required in the propositional logic). Clause form represents both the premises and the negation of the conclusion as a set of disjunction of literals.

Example1:

We use the following well-known argument to illustrate resolution proofs.

Premise 1) Socrates is a mortal.

Premise 2) All mortals will die.

Conclusion: Socrates will die.

First we represent this argument in the predicate logic. We use the predicates Mortal(x) and Will Die (x).

Premise 1) Mortal (Socrates)

Premise 2) (∀x) (Mortal (x) ⇒ Will _ Die (x))

Conclusion) ∴ Will _ Die (Socrates).

Next, the premises are converted into clause form:

Premise 1)	Mortal (Socrates)	//Already in CNF
Premise 2)	(∀x) (Mortal (x) ⇒ Will _ Die (x))	
	≡ (∀x)(¬ Mortal(x) ∨ Will_Die(x))	// Remove implication
	≡ ¬ Mortal (x) ∨ Will _ Die (x)	//Drop universal quantifier
Negate the conclusion: ¬ Will _ Die (Socrates)		//Already in CNF

Our Knowledge Base (KB) consists of:

1) Mortal (Socrates)

2) ¬ Mortal (x) ∨ Will Die (x)

3) ¬Will _ Die (Socrates)

Combining 2) with 3) under the substitution Socrates | x yields:

4) ¬Mortal (Socrates).

Finally, by combining 1) with 4) we derive

5) □

in other words, the empty clause, and so we have a contradiction. Therefore, the negation of what was assumed true, in other words, not (¬ Will _ Die (Socrates)), which is equivalent to Will_Die (Socrates), must be true. The original conclusion does logically follow from the argument's premises, and therefore the argument is valid.

Example 2:

1. All great chefs are Italian.

2. All Italians enjoy good food.

3. Either Michael or Louis is a great chef.

4. Michael is not a great chef.

5. Therefore, Louis enjoys good food.

We use the following predicates:

GC (x) : x is a great chef

I (x) : x is Italian

EF (x) : x enjoys good food

The argument can be represented in the predicate logic as:

1. (∀x) (GC (x) ⇒ I (x))

2. (∀x) (I (x) ⇒ EF (x))

3. GC (Michael) ∨ GC (Louis)

4. ~ GC (Michael)

Therefore:

5. EF (Louis)

Next, we must convert the premises into clause form where no quantifiers can be present. It is easy to remove universal quantifiers because we can assume that all variables are universally quantified.

The premises in clause form:

1. ~ GC (x) ∨ I (x)

2. ~ I (y) ∨ EF (y)

3. GC (Michael) ∨ GC (Louis)

4. ~ GC (Michael)

Negate the conclusion:

5. ~ EF (Louis) // already in CNF

We display the search for a contradiction in graphical form in Figure alongside.

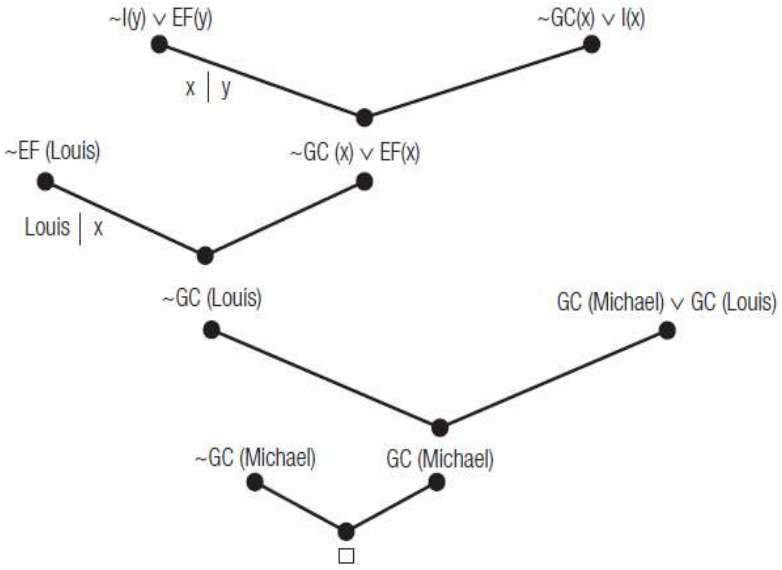


Figure
Resolution proof —a graphical representation.

