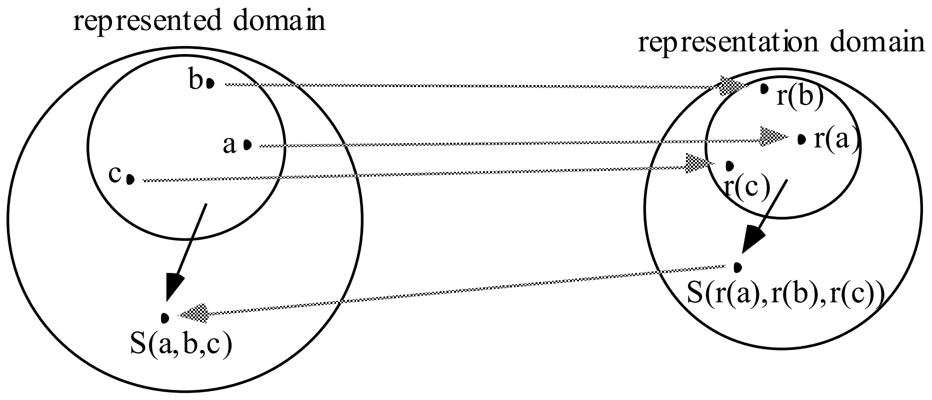
[Unit 4: Knowledge Representation]

Artificial Intelligence (MDS 556)

Master in Data Science

What is a representation?

A representation consists of two sets and a mapping between them. The elements of each set are objects, relations, classes, laws, actions. The first set is called the represented domain, and the second one is called the representation domain.



This mapping allows one to reason about the represented domain by performing reasoning processes in the representation domain and transferring the conclusions back into the represented domain (by using reverse mapping between the two sets).

The processes that manipulate the objects and the relationships in the representation domain are also part of the representation.

Knowledge representation is concerned with encoding the human knowledge into the form that is efficiently manipulated by the computer. Following are the kinds of knowledge that is represented in AI systems.

- ⇒ **Objects**: Guitar has Strings
- ⇒ **Events**: Renu played Guitar.
- ⇒ **Performance**: Dirt cleaned by vacuum cleaner
- ⇒ **Meta-Knowledge**: Renu knows that she can read street signs along the way to find where it is.
- ⇒ **Facts**: Truths about real world and what we represent

<u>**Knowledge:**</u>

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)

Knowledge consists of information that has been:

- interpreted,
- categorised,
- applied, experienced and revised.
-

In general, knowledge is more than just data, it consist of: facts, ideas, beliefs, heuristics, associations, rules, abstractions, relationships, customs.

Research literature classifies knowledge as follows:

Classification-based Knowledge	»	Ability to classify information
Decision-oriented Knowledge	»	Choosing the best option
Descriptive knowledge	»	State of some world (heuristic)
Procedural knowledge	»	How to do something
Reasoning knowledge	»	What conclusion is valid in what situation?
Assimilative knowledge	»	What its impact is?

Knowledge Representation

Knowledge representation (KR) is the study of how knowledge about the world can be represented and what kinds of reasoning can be done with that knowledge. Knowledge Representation is the method used to encode knowledge in Intelligent Systems.

Since knowledge is used to achieve intelligent behavior, the fundamental goal of knowledge representation is to represent knowledge in a manner as to facilitate inferencing (i.e. drawing conclusions) from knowledge. A successful representation of some knowledge must, then, be in a form that is *understandable* by humans, and must cause the system using the knowledge to *behave* as if it knows it.

Some issues that arise in knowledge representation from an AI perspective are:

- How do people represent knowledge?
- What is the nature of knowledge and how do we represent it?

- Should a representation scheme deal with a particular domain or should it be general purpose?
- How expressive is a representation scheme or formal language?
- Should the scheme be declarative or procedural?
-

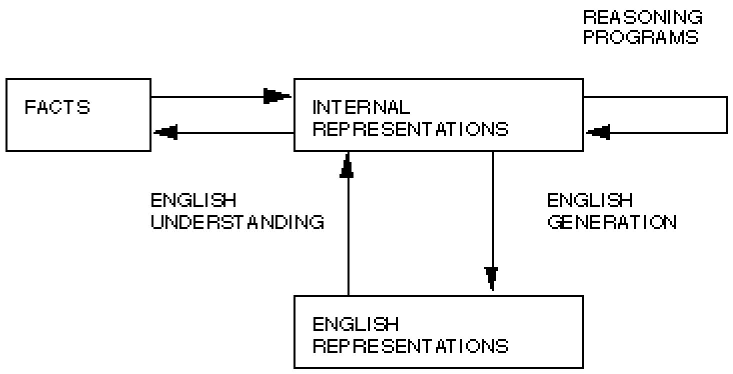


Fig: Two entities in Knowledge Representaion

For example: English or natural language is an obvious way of representing and handling facts. Logic enables us to consider the following fact: *spot is a dog* as *dog(spot)* We could then infer that all dogs have tails with: $\forall x$: *dog(x)* $\rightarrow$ *hasatail(x)*.

We can then deduce:

hasatail(Spot)

Using an appropriate backward mapping function the English sentence *Spot has a tail can be generated.*

Properties for Knowledge Representation Systems

The following properties should be possessed by a knowledge representation system.

Representational Adequacy
- the ability to represent the required knowledge;

Inferential Adequacy
- the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original;

Inferential Efficiency
- the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides;

Acquisitional Efficiency
- the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

General features of a representation

⇒ **Representational adequacy**: The ability to represent all of the kinds of knowledge that are needed in a certain domain.

⇒ **Inferential adequacy***:* The ability to represent all of the kinds of inferential procedures (procedures that manipulate the representational structures in such a way as to derive new structures corresponding to new knowledge inferred from old).

⇒ **Inferential efficiency***:* The ability to represent efficient inference procedures (for instance, by incorporating into the knowledge structure additional information that can be used to focus the attention of the inference mechanisms in the most promising directions).

⇒ **Acquisitional efficiency***:* The ability to acquire new information easily.

Knowledge Representation schemes can be classified as below:

⇒ Logical representation Scheme
⇒ Procedural representation Scheme
⇒ Network representation Scheme
⇒ Structured representation Scheme

Logical representation Scheme
It uses expression in the form of formal notation to represent the knowledge base. Propositional logic and predicate logic are examples of the logical representation Scheme. Give example your self.

Procedural representation Scheme
These representation schemes constitute rule based expert systems.

If type of light is sunlight then light is bright
If type of light is bulb then light is dim
If location is outdoor then light is sunlight
If location is indoor then light is bulb

Network representation Scheme
Network representation schemes represents knowledge in the form of a graph in which the nodes represent objects, situations, or events, and the arcs represent the relationships between them. Semantic networks, conceptual dependencies and conceptual graphs are examples of network representation.(Give example of semantic net yourself)

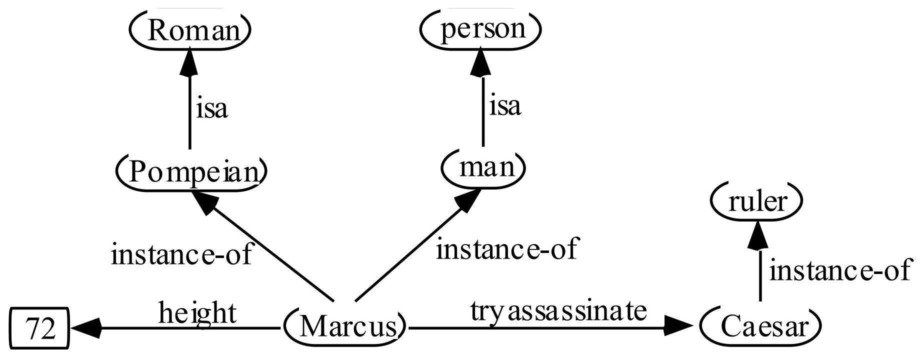
Structured representation Scheme
This is the extension of network representation schemes where nodes are complex data structures consisting of named slots with attached values. Scripts, frames and objects are examples of structured representation scheme.

Representing knowledge in semantic networks

The underlying idea of semantic networks is to represent knowledge in the form of a graph in which the nodes represent objects, situations, or events, and the arcs represent the relationships between them.

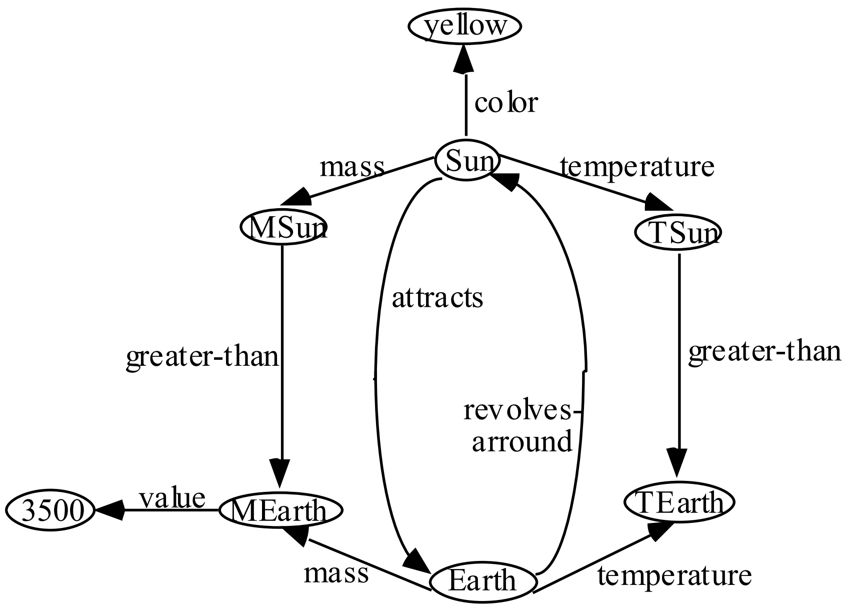
Semantic networks with binary relations

The following is an example of a semantic network with binary relations:



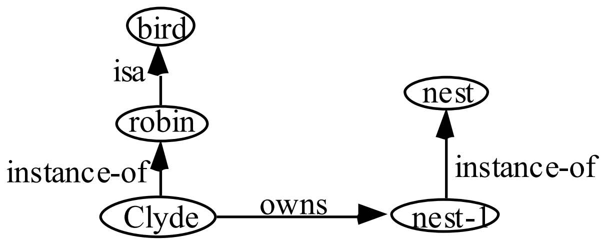
Some nodes represent individual objects (Marcus, Caesar) or constants (72).Other nodes represent classes of individuals (Pompeian, Roman, man, person, ruler). The relationship "instance-of" asserts that an individual is an instance (element) of a class. The relationship "isa" asserts that a class, (e.g. "man"), is a subclass of another class ("person"). "Man isa person" means that any element of man is also an element of person. That is, if someone is a man then he is a person. The relationship "tryassassinate", from Marcus to Caesar, states that Marcus tried to assassinate Caesar. The relationship "height", from Marcus to 72, states that the height of Marcus is 72. Marcus and Caesar are called instances because they represent individual objects.

The following is another example of a semantic network representing knowledge about our solar system. One may notice that this network states that the temperature of the Sun is greater than the temperature of the Earth, without indicating the values of these temperatures:



Representing non-binary predicates

Suppose we wish to represent the fact that Clyde, which is a robin, owns a nest. This may be encoded as follows:



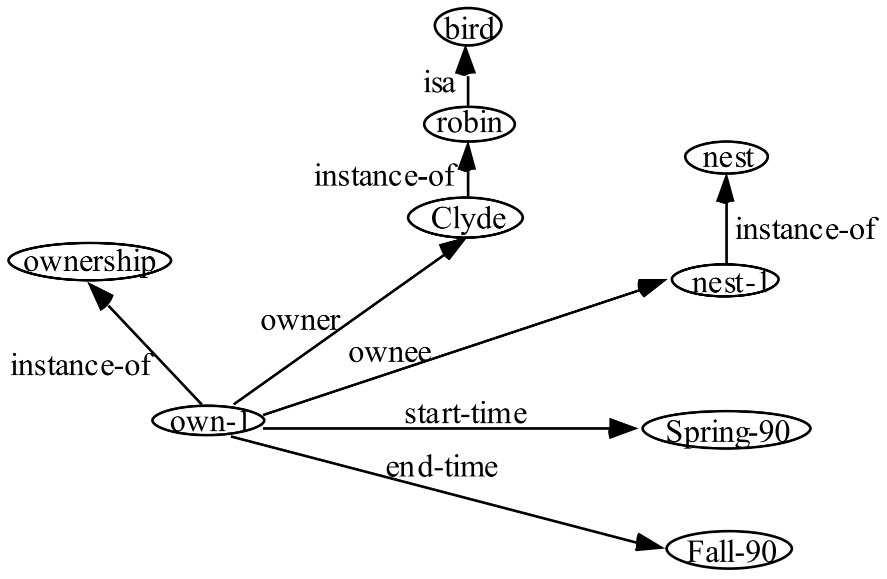
.