Since we get empty clause at the end the initial conclusion assumed is valid.

Example 3:

Everyone who loves all animals is loved by someone

Anyone who kills an animal is loved by no one.

Jack loves all animals

Either Jack or Curiosity killed the cat, who is named Tuna.

Did Curiosity kill the cat?

Solution:

The argument can be represented in the predicate logic as:

Premise 1) ∀ x [∀ y Animal(y) => Loves(x,y)] => ∃ y. Loves(y,x)

Premise 2) ∀x [∃ y Animal(y) ^ Kills(x,y)] => [∀z ¬Loves(z,x)

Premise 3) ∀x. Animal(x) => Loves(Jack, x)

Premise 4) Kills(Jack, Tuna) v Kills(Curiosity, Tuna)

Cat(Tuna)

∀x Cat(x) => Animal(x) //(background knowledge)

Premise 5) Kills(Curiosity, Tuna) //Conclusion

Now convert the representation to CNF to create KB as follows:

1 a. Animal(F(x)) v Loves(G(x),x)

1 b. ¬Loves(x,F(x)) v Loves(G(x), x)

2. ¬Animal(y) v ¬Kills(x,y) v ¬Loves(z,x)

3. ¬Animal(x) v Loves(Jack,x)

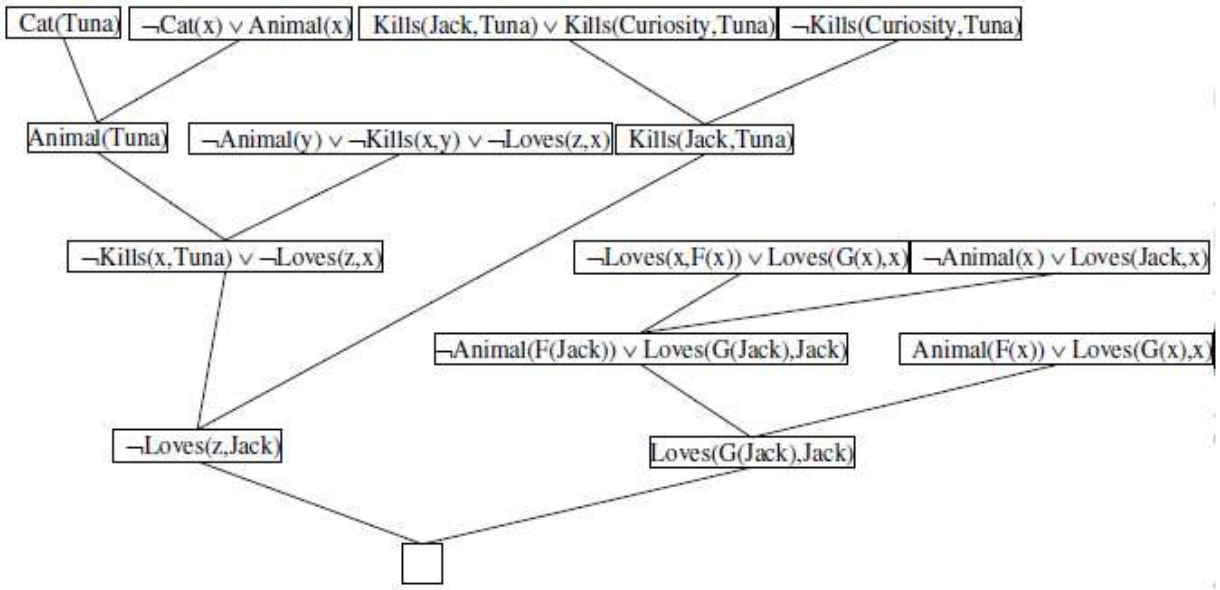
4 a. Kills(Jack, Tuna) v Kills(Curiosity, Tuna)