"nest-1" is the nest Clyde owns. It is an instance of "nest" which represents the class of all nests. Suppose one wants to encode the additional information that Clyde owned nest-1 from Spring 90 to Fall 90. This is impossible to do in the above network because the ownership situation is encoded as a link, and links, by their nature, can encode only binary relations.

The solution is to represent this ownership situation as a node characterized by four properties: the owner (Clyde), the object owned (nest-1), the time when the ownership started (Spring 90), and the time when the ownership ended (Fall 90). In this way one may represent the equivalent of a four-place predicate:

ownership(owner, ownee, start-time, end-time)

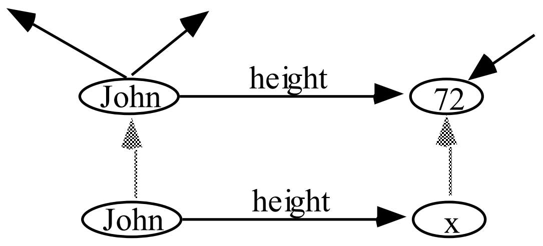
The corresponding semantic network is the following one:



Therefore, the semantic networks are not limited to representing only binary relationships

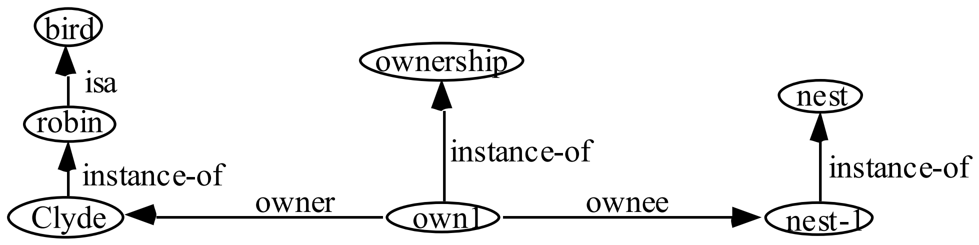
Network matching

A network fragment is constructed, representing a sought-for object or a query, and then matched against the network database to see if such an object exists. Variable nodes in the fragment are bound in the matching process to the values with which they match perfectly.

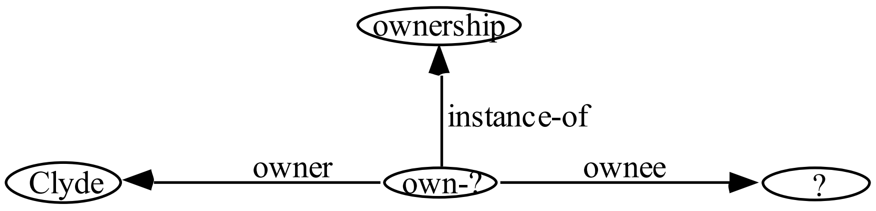


Question:
What is the height of John ?
Answer:
The height of John is 72.

As another example, let us suppose that we use the network given below as a database and suppose we wish to answer the question "What does Clyde own?".



We might construct the network in above figure which represents an instance of ownership in which Clyde is the owner. This fragment is then matched against the network database looking for an own node that has an owner link to Clyde. When it is found, the node that the ownee link points to is bound in the partial match and is the answer to the question.

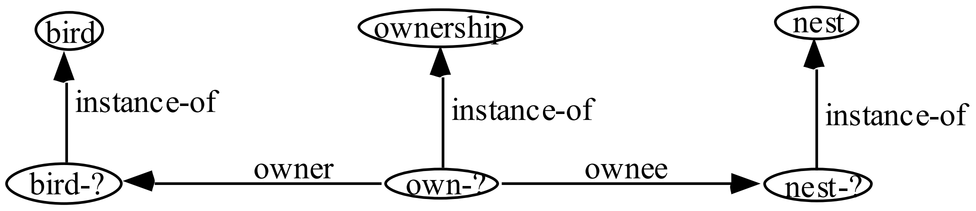


The matcher can make inferences during the matching process to create a network structure that is not explicitly present in the network.

For example, suppose we wish to answer the question

"Is there a bird who owns a nest?"

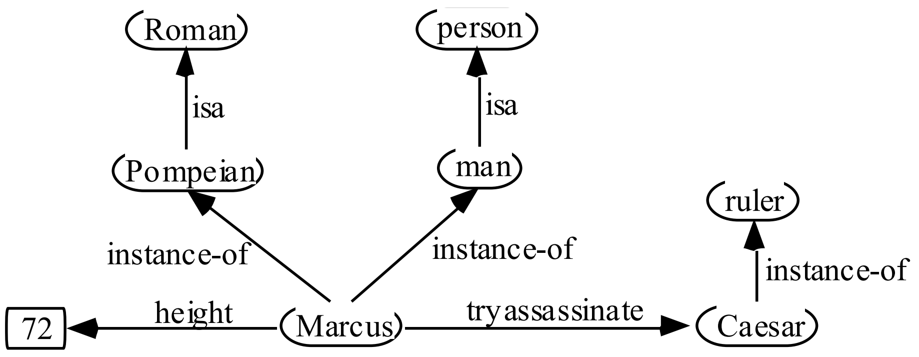
We could translate that question into the following network fragment:



Here, bird-?, nest-?, and own-? nodes represent the yet to be determined bird-owning-nest relation.

Notice that the query network fragment does not match the knowledge base exactly. The deduction procedure would have to construct an "instance-of" link from Clyde to bird to make the match possible. The matcher would bind bird-? to the node Clyde, own-? to own-1, and nest-? to nest-1, and the answer to the question would be "Yes, Clyde".

Consider the following network fragment:

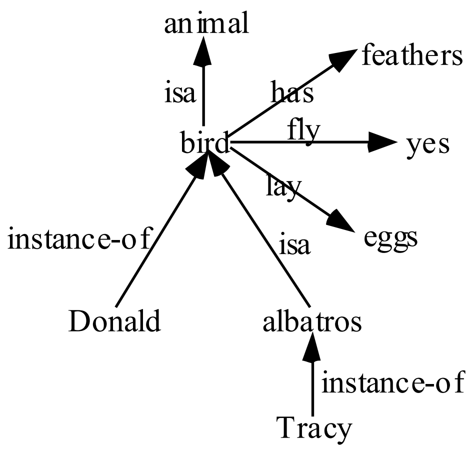


Exercises

1. Represent the following sentences into a semantic network.

Birds are animals.
Birds have feathers, fly and lay eggs.
Albatros is a bird.
Donald is a bird.
Tracy is an albatros.

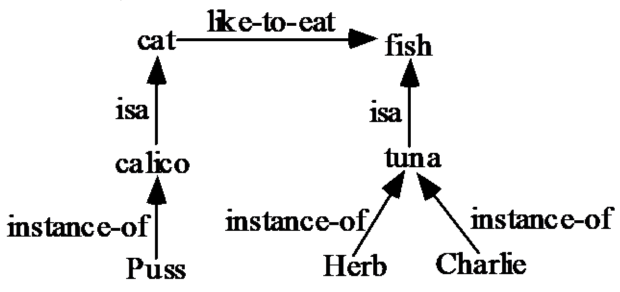
Solution:



2. Represent the following sentences into a semantic network:

Puss is a calico.
Herb is a tuna.
Charlie is a tuna.
All tunas are fishes.
All calicos are cats.
All cats like to eat all kinds of fishes.

Solution:



Object-based Approach: Frames

With this approach, knowledge may be represented in a data structure called a **frame**. A *frame* is a data structure containing typical knowledge about a concept or object (Marvin Minsky (mid 1970s)). A frame represents knowledge about real world things (or entities).

Each frame has **a name and slots**. **Slots** are the properties of the entity that has the name, and they have **values** or pointer to other frames (a table like data structure). A particular value may be:

- a default value
- an inherited value from a higher frame
- a procedure, called a daemon, to find a value
- a specific value, which might represent an exception.

When the slots of a frame are all filled, the frame is said to be **instantiated**, it now represents a specific entity of the type defined by the unfilled frame. Empty frames are sometimes called object prototypes

The idea of frame hierarchies is very similar to the idea of class hierarchies found in object-orientated programming. Frames are an application of the object-oriented approach to knowledge-based systems.

Disadvantages

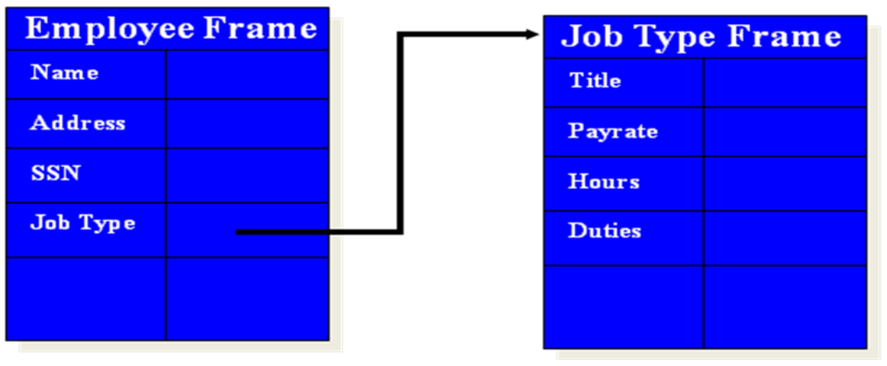
- complex
- reasoning (inferencing) is difficult
- explanation is difficult, expressive limitation

Advantages

- knowledge domain can be naturally structured [a similar motivation as for the O-O approach].
- easy to include the idea of default values, detect missing values, include specialised procedures and to add further slots to the frames

Examples:

(1.)



(2.)

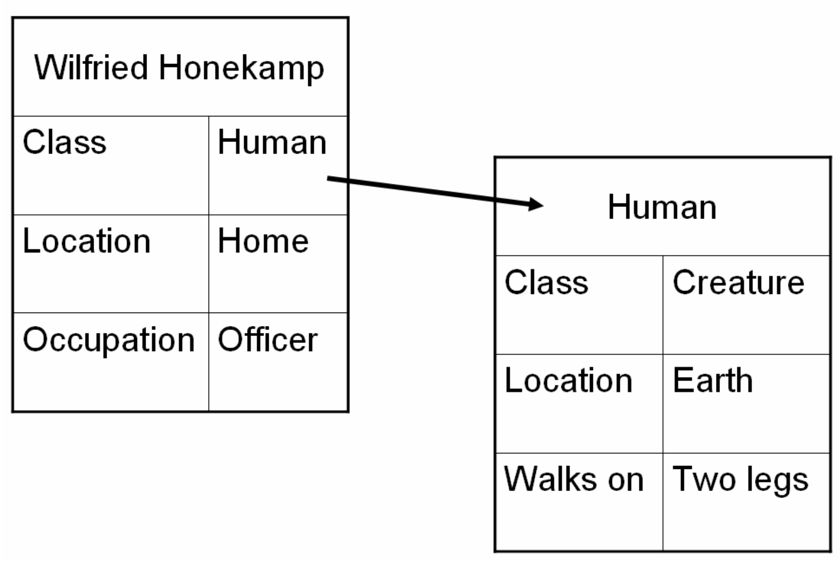


Fig: Two frames describing a human being

1) Create a frame of the person Ram who is a doctor. He is of 40. His wife name is Sita. They have two children Babu and Gita. They live in 100 kps street in the city of Delhi in India. The zip code is 756005.

```
(Ram
        (PROFESSION (VALUE Doctor))
        (AGE (VALUE 40))
        (WIFE (VALUE Sita))
        (CHILDREN (VALUE Bubu, Gita))
        (ADDRESS
                (STREET (VALUE 100 kps))
                (CITY(VALUE Delhi))
                (COUNTRY(VALUE India))
                (ZIP (VALUE 756005))))
```

Conceptual Dependencies

Conceptual Dependency originally developed to represent knowledge acquired from natural language input.

The goals of this theory are:

- To help in the drawing of inference from sentences.
- To be independent of the words used in the original input.
- That is to say: *For any 2 (or more) sentences that are identical in meaning there should be only one representation of that meaning.*

CD provides:

- a structure into which nodes representing information can be placed
- a specific set of primitives
- at a given level of granularity.

Sentences are represented as a series of diagrams depicting actions using both abstract and real physical situations.

- The agent and the objects are represented
- The actions are built up from a set of primitive acts which can be modified by tense.

Conceptual Dependency (CD) is a theory of how to represent the kind of knowledge about the events that is typically contained in natural language processing sentences. The goal is to represent the knowledge in a way that;

- facilitates drawing the inferences from the natural language sentences.
- is independent of the language in which natural language sentences were originally stated.

Because of the two concerns mentioned, the CD representation of a sentence is built not out of primitives corresponding to the words in the sentence, but rather out of conceptual primitives that can be combined to form the semantic meanings of the words in any particular language. The conceptual dependency theory was first developed by Schank in 1973 and was further more developed in 1975 by the same author. It has been implemented in a variety of programs that read and understand natural language text processing. *Unlike semantic nets which provide only a structure into which nodes representing information and a specific set of primitives, at a particular level of granularity, out of which representations of particular pieces of information can be constructed.*

In CD’s the symbols have the following meanings;

⟶ Arrows indicate the decision of dependency.

↔ Double arrow indicates two way link between actor and action

P indicates past tense.

ATRANS is one of the primitive symbol acts used by the CD theory. It indicates transfer of possession

O indicates the object case relation

R indicates the recipient case relation.

As a simple example of the way knowledge is represented in conceptual dependency theory, the event represented by the sentence.

In Conceptual dependency, the representations of actions are built from a set of primitive acts. Although there are slight differences in the exact set of primitives actions provided in the various sources on CD, a typical set is the following, which are taken from the Schanks conceptual dependency theory.

- RANS – transfer of an abstract relationship (give)
- PTRANS – transfer of the physical location of an object (go)
- PROPEL – application of physical force to an object (push)
- MOVE – movement of a body part by ots owner (kick)
- GRASP – grasping of an object by an actor (clutch)
- INGEST – ingestion of an object by an animal (eat)
- EXPEL – Expulsion of something from the body of an animal (cry)
- MTRANS – Transfer of mental information (tell)
- MBUILD – Building new information out of old (decide)
- SPEAK – production of sounds (say)
- ATTEND – Focusing of a sense organ toward a stimulus (listen)

Conceptual Dependency the symbols have the following meanings.

- Single arrows (→) indicate direction of dependency.
- Double arrows (↔) indicates two way link between action and actor.
- P indicates past tens.
- Primitives (ATRANS, PTRANS, PROPEL) indicates transfer of possession.
- O indicates object relation.
- R indicates recipient relation.

Conceptual Dependency introduced several interesting ideas:

(i) An internal representation that is language free, using primitive ACTs instead.
(ii) A small number of primitive ACTs rather than thousands.
(iii) Different ways of saying the same thing can map to the same internal representation.

There are some problems too:

(i) the expansion of some verbs into primitive ACT structures can be complex
(ii) graph matching in NP-hard.

Conceptual dependency theory offers a set of primitives conceptualizations from which the world of meaning is build.

These are equal and independent, They are :

ACTs : Actions
PPs : Object (picture procedure)
AAs : Modifiers of actions (action aiders)
PAs : Modifiers of objects i.e., PPS (picture aiders)
AA : Action aiders are properties or attributes of primitive actions.
PP : Picture produces are actors or physical objects that perform different acts or produces

1)	O	: Object case relationship
2)	R	: Recipient case relationship
3)	P	: Past
4)	F	: Future
5)	Nil	: Present
6)	T	: Transition
7)	Ts	: Start Transition
8)	Tf	: Finisher Transition
9)	K	: Continuing
10)	?	: Interrogative
11)	/	: Negative
12)	C	: Conditional

Also there are several rules in conceptual dependency

Also there are several rules in conceptual dependency

Rule 1: $\text{PP} \Longleftrightarrow \text{ACT}$

It describes the relationship between an actor and an event, he/she causes.

E.g. Ram ran

$\text{Ram} \overset{\text{P}}{\Longleftrightarrow} \text{PTRANS}$

Where P: Past Tense

Rule 2: $\text{PP} \Longleftrightarrow \text{PA}$

It describes the relationship between a PP and PA where the PA indicates one characteristics of PP. E.g. Ram is tall

$\text{Ram} \overset{\text{Nil}}{\Longleftrightarrow} \text{Tall}$ or $\text{Ram} \overset{\text{Nil}}{\Longleftrightarrow} \text{Height (> Average)}$

Rule 3: $\text{PP} \Longleftrightarrow \text{PP}$

It describes the relationship between two PPs where one PP is defined by other.

E.g. Ram is a doctor

$\text{Ram} \overset{\text{Nil}}{\Longleftrightarrow} \text{Doctor}$

Rule 4:

$$\begin{tikzcd} \text{PP} & \text{or} & \text{PA} \arrow[d] \\ \text{PA} \arrow[u] & & \text{PP} \end{tikzcd}$$

It describes the relationship between the PP and PA, where PA indicates one attributes of PP.

E.g. A nice boy is a doctor

$$\begin{tikzcd} \text{Boy} \arrow[r, Leftrightarrow, "\text{Nil}"] & \text{Doctor} \\ \text{Nice} \arrow[u] & \end{tikzcd}$$

7) My grandfather told me a story

$$\begin{tikzcd} \text{Grandfather} \arrow[r, Leftrightarrow, "P"] & \text{MTRANS} & \text{Story} \arrow[l, "O"'] \end{tikzcd}$$

8) Ira gave the man a dictionary

$$\begin{tikzcd} & & & \text{Ira} \\ \text{Ira} \arrow[r, Leftrightarrow, "P"] & \text{ATRANS} & \text{book} \arrow[l, "O"'] \arrow[ur] & \\ & & & \text{Man} \arrow[ul, dash] \end{tikzcd}$$

<u>**Objective Case (O)**</u>

1. **John took the book**

$$\begin{tikzcd} \text{PP[John]} \arrow[r, Leftrightarrow] & \text{ACT[took]} & \text{PP[book]} \arrow[l, "O"'] \end{tikzcd}$$

<u>**Recipient Case (R)**</u>

2. **John took the book from Mary**

$$\begin{tikzcd} & & & \text{PP[John]} \\ \text{PP} \arrow[r, Leftrightarrow] & \text{ACT} & \cdot \arrow[l, "R"'] \arrow[ur] & \\ & \text{PP} \arrow[u, "O"'] & & \text{<PP[Mary]} \arrow[ul, dash] \end{tikzcd}$$

Dept. of CSE, MBM Engg.

<u>Scripts</u>

A *script* is a structure that prescribes a set of circumstances which could be expected to follow on from one another. Scripts are used for representing knowledge about common sequences of events.

It is similar to a thought sequence or a chain of situations which could be anticipated.

It could be considered to consist of a number of slots or frames but with more specialised roles.

Scripts are beneficial because:

- Events tend to occur in known runs or patterns.
- Causal relationships between events exist.

- Entry conditions exist which allow an event to take place
- Prerequisites exist upon events taking place. *E.g.* when a student progresses through a degree scheme or when a purchaser buys a house.

The components of a script include:

Entry Conditions
-- these must be satisfied before events in the script can occur.
Results
-- Conditions that will be true after events in script occur.
Props
-- Slots representing objects involved in events.
Roles
-- Persons involved in the events.
Track
-- Variations on the script. Different tracks may share components of the same script.
Scenes
-- The sequence of *events* that occur. *Events* are represented in *conceptual dependency* form.

The classic example is the restaurant script:

Scene: A restaurant with an entrance and tables.
Actors: The diners, servers, chef.
Props: The table setting, menu, table, chair.
Acts: Entry, Seating, Ordering a meal, Serving a meal, Eating the meal, requesting the check, paying, leaving.

Advantages of Scripts:

- Ability to predict events.
- A single coherent interpretation may be build up from a collection of observations.

Disadvantages:

- Less general than frames.
- May not be suitable to represent all kinds of knowledge.

Example:-Script for going to the bank to withdraw money.

SCRIPT : Withdraw money

TRACK : Bank

PROPS : Money

Counter

Form

Token

Roles : P= Customer

E= Employee

C= Cashier

Entry conditions: P has no or less money.

The bank is open.

Results : P has more money.

Scene 1: Entering

P PTRANS P into the Bank

P ATTEND eyes to E

P MOVE P to E

Scene 2: Filling form

P MTRANS signal to E

E ATRANS form to P

P PROPEL form for writing

P ATRANS form to P

E ATRANS form to P

Scene 3: Withdrawing money

P ATTEND eyes to counter

P PTRANS P to queue at the counter

P PTRANS token to C

C ATRANS money to P

Scene 4: Exiting the bank

P PTRANS P to out of bank

Now let us look on a movie script description according to the above component.

a)	Script name	: Movie
b)	Track	: CINEMA HALL
c)	Roles	: Customer(c), Ticket seller(TS), Ticket Checker(TC), Snacks Sellers (SS)
d)	Probes	: Ticket, snacks, chair, money, Ticket, chart
e)	Entry condition	: The customer has money The customer has interest to watch movie.

6) **Scenes:**

a. SCENE-1 (Entering into the cinema hall)

C	PTRANS C into the cinema hall
C	ATTEND eyes towards the ticket counter
C	PTRANS C towards the ticket counters
C	ATTEND eyes to the ticket chart
C	MBUILD to take which class ticket
C	MTRANS TS for ticket
C	ATRANS money to TS
TS	ATRANS ticket to C

b. SCENE-2 (Entering into the main ticket check gate)

C	PTRANS C into the queue of the gate
C	ATRANS ticket to TC
TC	ATTEND eyes onto the ticket
TC	MBUILD to give permission to C for entering into the hall
TC	ATRANS ticket to C
C	PTRANS C into the picture hall.

c. **SCENE-3 (Entering into the picture hall)**

C	ATTEND eyes into the chair
TC	SPEAK where to sit
C	PTRANS C towards the sitting position
C	ATTEND eyes onto the screen

d. SCENE-4 (Ordering snacks)

C	MTRANS SS for snacks
SS	ATRANS snacks to C
C	ATRANS money to SS
C	INGEST snacks

e. SCENE-5 (Exit)

C	ATTEND eyes onto the screen till the end of picture
C	MBUILD when to go out of the hall
C	PTRANS C out of the hall

7) Result:

The customer is happy

The customer has less money

Rule Based System

A **rule based system / production system** (or **production rule system**) is a way to represent knowledge in some form of artificial intelligence, which consists primarily of a set of rules about behavior. These rules, termed **productions**, are a basic representation found useful in automated planning, expert systems and action selection. A production system provides the mechanism necessary to execute productions in order to achieve some goal for the system.

Productions are the set of rules that consist of two parts: **a sensory precondition (or "IF" statement) (antecedent) and an action (or "THEN") (consequent)**. If a production's precondition matches the current state of the world, then the production is said to be *triggered*. If a production's action is executed, it is said to have *fired*.

A rule based system contains a database of rules known as rule base, a working memory which maintains data about current state or knowledge, and a rule interpreter or inference engine. The rule interpreter must provide a mechanism for prioritizing productions when more than one is triggered.

The underlying idea of production systems is to represent knowledge in the form of condition-action pairs called production rules:

For Eg:

If the condition C is satisfied then the action A is appropriate.

Types of production rules

Situation-action rules

If it is raining then open the umbrella.

Inference rules

If Cesar is a man then Cesar is a person

Architecture of Rule Based System:

Short Term Memory:

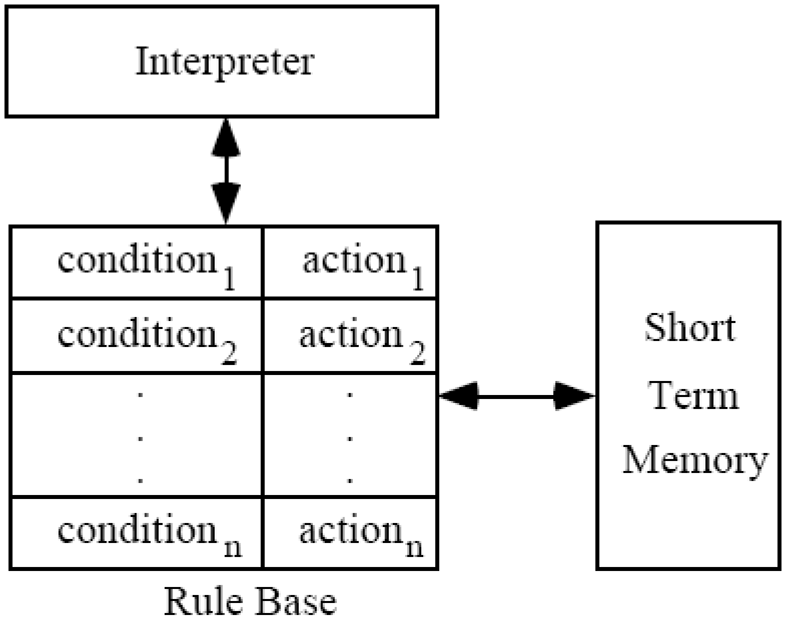
- Contains the description of the current state.