4 b. Cat(Tuna)

4 c. ¬Cat(x) v Animal(x)

5. ¬Kills(Curiosity, Tuna) //Negation of Conclusion

Now, use resolution as follows:



Example 4

Consider the following story of the "happy student":

Anyone passing his history exams and winning the lottery is happy. But anyone who studies or is lucky can pass all his exams. John did not study but he is lucky. Anyone who is lucky wins the lottery. Is John happy?

Solution:

First change the sentences to predicate form:

1) Anyone passing his history exams and winning the lottery is happy.

$\forall$ X (pass (X,history) $\wedge$ win (X,lottery) $\rightarrow$ happy (X))

2) Anyone who studies or is lucky can pass all his exams.

$\forall$ X $\forall$ Y (study (X) $\vee$ lucky (X) $\rightarrow$ pass (X,Y))

3) John did not study but he is lucky.

$\neg$ study (john) $\wedge$ lucky (john)

4) Anyone who is lucky wins the lottery.

$\forall$ X (lucky (X) $\rightarrow$ win (X,lottery))

Convert the sentences to Conjunctive Normal Form

1. ∀X: [pass(X, history) /\ win(X, lottery)] → happy(X)

   ≡ ∀X : ¬ [pass(X, history) /\ win(X, lottery)] ∨ happy(X)

   ≡ ∀X: ¬pass(X, history) ∨ ¬win(X, lottery) ∨ happy(X)

   ≡ ¬pass(X, history) ∨ ¬win(X, lottery) ∨ happy(X)

2. ∀X∀Y: [study(X) ∨ lucky(X)] → pass(X,Y)

   ≡ ∀X∀Y: ¬[study(X) ∨ lucky(X)] ∨ pass(X,Y)

   ≡∀X∀Y: [¬study(X) /\ ¬lucky(X)] ∨ pass(X,Y)

   ≡ ∀X∀Y: (¬study(X) ∨ pass(X,Y)) /\ (¬lucky(X) ∨ pass(X,Y))

   ≡ (¬study(X) ∨ pass(X,Y)) /\ (¬lucky(X) ∨ pass(X,Y))

Therefore the Clauses are:

¬study(X) ∨ pass(X,Y)

¬lucky(x) ∨ pass(X,Y)

3. ¬ study (john) ∧ lucky (john) //Already in CNF

Therefore the Clauses are:

¬study(John)

lucky(John)

4. ∀ X (lucky (X) → win (X,lottery))

   ≡∀x: ¬lucky(X) ∨ win(X,lottery)

   ≡¬lucky(X) ∨ win(X,lottery)

5. ¬happy(John) //Negation of conclusion, already in CNF

Now create a KB by standardizing variables apart as follows:

1) ¬pass(X, history) ∨ ¬win(X, lottery) ∨ happy(X)

2 a) ¬study(Y) ∨ pass(Y,Z)

2 b) ¬lucky(V) ∨ pass(V,W)