Set of Production Rules:

- Set of condition-action pairs and defines a single chunk of problem solving knowledge.

Interpreter:

- A mechanism to examine the short term memory and to determine which rules to fire (According to some strategies such as DFS, BFS, Priority, first-encounter etc)



The execution of a production system/rule based system can be defined as a series of recognize-act cycles: ***Match*** –memory contain matched against condition of production rules**,** this produces a subset of production called **conflict set**. ***Conflict resolution*** –one of the production in the conflict set is then selected**,** ***Apply the rule.***

Consider an example:

Problem: Sorting a string composed of letters a, b & c.

Short Term Memory: cbaca

Production Set:

1. ba → ab
2. ca → ac.
3. cb → bc

Interpreter: Choose one rule according to some strategy.

Iteration #	Memory	Conflict Set	Rule fired
0	cbaca	1, 2, 3	1
1	cabca	2	2
2	acbca	2, 3	2
3	acbac	1, 3	1
4	acabc	2	2
5	aacbc	3	3
6	aabcc	ø	halt

Formal logic-connectives:

In logic, a **logical connective** (also called a **logical operator**) is a symbol or word used to connect two or more sentences (of either a formal or a natural language) in a grammatically valid way, such that the compound sentence produced has a truth value dependent on the respective truth values of the original sentences.

Each logical connective can be expressed as a function, called a truth function. For this reason, logical connectives are sometimes called **truth-functional connectives**.

Commonly used logical connectives include:

- Negation (not) (¬ or ~)
- Conjunction (and) ($\wedge$, &, or ·)
- Disjunction (or) ($\vee$ or ∨)
- Material implication (if...then) ($\rightarrow$, $\Rightarrow$ or $\supset$)
- Biconditional (if and only if) (iff) (xnor) ($\leftrightarrow$, $\equiv$, or =)

For example, the meaning of the statements *it is raining* and *I am indoors* is transformed when the two are combined with logical connectives:

- It is raining **and** I am indoors ($P \wedge Q$)
- **If** it is raining, **then** I am indoors ($P \rightarrow Q$)
- It is raining **if** I am indoors ($Q \rightarrow P$)
- It is raining **if and only if** I am indoors ($P \leftrightarrow Q$)
- It is **not** raining ($\neg P$)

For statement *P* = *It is raining* and *Q* = *I am indoors*.

Truth Table:

A proposition in general contains a number of variables. For example ($P \vee Q$) contains variables P and Q each of which represents an arbitrary proposition. Thus a proposition takes different values depending on the values of the constituent variables. This relationship of the value of a proposition and those of its constituent variables can be represented by a table. It tabulates the value of a proposition for all possible values of its variables and it is called a truth table.

For example the following table shows the relationship between the values of P, Q and P ∨ Q:

OR		
P	**Q**	**(P ∨Q)**
F	F	F
F	T	T
T	F	T
T	T	T

Logic:

Logic is a formal language for representing knowledge such that conclusions can be drawn. **Logic** makes statements about the world which are true (or false) if the state of affairs it represents is the case (or not the case). 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, expressive, context insensitive, effective for inferences.

It has syntax, semantics, and proof theory.

Syntax: Describe possible configurations that constitute sentences.

Semantics: Determines what fact in the world, the sentence refers to i.e. the interpretation. Each sentence make claim about the world (meaning of sentence).Semantic property include truth and falsity.

Syntax is concerned with the rules used for constructing, or transforming the symbols and words of a language, as contrasted with the semantics of a language which is concerned with its meaning.

Proof theory (Inference method): set of rules for generating new sentences that are necessarily true given that the old sentences are true.

We will consider two kinds of logic: **propositional logic** and **first-order logic** or more precisely first-order **predicate calculus**. Propositional logic is of limited expressiveness but is useful to introduce many of the concepts of logic's syntax, semantics and inference procedures.

Entailment:

Entailment means that one thing follows from another:

$KB \models \alpha$

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

E.g., x + y =4 entails 4=x + y

Entailment is a relationship between sentences (i.e., syntax) that is based on semantics.

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.

Example:

P	$P \rightarrow Q$	Q
True	True	True
True	False	False
False	True	True
False	True	False

Therefore {P, P→Q} |= Q. Here, only row where both P and P→Q are True, Q is also True. Here, S= (P, P→Q} and P= {Q}.

Models

Logicians typically think in terms of models, in place of "possible world", which are formally structured worlds with respect to which truth can be evaluated.

m is a model of a sentence α if α is true in m.

M(α) is the set of all models of α.

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.

There are infinitely many tautologies. Examples include:

- $(A \vee \neg A)$("*A* or not *A*"), the law of the excluded middle. This formula has only one propositional variable, *A*. Any valuation for this formula must, by definition, assign *A* one of the truth values *true* or *false*, and assign ¬*A* the other truth value.
- $(A \rightarrow B) \Leftrightarrow (\neg B \rightarrow \neg A)$("if *A* implies *B* then not-*B* implies not-*A*", and vice versa), which expresses the law of contraposition.
- $((A \rightarrow B) \wedge (B \rightarrow C)) \rightarrow (A \rightarrow C)$("if *A* implies *B* and *B* implies *C*, then *A* implies *C*"), which is the principle known as syllogism.

The definition of ***tautology*** can be extended to sentences in predicate logic, which may contain quantifiers, unlike sentences of propositional logic. In propositional logic, there is no distinction between a tautology and a **logically valid formula**. In the context of predicate logic, many authors define a tautology to be a sentence that can be obtained by taking a tautology of propositional logic and uniformly replacing each propositional variable by a first-order formula (one formula per propositional variable). The set of such formulas is a proper subset of the set of logically valid sentences of predicate logic (which are the sentences that are true in every model).

There are also propositions that are always false such as $(P \wedge \neg P)$. Such a proposition is called a **contradiction**.

A proposition that is neither a tautology nor a contradiction is called a **contingency**. For example $(P \vee Q)$ is a contingency.

<u>Validity:</u>

The term **validity** in logic (also **logical validity**) is largely synonymous with logical truth, however the term is used in different contexts. Validity is a property of formulae, statements and arguments**. A logically valid argument is one where the conclusion follows from the premises. An invalid argument is where the conclusion does not follow from the premises.** A formula of a formal language is a valid formula if and only if it is true under every possible interpretation of the language.

Saying that an argument is valid is equivalent to saying that it is logically impossible that the premises of the argument are true and the conclusion false. A less precise but intuitively clear way of putting this is to say that in a valid argument **IF the premises are true, then the conclusion must be true.**

An argument that is not valid is said to be “invalid”.

An example of a valid argument is given by the following well-known syllogism:

All men are mortal.
Socrates is a man.
Therefore, Socrates is mortal.

What makes this a valid argument is not that it has true premises and a true conclusion, but the logical necessity of the conclusion, given the two premises.

The following argument is of the same logical form but with false premises and a false conclusion, and it is equally valid:

All women are cats.
All cats are men.
Therefore, all women are men.

This argument has false premises and a false conclusion. This brings out the hypothetical character of validity. What the validity of these arguments amounts to, is that it assures us the **conclusion must be true IF the premises are true**.

Thus, an argument **is valid if the premises and conclusion follow a logical form**. This essentially means that the conclusion logically follows from the premises. An argument is valid if and only if the truth of its premises entails the truth of its conclusion. It would be self-contradictory to affirm the premises and deny the conclusion

Deductive Reasoning:

Deductive reasoning, also called **Deductive logic**, is reasoning which constructs or evaluates deductive arguments. Deductive arguments are attempts to show that a conclusion necessarily follows from a set of premises. **A deductive argument is valid if the conclusion does follow necessarily from the premises, i.e., if the conclusion must be true provided that the premises are true**. A deductive argument is sound if it is valid AND its premises are true. Deductive arguments are valid or invalid, sound or unsound, but are never false or true.

An example of a deductive argument:

1. All men are mortal
2. Socrates is a man
3. Therefore, Socrates is mortal

The first premise states that all objects classified as 'men' have the attribute 'mortal'. The second premise states that 'Socrates' is classified as a man- a member of the set 'men'. The conclusion states that 'Socrates' must be mortal because he inherits this attribute from his classification as a man.

Deductive arguments are generally evaluated in terms of their *validity* and *soundness*. An argument is *valid* if it is impossible both for its premises to be true and its conclusion to be false. An argument can be valid even though the premises are false.

This is an example of a valid argument. The first premise is false, yet the conclusion is still valid.

All fire-breathing rabbits live on Mars
All humans are fire-breathing rabbits
Therefore, all humans live on Mars

This argument is valid but not *sound* In order for a deductive argument to be sound, the deduction must be valid and the premise must **all** be true.

Let’s take one of the above examples.

1. All monkeys are primates
2. All primates are mammals
3. All monkeys are mammals

This is a sound argument because it is actually true in the real world. The premises are true and so is the conclusion. They logically follow from one another to form a concrete argument that can’t be denied. Where validity doesn’t have to do with the actual truthfulness of an argument, soundness does.

A theory of deductive reasoning known as categorical or term logic was developed by Aristotle, but was superseded by propositional (sentential) logic and predicate logic.

Deductive reasoning can be contrasted with inductive reasoning. In cases of inductive reasoning, it is possible for the conclusion to be false even though the premises are true and the argument's form is cogent.

Well Formed Formula: (WFF)

It is a syntactic object that can be given a semantic meaning. A formal language can be considered to be identical to the set containing all and only its WFF.

A key use of WFF is in propositional logic and predicate logics such as first-order logic. In those contexts, a formula is a string of symbols φ for which it makes sense to ask "is φ true?", once any free variables in φ have been instantiated. In formal logic, proofs can be represented by sequences of WFF with certain properties, and the final WFF in the sequence is what is proven.

The well-formed formulas of **propositional calculus** are expressions such as $(A \wedge (B \vee C))$. Their definition begins with the arbitrary choice of a set *V* of propositional variables. The alphabet consists of the letters in *V* along with the symbols for the propositional connectives and parentheses "(" and ")", all of which are assumed to not be in *V*. The wffs will be certain expressions (that is, strings of symbols) over this alphabet.

The well-formed formulas are inductively defined as follows:

- Each propositional variable is, on its own, a WFF.
- If φ is a WFF, then ¬φ is a WFF.
- If φ and ψ are WFFs, and • is any binary connective, then (φ • ψ) is a WFF. Here • could be ∨, ∧, →, or ↔.

The WFF for **predicate calculus** is defined to be the smallest set containing the set of atomic WFFs such that the following holds:

1. $\neg\phi$ is a WFF when ϕ is a WFF
2. $(\phi \wedge \psi)$ and $(\phi \vee \psi)$ are WFFs when ϕ and ψ are WFFs;
3. $\exists x\ \phi$ is a WFF when *x* is a variable and ϕ is a WFF;
4. $\forall x\ \phi$ is a WFF when x is a variable and ϕ is a WFF (alternatively, $\forall x\ \phi$ could be defined as an abbreviation for $\neg\exists x\ \neg\phi$).

If a formula has no occurrences of $\exists x$ or $\forall x$, for any variable x, then it is called *quantifier-free*. An *existential formula* is a string of existential quantification followed by a quantifier-free formula.

Propositional Logic:

Propositional logic represents knowledge/ information in terms of propositions. Prepositions are facts and non-facts that can be true or false. Propositions are expressed using ordinary declarative sentences. Propositional logic is the simplest logic.

Syntax:

The syntax of propositional logic defines the allowable sentences. The atomic sentences- the indivisible syntactic elements- consist of single proposition symbol. Each such symbol stands for a proposition that can be true or false. We use the symbols like P1, P2 to represent sentences.

The complex sentences are constructed from simpler sentences using logical connectives. There are five connectives in common use:

$\neg$ (*negation*), ^ (*conjunction*), $\vee$ (*disjunction*), $\Rightarrow$ (*implication*), $\Leftrightarrow$ (*biconditional*)

The order of precedence in propositional logic is from (highest to lowest): $\neg$, ^ , $\vee$, $\Rightarrow$, $\Leftrightarrow$.

Propositional logic is defined as:

If S is a sentence, ¬S is a sentence (*negation*)
If S1 and S2 are sentences, S1 ^ S2 is a sentence (*conjunction*)
If S1 and S2 are sentences, S1 $\vee$ S2 is a sentence (*disjunction*)
If S1 and S2 are sentences, S1 $\Rightarrow$ S2 is a sentence (*implication*)
If S1 and S2 are sentences, S1 $\Leftrightarrow$ S2 is a sentence (*biconditional*)

Formal grammar for propositional logic can be given as below:

Sentence	$\rightarrow$ AutomicSentence | ComplexSentence
AutomicSentence	$\rightarrow$ True | False | Symbol
Symbol	$\rightarrow$ P | Q | R …………
ComplexSentence	$\rightarrow$ $\neg$Sentence
	| (Sentence ^ Sentence)
	| (Sentence $\vee$ Sentence)
	| (Sentence $\Rightarrow$ Sentence)
	| (Sentence $\Leftrightarrow$ Sentence)

Semantics:

Each model specifies true/false for each proposition symbol

Rules for evaluating truth with respect to a model:

$\neg$S is true if, S is false
S1 ^ S2 is true if, S1 is true and S2 is true
S1 $\vee$ S2 is true if, S1 is true or S2 is true
S1 $\Rightarrow$ S2 is true if, S1 is false or S2 is true
S1 $\Leftrightarrow$ S2 is true if, S1 $\Rightarrow$ S2 is true and S2 $\Rightarrow$ S1 is true

Truth Table showing the evaluation of semantics of complex sentences:

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

Logical equivalence:

Two sentences α and ß are *logically equivalent* (α ≡ ß) iff true they are true in same set of models or Two sentences α and ß are *logically equivalent* (α ≡ ß) iff α \|= ß and ß \|= α.

$$
\begin{array}{rcll}
(\alpha \land \beta) & \equiv & (\beta \land \alpha) & \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv & (\beta \lor \alpha) & \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv & (\alpha \land (\beta \land \gamma)) & \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv & (\alpha \lor (\beta \lor \gamma)) & \text{associativity of } \lor \\
\neg(\neg\alpha) & \equiv & \alpha & \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\beta \Rightarrow \neg\alpha) & \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\alpha \lor \beta) & \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv & (\neg\alpha \lor \neg\beta) & \text{de Morgan} \\
\neg(\alpha \lor \beta) & \equiv & (\neg\alpha \land \neg\beta) & \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv & ((\alpha \land \beta) \lor (\alpha \land \gamma)) & \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv & ((\alpha \lor \beta) \land (\alpha \lor \gamma)) & \text{distributivity of } \lor \text{ over } \land
\end{array}
$$

<u>Validity:</u>

A sentence is *valid* if it is true in all models,

e.g., *True*, A∨¬A, A ⇒ A, (A ∧ (A ⇒ B)) ⇒ B

Valid sentences are also known as tautologies. Every valid sentence is logically equivalent to True

<u>Satisfiability:</u>

A sentence is *satisfiable* if it is true in *some* model

– e.g., A ∨ B, C

A sentence is *unsatisfiable* if it is true in *no* models

– e.g., A¬∧A

A propositional logic is said to be **satisfiable** if its either a **tautology** or **contingency**. Hence if a logic is a contradiction then it is said to be **unsatisfiable**. By **contingency** we mean that logic can be true or false i.e. nothing can be said for sure about the logic.

<u>Inference rules in Propositional Logic</u>

Modus Ponens

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

And-elimination

$$\frac{\alpha \land \beta}{\alpha}$$

Monotonicity: the set of entailed sentences can only increase as information is added to the knowledge base.

For any sentence α and β if KB $\models \alpha$ then KB $\wedge\, \beta \models \alpha$.

Inference using Resolution

Unit resolution rule:

Unit resolution rule takes a clause – a disjunction of literals – and a literal and produces a new clause. Single literal is also called unit clause.

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

Where l_i and m are complementary literals

Generalized resolution rule:

Generalized resolution rule takes two clauses of any length and produces a new clause as below.

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

For example:

$$\frac{\ell_1 \vee \ell_2, \qquad \neg\ell_2 \vee \ell_3}{\ell_1 \vee \ell_3}$$

Resolution Uses CNF (Conjunctive normal form)

- **Conjunction of disjunctions of literals (clauses)**

The resolution rule is sound:

- Only entailed sentences are derived

Resolution is complete in the sense that it can always be used to either confirm or refute a sentence.

Conversion to CNF:

A sentence that is expressed as a conjunction of disjunctions of literals is said to be in conjunctive normal form (CNF). A sentence in CNF that contains only k literals per clause is said to be in k-CNF.

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

Example: Let's illustrate the conversion to CNF by using an example.

$B \Leftrightarrow (A \vee C)$

- Eliminate ⇔, replacing $\alpha \Leftrightarrow ß$ with $(\alpha \Rightarrow ß) \wedge (ß \Rightarrow \alpha)$.
  - $(B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B)$
- Eliminate ⇒, replacing $\alpha \Rightarrow ß$ with $\neg \alpha \vee ß$.
  - $(\neg B \vee A \vee C) \wedge (\neg(A \vee C) \vee B)$
- Move ¬ inwards using de Morgan's rules and double-negation:
  - $(\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B)$
- Apply distributivity law (∧ over ∨) and flatten:
  - $(\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B)$

Resolution algorithm

- Convert KB into CNF
- Add negation of sentence to be entailed into KB i.e. (KB ∧ ¬α)
- Then apply resolution rule to resulting clauses.
- The process continues until:
  - There are no new clauses that can be added
    Hence *KB* **does not** entail α
  - Two clauses resolve to entail the empty clause.
    Hence *KB* **does** entail α

Example: Consider the knowledge base given as: *KB* = (B ⇔ (A∨ C)) ∧ ¬B
Prove that ¬A can be inferred from above KB by using resolution.

<u>Solution:</u>

At first, convert KB into CNF

B ⇒ (A ∨ C)) ∧ ((A ∨ C) ⇒ B) ∧¬ B

(¬B ∨ A ∨ C) ∧ (¬(A ∨ C) ∨ B) ∧¬ B

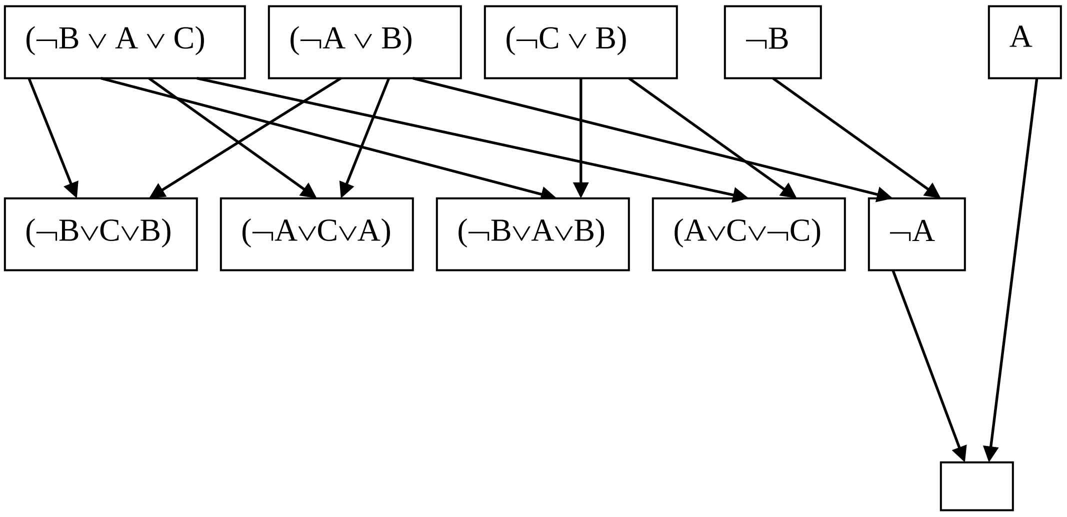
(¬B ∨ A ∨ C) ∧ ((¬A ∧ ¬C) ∨ B) ∧¬ B

(¬B ∨ A ∨ C) ∧ (¬A ∨ B) ∧ (¬C ∨ B) ∧¬ B

Add negation of sentence to be inferred from KB into KB

Now KB contains following sentences all in CNF
(¬B ∨ A ∨ C)
(¬A ∨ B)
(¬C ∨ B)
¬ B
A (negation of conclusion to be proved)

Now use Resolution algorithm



Resolution: More Examples

1. **Given a KB= {($G \vee H$) → ($\neg J \wedge \neg K$), G}. Show that KB entails $\neg J$.**

Solution:

Clausal form of ($G \vee H$) → ($\neg J \wedge \neg K$) is

$\{\neg G \vee \neg J,\ \neg H \vee \neg J,\ \neg G \vee \neg K,\ \neg H \vee \neg K\}$

1. $\neg G \vee \neg J$ [Premise]
2. $\neg H \vee \neg J$ [Premise]
3. $\neg G \vee \neg K$ [Premise]
4. $\neg H \vee \neg K$ [Premise]
5. G [Premise]
6. J [$\neg$ Conclusion]
7. $\neg G$ [1, 6 Resolution]
8. _ [5, 7 Resolution]

Hence KB entails $\neg J$

2. **Consider a KB= {$P \rightarrow \neg Q$, $\neg Q \rightarrow R$}. Show that $P \rightarrow R$ can be inferred using resolution.**

Solution:

1. $\neg P \vee \neg Q$ [Premise]
2. $Q \vee R$ [Premise]
3. P [$\neg$ Conclusion]
4. $\neg R$ [$\neg$ Conclusion]
5. $\neg Q$ [1, 3 Resolution]
6. R [2, 5 Resolution]
7. _ [4, 6 Resolution]

Hence, KB $\vdash P \rightarrow R$

3. Given the following hypotheses:

1. If it rains, Joe brings his umbrella (r -> u)
2. If Joe has an umbrella, he doesn't get wet (u -> ¬ w)
3. If it doesn't rain, Joe doesn't get wet (¬ r -> ¬ w)

Prove that Joes doesn't get wet (¬ w)

We first put each hypothesis in CNF:

1. r -> u = (¬ r ∨ u)
2. u -> ¬ w = (¬ u ∨ ¬ w)
3. ¬ r -> ¬ w = (r ∨ ¬ w)

Proof (about Joe and his umbrella) using proof by contradiction with resolution:

1. ¬ r ∨ u	Premise
2. ¬ u ∨ ¬ w	Premise
3. r ∨ ¬ w	Premise
4. w	Negation of conclusion
5. ¬ r ∨ ¬ w	Using resolution on 1, 2
6. ¬ w ∨ ¬ w	Using resolution on 3, 5
7. ¬ w	Using resolution on 4, 6
8. FALSE	Using resolution on 4, 7

4. Consider a KB containing following statements:

If it is sunny and warm day you will enjoy.
If it is warm and pleasant day you will do strawberry picking
If it is raining then you will not do strawberry picking
If it is raining you will get wet
It is warm day
It is raining
It is sunny.

Now infer that you will not do strawberry picking.