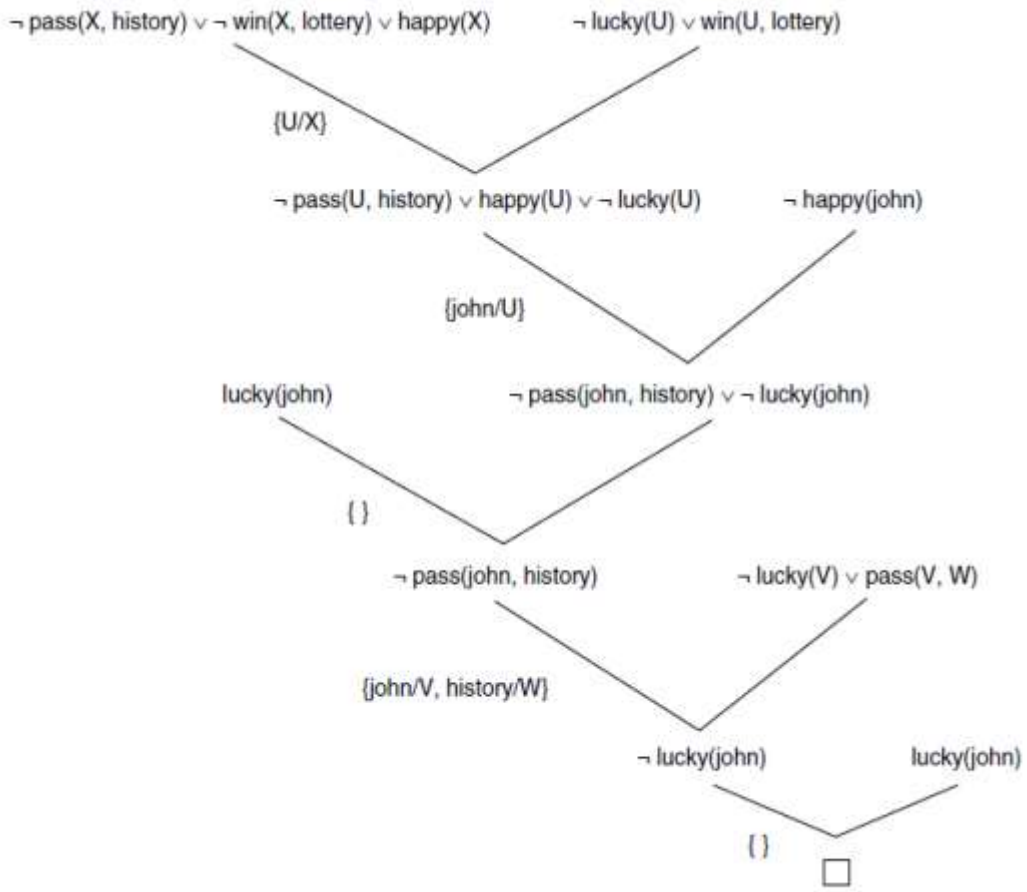
3 a) ¬study(John)

3 b) lucky(John)

4) ¬lucky(U) ∨ win(U,lottery)

5) ¬happy(John)

Now use resolution as follows:



Fuzzy Logic

In the FOPL, predicates are classified as either true or false. In our world, truth and falsehood often come with shades of gray. Some politicians believe that: "All taxes are bad." Is this posit true for a tax on cigarettes—a tax that could possibly cause some smokers to quit smoking, because of the increased price tag on their habit, and thereby live longer? Consider: "New Yorkers are polite." Most probably are, and these people probably will offer directions if they see you consulting a map; yet exceptions do exist. You might wish to maintain "New Yorkers are polite" is true to a certain extent. Fuzzy logic permits this latitude in truth-values. A logical expression can vary anywhere from false (0.0 degree of truth) to certainty (1.0 degree of truth). Fuzzy logic has found many applications in the control of modern conveniences. For example, the wash cycle of a washing machine should be longer if the clothes are especially dirty because dirtier clothes need to be washed longer. The shutter speed should be faster on a digital camera if it is sunny. "It is a sunny day" can vary anywhere from 0.0 to 1.0 in truth-value depending on whether it is nighttime, daytime and cloudy, or it is noon and there is not a cloud in the sky. Fuzzy logic controllers often have simpler logic design than their "non-fuzzy" counterparts. The founding father of fuzzy logic is Lofti Zadeh.

Semantic networks

Semantic networks were first introduced by Ross Quillian in 1968. They are a versatile form knowledge representation and are intended to be a model of how human associative memory works. They are a convenient formalism that helps to tell a story about an object, concept, event, or situation (represented by nodes—circles or boxes) and lines with arrows (arcs) representing the relationship between nodes.