Part(II) : Propositional Statements

(1) enjoy ← sunny ∧ warm

(2) strawberry_picking ← warm ∧ pleasant

(3) ~strawberry_picking ← raining

(4) wet ← raining

(5) warm

(6) raining

(7) sunny

Part(III) : CNF of Part(II)

(1) (enjoy ∨~sunny∨~warm) ∧

(2) (strawberry_picking ∨~warm∨~pleasant) ∧

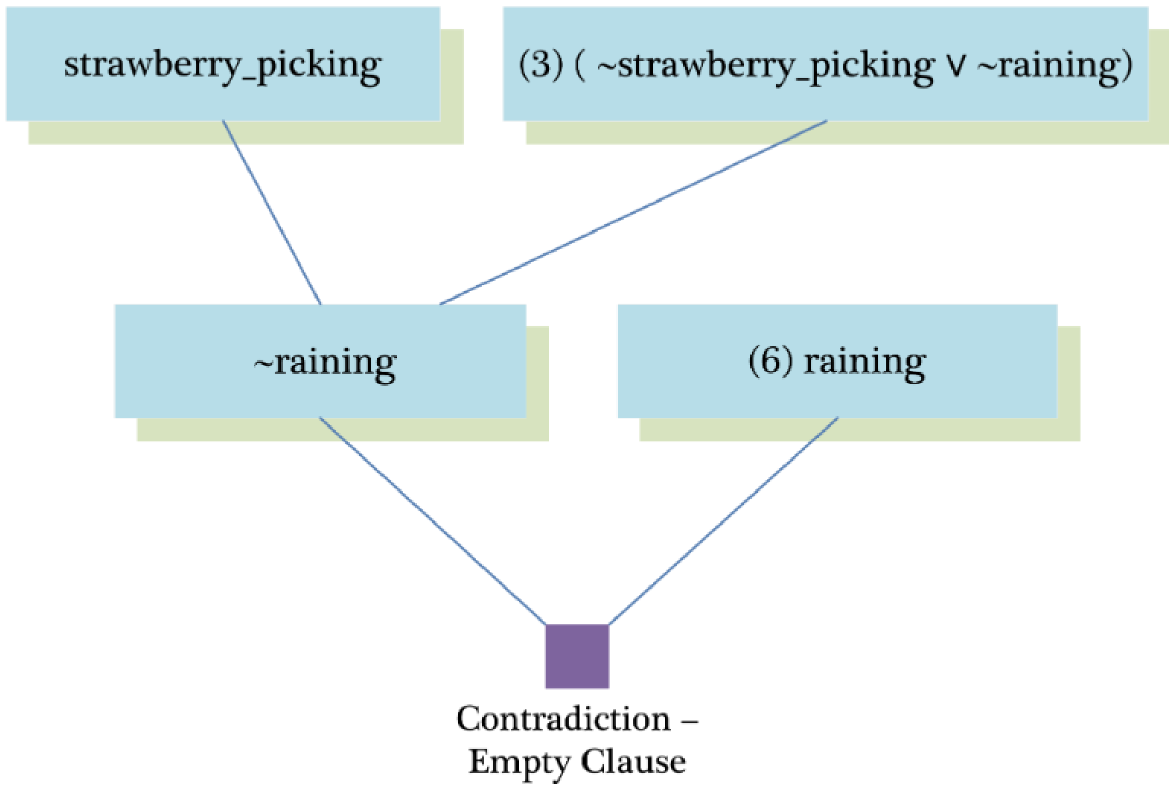
(3) (~strawberry_picking ∨~raining) ∧

(4) (wet ∨~raining) ∧

(5) (warm) ∧

(6) (raining) ∧

(7) (sunny)



Example

If it is raining then it is not cold. John is not wearing a coat if it is not raining. It is cold. There are clouds in the sky or it is not raining. The moon is made of cheese and there are clouds in the sky. Transform the given statements to propositional statements. Using resolution infer that it is raining and John is not wearing a coat.

Forward and backward chaining

The completeness of resolution makes it a very important inference model. But in many practical situations full power of resolution is not needed. Real-world knowledge bases often contain only clauses of restricted kind called **Horn Clause.** A Horn clauses is disjunction of literals with at most one positive literal
Three important properties of Horn clause are:

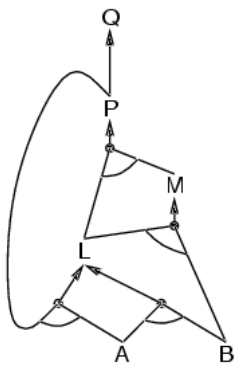
- ✓ Can be written as an implication
- ✓ Inference through forward chaining and backward chaining.
- ✓ Deciding entailment can be done in a time linear size of the knowledge base.

Forward chaining:

Idea: fire any rule whose premises are satisfied in the *KB*,

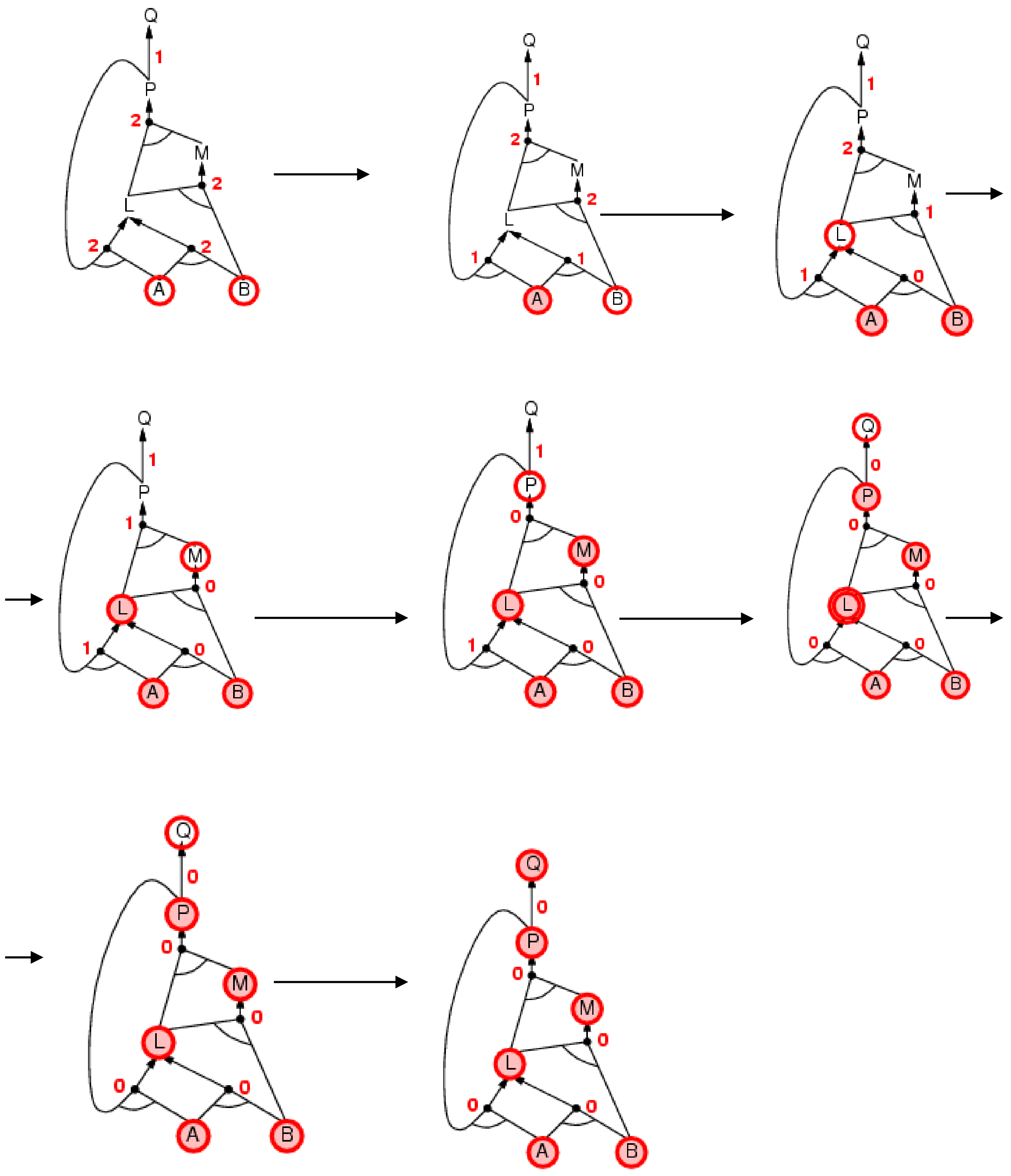
- add its conclusion to the *KB*, until query is found

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Prove that Q can be inferred from above KB

Solution:


Q
P
M
L
A
B

Backward chaining:

Idea: work backwards from the query *q*: to prove *q* by BC,

Check if *q* is known already, or

Prove by BC all premises of some rule concluding *q*

For example, for above KB (as in forward chaining above)

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Prove that Q can be inferred from above KB

Solution:

We know $P \Rightarrow Q$, try to prove P

$L \wedge M \Rightarrow P$

Try to prove L and M

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

Try to prove B, L and A and P

A and B is already known, since $A \wedge B \Rightarrow L$, L is also known

Since, $B \wedge L \Rightarrow M$, M is also known

Since, $L \wedge M \Rightarrow P$, p is known, hence the **proved.**

<u>First-Order Logic</u>

Pros and cons of propositional logic

- Propositional logic is declarative
- Propositional logic allows partial/disjunctive/negated information
  - (unlike most data structures and databases)
- Propositional logic is compositional:
  - meaning of $B \wedge P$ is derived from meaning of B and of P
- Meaning in propositional logic is context-independent
  - (unlike natural language, where meaning depends on context)
- Propositional logic has very limited expressive power
  - (unlike natural language)

Propositional logic assumes the world contains facts, whereas first-order logic (like natural language) assumes the world contains:

- Objects: people, houses, numbers, colors, baseball games, wars, …
- Relations: red, round, prime, brother of, bigger than, part of, comes between,…
- Functions: father of, best friend, one more than, plus, …

Logics in General

The primary difference between PL and FOPL is their ontological commitment:

Ontological Commitment: What exists in the world — TRUTH

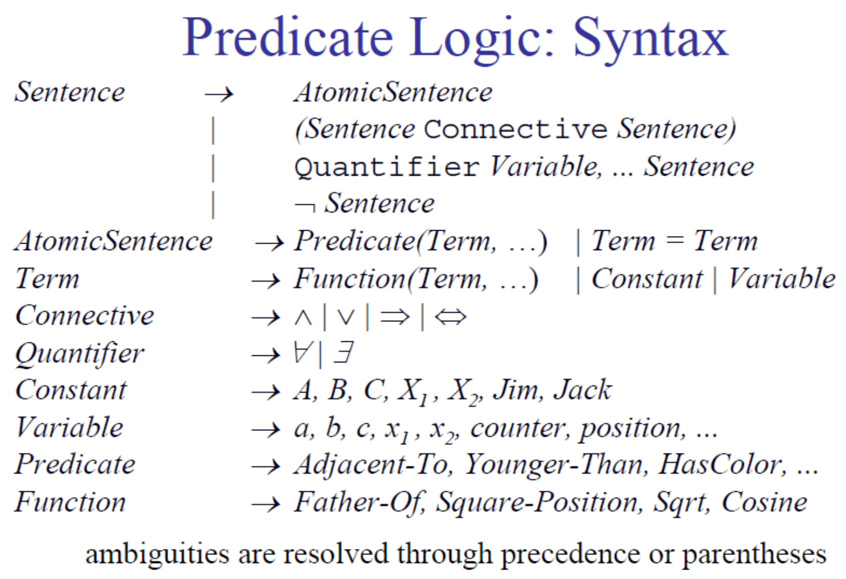
- PL: facts hold or do not hold.
- FL : objects with relations between them that hold or do not hold

Another difference is:

Epistemological Commitment: What an agent believes about facts — BELIEF

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0,1]$
Fuzzy logic	degree of truth $\in [0,1]$	known interval value

FOPL: Syntax



Predicate Logic: Syntax

Sentence	→	*AtomicSentence*
	|	*(Sentence* `Connective` *Sentence)*
	|	`Quantifier` *Variable, ... Sentence*
	|	¬ *Sentence*
AtomicSentence	→	*Predicate(Term, ...)* | *Term = Term*
Term	→	*Function(Term, ...)* | *Constant* | *Variable*
Connective	→	$\wedge \mid \vee \mid \Rightarrow \mid \Leftrightarrow$
Quantifier	→	$\forall \mid \exists$
Constant	→	*A, B, C,* X_1, X_2, *Jim, Jack*
Variable	→	*a, b, c,* x_1, x_2, *counter, position, ...*
Predicate	→	*Adjacent-To, Younger-Than, HasColor, ...*
Function	→	*Father-Of, Square-Position, Sqrt, Cosine*

ambiguities are resolved through precedence or parentheses

Representing knowledge in first-order logic

The objects from the real world are represented by constant symbols (a,b,c,...). For instance, the symbol "Tom" may represent a certain individual called Tom.

Properties of objects may be represented by predicates applied to those objects (P(a), ...): e.g "**male(Tom)**" represents that **Tom is a male**.

Relationships between objects are represented by predicates with more arguments: "**father(Tom, Bob)**" represents the fact that **Tom is the father of Bob**.

The value of a predicate is one of the boolean constants T (i.e. true) or F (i.e. false)."father(Tom, Bob) = T" means that the sentence "Tom is the father of Bob" is true. "father(Tom, Bob) = F" means that the sentence "Tom is the father of Bob" is false.

Besides constants, the arguments of the predicates may be functions (f,g,...) or variables (x,y,...).

Function symbols denote mappings from elements of a domain (or tuples of elements of domains) to elements of a domain. For instance, weight is a function that maps objects to their weight: weight (Tom) = 150.Therefore the predicate **greater-than (weight (Bob), 100) means that the weight of Bob is greater than 100**. The arguments of a function may themselves be functions.

Variable symbols represent potentially any element of a domain and allow the formulation of general statements about the elements of the domain.

The quantifier's ∀ and ∃ are used to build new formulas from old ones.
"∃x P(x)" expresses that there is at least one element of the domain that makes P(x) true.
"∃x mother(x, Bob)" means that there is x such that x is mother of Bob or, otherwise stated, Bob has a mother.
"∀x P(x)" expresses that for all elements of the domain P(x) is true.

Quantifiers

Allows us to express properties of collections of objects instead of enumerating objects by name. Two quantifiers are:

Universal: "for all" ∀
Existential: "there exists" ∃

Universal quantification:

∀*<Variables> <sentence>*

Eg: Everyone at UAB is smart:
$\forall x\ At(x,UAB) \Rightarrow Smart(x)$

∀x *P* is true in a model *m* iff *P* is true for all *x* in the model

Roughly speaking, equivalent to the conjunction of instantiations of *P*

At(KingJohn,UAB) ⇒ Smart(KingJohn) ∧ At(Richard,UAB) ⇒ Smart(Richard)∧At(UAB,UAB) ⇒ Smart(UAB)∧ ...

Typically, ⇒ is the main connective with ∀

- A universally quantifier is also equivalent to a set of implications over all objects

Common mistake: using ∧ as the main connective with ∀:

∀x At(x, UAB) ∧ Smart(x)

Means "Everyone is at UAB and everyone is smart"

Existential quantification

∃<*variables*> <*sentence*>

Someone at UAB is smart:

∃*x* At(x, UAB) ∧ Smart(x)

∃*x P* is true in a model *m* iff *P* is true for at least one *x* in the model

Roughly speaking, equivalent to the disjunction of instantiations of *P*

At(KingJohn,UAB) ∧ Smart(KingJohn)∨At(Richard,UAB) ∧ Smart(Richard) ∨At(UAB, UAB) ∧ Smart(UAB) ∨ ...

Typically, ∧ is the main connective with ∃

Common mistake: using ⇒ as the main connective with ∃:

∃*x* At(x, UAB) ⇒ Smart(x) is true even if there is anyone who is not at UAB!

FOPL: Semantic

An interpretation is required to give semantics to first-order logic. The interpretation is a non-empty "domain of discourse" (set of objects). The truth of any formula depends on the interpretation.

The interpretation provides, for each:

constant symbol an object in the domain
function symbols a function from domain tuples to the domain
predicate symbol a relation over the domain (a set of tuples)

Then we define:

universal quantifier $\forall x P(x)$ is True iff $P(a)$ is True for all assignments of domain elements a to x

existential quantifier $\exists x P(x)$ is True iff $P(a)$ is True for at least one assignment of domain element a to x

FOPL: Inference (Inference in first-order logic)

First order inference can be done by converting the knowledge base to PL and using propositional inference.

- How to convert universal quantifiers?
  - Replace variable by ground term.
- How to convert existential quantifiers?
  - Skolemization.

Universal instantiation (UI)

Substitute ground term (term without variables) for the variables.

For example consider the following KB

$\forall$ x King (x) $\wedge$ Greedy (x) $\Rightarrow$ Evil(x)
King (John)
Greedy (John)
Brother (Richard, John)

It's UI is:

King (John) $\wedge$ Greedy (John) $\Rightarrow$ Evil(John)
King (Richard) $\wedge$ Greedy (Richard) $\Rightarrow$ Evil(Richard)
King (John)
Greedy (John)
Brother (Richard, John)

Note: Remove universally quantified sentences after universal instantiation.

Existential instantiation (EI)

For any sentence α and variable v in that, introduce a constant that is not in the KB (called skolem constant) and substitute that constant for v.

E.g.: Consider the sentence, $\exists$ x Crown(x) $\wedge$ OnHead(x, John)

After EI,

Crown(C1) $\wedge$ OnHead(C1, John)   where C1 is Skolem Constant.

Resolution for FOPL:

- Based on resolution for propositional logic
- Extended syntax: allow variables and quantifiers
- Define "clausal form" for first-order logic formulae (CNF)
- Eliminate quantifiers from clausal forms
- Adapt resolution procedure to cope with variables (unification)

Conversion to CNF:

1. Eliminate implications and bi-implications as in propositional case
2. Move negations inward using De Morgan's laws

   plus rewriting $\neg \forall x P$ as $\exists x \neg P$ and $\neg \exists x P$ as $\forall x \neg P$
3. Eliminate double negations
4. Rename bound variables if necessary so each only occurs once

   e.g. $\forall x P(x) \vee \exists x Q(x)$ becomes $\forall x P(x) \vee \exists y Q(y)$
5. Use equivalences to move quantifiers to the left

   e.g. $\forall x P(x) \wedge Q$ becomes $\forall x\, (P(x) \wedge Q)$ where x is not in Q

   e.g. $\forall x P(x) \wedge \exists y Q(y)$ becomes $\forall x \exists y (P(x) \wedge Q(y))$
6. Skolemise (replace each existentially quantified variable by a **new** term)

   $\exists x P(x)$ becomes $P(a_0)$ using a Skolem constant a_0 since $\exists x$ occurs at the outermost level

   $\forall x \exists y P(x, y)$ becomes $P(x, f_0(x))$ using a Skolem function f_0 since $\exists y$ occurs within $\forall x$
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

For Example:

1.) $\neg [\exists x \forall y \forall z\, ((P(y) \vee Q(z)) \rightarrow (P(x) \vee Q(x)))]$

Solution:

1. $\neg \exists x \forall y \forall z\, (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$

2. $\forall x \neg \forall y \forall z\, (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$

2. $\forall x \exists y \neg \forall z\, (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$

2. $\forall x \exists y \exists z \neg (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$

2. $\forall x \exists y \exists z\, ((P(y) \vee Q(z)) \wedge \neg (P(x) \vee Q(x)))$

6. $\forall x\, ((P(f(x)) \vee Q(g(x))) \wedge \neg P(x) \wedge \neg Q(x))$

7. $(P(f(x)) \vee Q(g(x)) \wedge \neg P(x) \wedge \neg Q(x)$

8. $\{P(f(x)) \vee Q(g(x)),\ \neg P(x),\ \neg Q(x)\}$

Unification and Lifting:

A unifier of two atomic formulae is a substitution of terms **for variables** that makes them identical.

- Each variable has at most one associated term
- Substitutions are applied simultaneously

Unifier of $P(x, f(a), z)$ and $P(z, z, u)$: $\{x/ f(a), z/ f(a), u/ f(a)\}$

We can get the inference immediately if we can find a substitution α such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

α = {x/John,y/John} works

Unify(α ,β) = θ if αθ = θβ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,OJ)	{fail}

Last unification is failed due to overlap of variables. x can not take the values of John and OJ at the same time.

We can avoid this problem by renaming to avoid the name clashes (standardizing apart)

E.g.

Unify{Knows(John,x) Knows(z,OJ) } = {x/OJ, z/John}

Let C1 and C2 be two clauses. If C1 and C2 have no variables in common, then they are said to be **standardized apart**. **Standardized apart** eliminates overlap of variables to avoid clashes by renaming variables.

Another complication:

To unify *Knows(John,x)* and *Knows(y,z)*,

Unification of Knows*(John, x)* and *Knows(y, z)* gives α ={y/John, x/z } or α={y/John, x/John, z/John}

First unifier gives the result Knows(John, z) and second unifier gives the result Knows(John, John). Second can be achieved from first by substituting john in place of z. The first unifier is more general than the second.

There is a single most general unifier (MGU) that is unique up to renaming of variables.

MGU = {y/John, x/z }

Resolution for First-Order Logic

- Based on resolution for propositional logic
- Extended syntax: allow variables and quantifiers
- Define “clausal form” for first-order logic formulae
- Eliminate quantifiers from clausal forms
- Adapt resolution procedure to cope with variables (unification)

First-Order Resolution

For clauses $P \vee Q$ and $\neg Q' \vee R$ with Q, Q' atomic formulae

$$\begin{tikzcd}
P \vee Q \arrow[dr, dash] & & \neg Q' \vee R \arrow[dl, dash] \\
& (P \vee R)\theta &
\end{tikzcd}$$

Where, θ is a most general unifier for Q and Q'

$(P \vee R)\theta$ is the resolvent of the two clauses

Applying Resolution Refutation

- Negate query to be proven (resolution is a refutation system)
- Convert knowledge base and negated query into CNF and extract clauses
- Repeatedly apply resolution to clauses or copies of clauses until either the empty clause (contradiction) is derived or no more clauses can be derived (a copy of a clause is the clause with all variables renamed)
- If the empty clause is derived, answer ‘yes’ (query follows from knowledge base), otherwise answer ‘no’ (query does not follow from knowledge base)

Resolution: Examples

Example1:

“Tom is a dog”. “All dogs are animal”. “Animals will die”. Prove that “Tom will die”.

Constants: Tom
Predicates: dog(x), animal(x), die(x)

The predicate forms are
1.$\forall$(x) (dog(x) $\rightarrow$ animal(x))

2. dog (Tom)
3. ∀(y) (animal (y) → die(y))

The Knowledge Base in CNF Form is:

1.¬ dog(x) v animal(x)
2. dog (Tom)
3. ¬ animal (y) v die(y)
4. ¬die (Tom) [Negation of Conclusion]

Now, Using Resolution in 1 and 2 by lifting x/Tom;

5. animal(Tom)

Using Resolution in 3 and 5 by lifting y/Tom;

6. die(Tom)

Using resolution in 4 and 6, it results empty clause. Hence “Tom will die” can be inferred.

Example 2:

1. Marcus is a person.
2. Marcus is a Pompeian.
3. All Pompeians are Roman.
4. Caesar is a ruler.
5. All Romans are either loyal to Caesar or hate Caesar.
6. Everyone is loyal to someone.
7. People only try to assassinate rulers to whom they are not loyal.
8. Marcus tried to assassinate Caesar.

Constants: Marcus, Caesar

Predicates: People(x), Pompeian(x), Roman(x), Ruler(x), Loyal(x,y), Hate(x,y), Assassinate(x,y)

1. $\text{People}(\text{Marcus})$
2. $\text{Pompeian}(\text{Marcus})$
3. $\forall x \text{Pompeian}(x) \rightarrow \text{Roman}(x)$
4. $\text{Ruler}(\text{Caesar})$
5. $\forall x \text{Roman}(x) \rightarrow [\text{Loyal}(x, \text{Caesar}) \vee \text{Hate}(x, \text{Caesar})]$
6. $\forall x \exists y \text{Loyal}(x, y)$
7. $\forall x \forall y (\text{People}(x) \wedge \text{Ruler}(y) \wedge \text{Assassinate}(x, y)) \rightarrow \neg\text{Loyal}(x, y)$
8. $\text{Assassinate}(\text{Marcus}, \text{Caesar})$

Negation of Query $\neg\text{Hate}(\text{Marcus}, \text{Caesar})$

Knowledge Base in CNF

1. $\text{People}(\text{Marcus})$
2. $\text{Pompeian}(\text{Marcus})$
3. $\neg\text{Pompeian}(x_1) \vee \text{Roman}(x_1)$
4. $\text{Ruler}(\text{Caesar})$
5. $\neg\text{Roman}(x_2) \vee \text{Loyal}(x_2, \text{Caesar}) \vee \text{Hate}(x_2, \text{Caesar})$
6. $\text{Loyal}(x_3, F(x_3))$
7. $\neg\text{People}(x_4) \vee \neg\text{Ruler}(x_5) \vee \neg\text{Assassinate}(x_4, x_5) \vee \neg\text{Loyal}(x_4, x_5)$
8. $\text{Assassinate}(\text{Marcus}, \text{Caesar})$

Negation of Query $\neg\text{Hate}(\text{Marcus}, \text{Caesar})$

Resolution Proof

Sentence 1	Sentence 2	SUBST	Result
Query	5.	$\theta = \{x_2/\text{Marcus}\}$	9. $\neg\text{Roman}(\text{Marcus}) \lor \text{Loyal}(\text{Marcus}, \text{Caesar})$
9.	3.	$\theta = \{x_1/\text{Marcus}\}$	10. $\text{Loyal}(\text{Marcus}, \text{Caesar}) \lor \neg\text{Pompeian}(\text{Marcus})$
10.	2		11. $\text{Loyal}(\text{Marcus}, \text{Caesar})$
11.	7.	$\theta = \{x_4/\text{Marcus}, x_5.\text{Caesar}\}$	12. $\neg\text{People}(\text{Marcus}) \lor \neg\text{Ruler}(\text{Caesar}) \lor \neg\text{Assassinate}(\text{Marcus}, \text{Caesar})$
12.	1.		13. $\neg\text{Ruler}(\text{Caesar}) \lor \neg\text{Assassinate}(\text{Marcus}, \text{Caesar})$
13.	8.		14. $\neg\text{Ruler}(\text{Caesar})$
14.	4.		

Contradiction found. Therefore, we can conclude that Marcus hates Caesar.

Example 3:

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 John is lucky. Anyone who is lucky wins the lottery. Is John happy?