"A semantic network is fundamentally a system for capturing, storing and transferring information that works much the same as (and is, in fact, modeled after) the human brain. It is robust, efficient and flexible. It is also the basis for many efforts to produce artificial intelligence. Semantic networks can grow to extraordinary complexity, necessitating a sophisticated approach to knowledge visualization, balancing the need for simplicity with the full expressive power of the network. Semantic networks may be traversed via concept list views, via their relations, or by retracing the user's history."

Semantic networks have certainly been useful for computer programmers and AI researchers as a form of knowledge representation, but elements of set membership and precision are missing. These would be more readily available from other forms of knowledge representation, such as logic. An example is shown in Figure below. We see that Mary owns Toby, which is a dog. A dog can be a pet, and so dogs are a subset of pets. So we see multiple inheritance, in that Mary owns Toby, and Mary owns a pet, of which Toby happens to be a member. Toby is a member of the class of objects called Dog. Her dog happens to be a pet, but not all dogs are pets. For example, Rotweilers are dogs that are pets for some people, and for other people they pose a menacing threat.

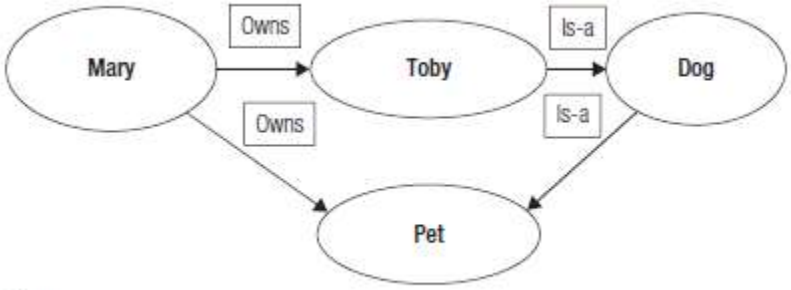


Figure
Toby is dog; Mary owns a Pet, but not all dogs are pets

Is-A relationships are frequently used in semantic networks, though they do not always represent what is true in the real world. Sometimes they might represent set membership and at other times they might mean equality. For example, a penguin Is-A bird, and we know birds can fly, but penguins do not fly. This is because although most birds (superclass) can fly, not all birds can fly (subclass).

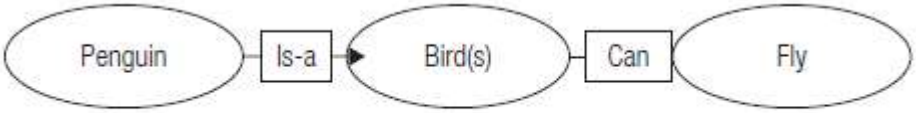


Figure
A penguin is a bird; birds can fly, but penguins can't fly.

Although semantic networks provide an intuitive way of representing the world, they do not represent many details about the real world that must be accounted for. Figure below illustrates a more complex semantic network representing a college. The college consists of students, academic departments, administration, and a library. The college might have a number of academic departments, one of which is Computer Science. Departments consist of faculty and staff. Students take classes, have records, and organize clubs. Students are required to do assignments and receive

grades from faculty who give assignments and give grades. Students and faculty are joined by classes, class codes, and grades.

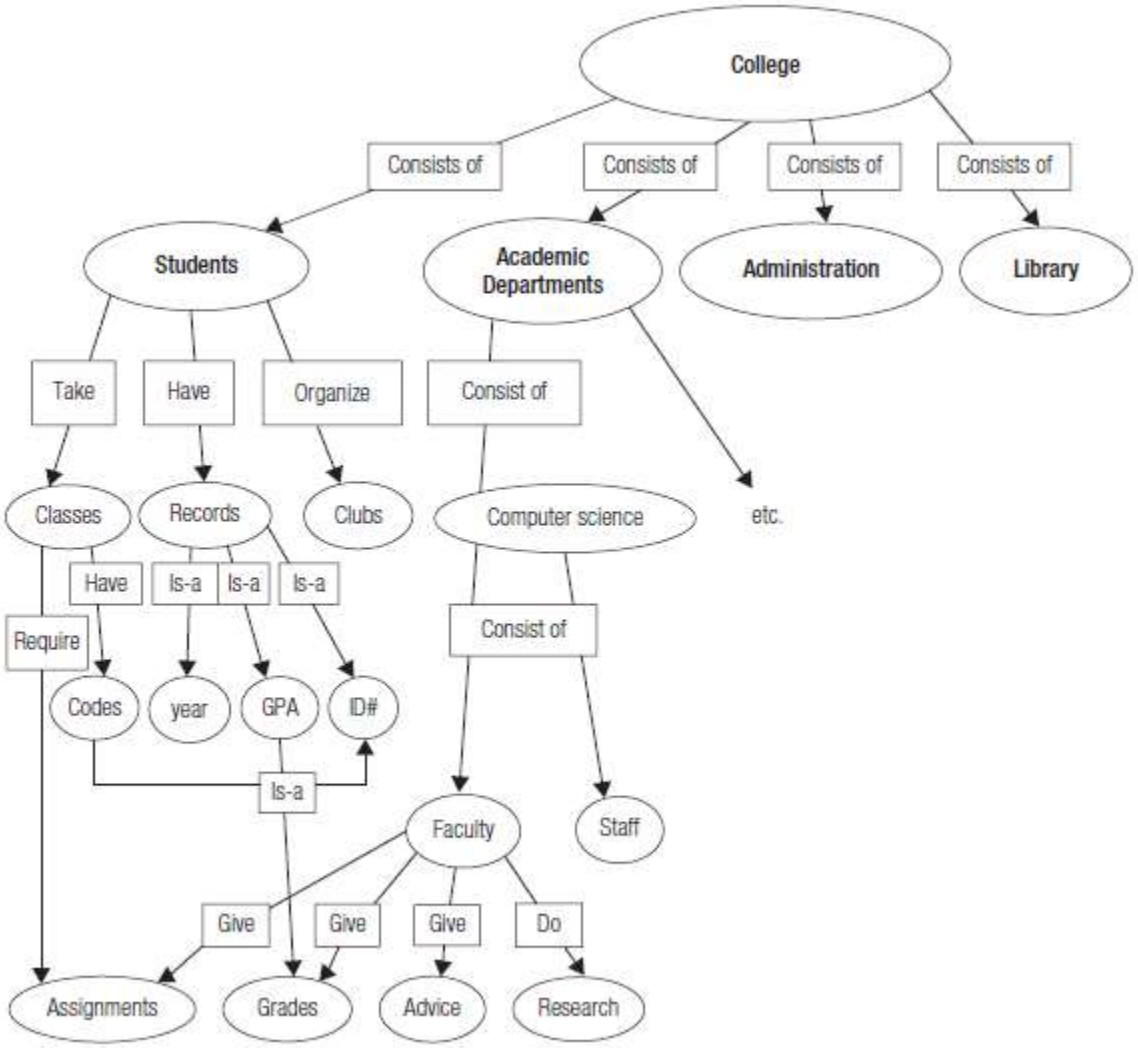


Figure
Semantic network representation of a college.

References

[1] S. Lucci and D. Kopec, ArtificiAl intelligence in the 21st century: A Living Introduction, Virginia: Mercury Learning and Information, 2016.

[2] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, New Jersey: Pearson Education, Inc., 2010.

[3] K. H. Rosen, Discrete mathematics and its applications, New York: McGraw-Hil, 2019.

[4] M. L. Ginsberg, Essentials of artificial intelligence, San Francisco, CA: Morgan Kaufmann Publishers, 1993.

[5] D. L. Poole and A. K. Mackworth, Artificial Intelligence: Foundations of Computational Agents, New York: CAMBRIDGE UNIVERSITY PRESS, 2010.