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

   $\forall x\ \text{Pass}(x, \text{History}) \land \text{Win}(x, \text{Lottery}) \Rightarrow \text{Happy}(x)$

2. But anyone who studies or is lucky can pass all his exams.

   $\forall x\ \forall y\ \text{Study}(x) \lor \text{Lucky}(x) \Rightarrow \text{Pass}(x,y)$

3. John did not study, but John is lucky

   $\neg\ \text{Study}(\text{John}) \land \text{Lucky}(\text{John})$

4. Anyone who is lucky wins the lottery.

   $\forall x\ \text{Lucky}(x) \Rightarrow \text{Win}(x, \text{Lottery})$

Now, Convert the KB to CNF:

Eliminate implications:

1. $\forall x \neg (\text{Pass}(x, \text{History}) \land \text{Win}(x, \text{Lottery})) \lor \text{Happy}(x)$

2. $\forall x\ \forall y \neg (\text{Study}(x) \lor \text{Lucky}(x)\ ) \lor \text{Pass}(x,y)$

3. $\neg\ \text{Study}(\text{John}) \land \text{Lucky}(\text{John})$

4. $\forall x \neg \text{Lucky}(x) \lor \text{Win}(x, \text{Lottery})$

Move $\neg$ inward

1. $\forall x \neg \text{Pass}(x, \text{History}) \lor \neg \text{Win}(x, \text{Lottery})) \lor \text{Happy}(x)$
2. $\forall x\ \forall y\ (\neg\ \text{Study}(x) \land \neg\text{Lucky}(x)\ ) \lor \text{Pass}(x,y)$
3. $\neg\ \text{Study}(\text{John}) \land \text{Lucky}(\text{John})$
4. $\forall x \neg \text{Lucky}(x) \lor \text{Win}(x, \text{Lottery})$

Distribute ∧ over ∨

1. ¬ Pass(x, History) ∨ ¬ Win(x, Lottery)) ∨ Happy(x)

2. (¬ Study(x) ∨ Pass(x,y)) ∧ (¬ Lucky(x) ∨ Pass(x,y))

3. ¬ Study(John) ∧ Lucky(John)

4. ¬ Lucky(x) ∨ Win(x, Lottery)

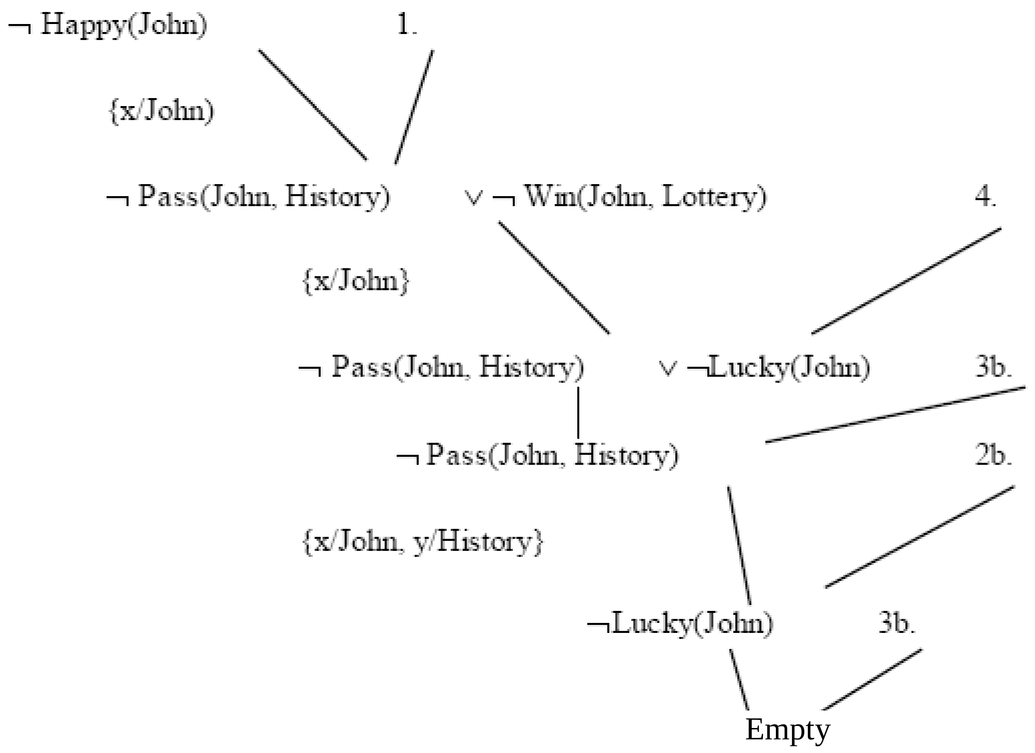
Now, the KB contains:

1. ¬ Pass(x, History) ∨ ¬ Win(x, Lottery) ∨ Happy(x)
2. a. ¬ Study(x) ∨ Pass(x,y)
2. b. ¬Lucky(x) ∨ Pass(x,y)
3. a. ¬ Study(John)
   b. Lucky(John)
4. ¬ Lucky(x) ∨ Win(x, Lottery)

Standardize the variables apart:

1. ¬ Pass(x1, History) ∨ ¬ Win(x1, Lottery) ∨ Happy(x1)
2. a. ¬ Study(x2) ∨ Pass(x2,y1)
2. b. ¬Lucky(x3) ∨ Pass(x3,y2)
3. a. ¬ Study(John)
   b. Lucky(John)
4. ¬ Lucky(x4) ∨ Win(x4, Lottery)
   ¬ Happy(John) **5.** **(Negation of the conclusion added)**

<u>Now Use resolution as below:</u>


¬ Happy(John)
1.
{x/John)
¬ Pass(John, History) ∨ ¬ Win(John, Lottery)
4.
{x/John}
¬ Pass(John, History) ∨ ¬Lucky(John)
3b.
¬ Pass(John, History)
2b.
{x/John, y/History}
¬Lucky(John)
3b.
Empty

Symbolic versus statistical reasoning:

The (Symbolic) methods basically represent statements as being

- True,
- False, *or*
- Neither True nor False.

Statistical methods provide a method for representing beliefs and statements that are not certain (or uncertain) but for which there may be some supporting (or contradictory) evidence.

Uncertain Knowledge:

Let action A_t = leave for airport t minutes before flight. Will A_t get me there on time?

Problems:

1. Partial observability (road state, other drivers' plans, etc.)
2. Noisy sensors (radio traffic reports)
3. Uncertainty in action outcomes (flat tyre, etc.)
4. Complexity of modeling and predicting traffic

Hence a purely logical approach either

1. Risks falsehood: "A_{25} will get me there on time" or
2. Leads to conclusions that are too weak for decision making: "A_{25} will get me there on time if there's no accident on the bridge and it doesn't rain and my tires remain intact etc."

A_{1440} might reasonably be said to get me there on time but I'd have to stay overnight in the airport… **(Refer Russel and Norvig for Detail)**

Handling Uncertainty:

Instead of providing all condition it can express with degree of beliefs in the relevant sentences.

Example:

Say we have a rule

if toothache then problem is cavity

But not all patients have toothaches because of cavities (although perhaps most do)

So we could set up rules like

if toothache and not(gum disease) and not(filling) and ......then problem is cavity

This gets very complicated! a better method would be to say

if toothache then problem is cavity with probability 0.8

Given the available evidence,

A_{25} will get me there on time with probability 0.04

A most important tool for dealing with degree of beliefs is probability theory, which assigns to each sentence a numerical degree of belief between 0 & 1.

Making decisions under uncertainty:

Suppose I believe the following:

$P(A_{25}$ *gets me there on time*$|\ldots) = 0.04$

$P(A_{90}$ *gets me there on time*$|\ldots) = 0.70$

$P(A_{120}$ *gets me there on time*$|\ldots) = 0.95$

$P(A_{1440}$ *gets me there on time*$|\ldots) = 0.9999$

Which action to choose?

- Depends on my preferences for missing flight vs. length of wait at airport, etc.
  Utility theory is used to represent and infer preferences
  Decision theory = utility theory + probability theory

The rational decision depends on both the relative importance of various goals and the likelihood that, and degree to which, they will be achieved.

Basic Statistical methods – Probability:

The basic approach statistical methods adopt to deal with uncertainty is via the axioms of probability:

- Probabilities are (real) numbers in the range 0 to 1.
- A probability of $P(A) = 0$ indicates total uncertainty in A, $P(A) = 1$ total certainty and values in between some degree of (un)certainty.
- Probabilities can be calculated in a number of ways.

Very Simply

Probability = (number of desired outcomes) / (total number of outcomes)

So given a pack of playing cards the probability of being dealt an ace from a full normal deck is 4 (the number of aces) / 52 (number of cards in deck) which is 1/13. Similarly the probability of being dealt a spade suit is 13 / 52 = 1/4.

Conditional probability, *P*(*A*|*B*), indicates the probability of of event *A* given that we know event *B* has occurred.

The aim of a **probabilistic logic** (or **probability logic**) is to combine the capacity of probability theory to handle uncertainty with the capacity of deductive logic to exploit structure. The result is a richer and more expressive formalism with a broad range of possible application areas. **Probabilistic logic is a natural extension of traditional logic truth tables: the results they define are derived through probabilistic expressions instead.** The difficulty with probabilistic logics is that they tend to multiply the computational complexities of their probabilistic and logical components.

Random Variables:

In probability theory and statistics, a **random variable** (or **stochastic variable**) is a way of assigning a value (often a real number) to each possible outcome of a random event. These values might represent the possible outcomes of an experiment, or the potential values of a quantity whose value is uncertain (e.g., as a result of incomplete information or imprecise measurements.) Intuitively, a random variable can be thought of as a quantity whose value is not fixed, but which can take on different values; normally, a probability distribution is used to describe the probability of different values occurring. Random variables are usually real-valued, but one can consider arbitrary types such as boolean values, complex numbers, vectors, matrices, sequences, trees, sets, shapes, manifolds and functions. The term *random element* is used to encompass all such related concepts.

For example: There are two possible outcomes for a coin toss: heads, or tails. The possible outcomes for one fair coin toss can be described using the following random variable:

$$X = \begin{cases} \text{head}, \\ \text{tail}. \end{cases}$$

and if the coin is equally likely to land on either side then it has a probability mass function given by:

$$\rho_X(x) = \begin{cases} \frac{1}{2}, & \text{if } x = \text{head}, \\ \frac{1}{2}, & \text{if } x = \text{tail}. \end{cases}$$

Example: A simple world consisting of two random variables:
Cavity– a Boolean variable that refers to whether my lower left wisdom tooth has a cavity

Toothache- a Boolean variable that refers to whether I have a toothache or not

We use the single capital letters to represent unknown random variables

P induces a probability distribution for any random variables X.

Each RV has a domain of values that it can take it, e. g. domain of *Cavity* is {true, false}

Random Variable domain are: Boolean, Discrete and Continuous

Discrete random variables (finite or infinite)
e.g., *Weather* is one of *(sunny, rain, cloudy, snow)*
Weather = *rain* is a proposition

Continuous random variables (bounded or unbounded)
e.g., *Temp* = *21.6*, also allow, e.g., *Temp* < *22.0*.

Atomic Event:

An **atomic event** is a complete specification of the state of the world about which the agent is uncertain.

Example:
In the above world with two random variables (Cavity and Toothache) there are only four distinct atomic events, one being:
Cavity = false, Toothache = true
Which are the other three atomic events?

Propositions:

Think of a proposition as the event (set of sample points) where the proposition is true

Given Boolean random variables A and B:

event α = set of sample points where $A(\omega)$ = true
event $\neg\alpha$ = set of sample points where $A(\omega)$ = false
event $a \wedge b$ = points where $A(\omega)$ = true and $B(\omega)$ = true

Often in AI applications, the sample points are defined by the values of a set of random variables, i.e., the sample space is the Cartesian product of the ranges of the variables.

With Boolean variables, sample point = propositional logic model
e.g., A = true, B = false, or $a \wedge \neg b$.

Proposition = disjunction of atomic events in which it is true
e.g., $(a \vee b) \equiv (\neg a \wedge b) \vee (a \wedge \neg b) \vee (a \wedge b)$

$P(a \vee b) = P(\neg a \wedge b) + P(a \wedge \neg b) + P(a \wedge b)$

Propositional or Boolean random variables

e.g., *Cavity*(do I have a cavity?)

Prior Probability:

The prior or unconditional probability associated with a proposition is the degree of belief accorded to it in the absence of any other information.

Example:
P(Weather= sunny) = 0.72, P(Weather= rain) = 0.1, P(Weather= cloudy) = 0.08, P(Weather= snow) = 0.1

Probability distribution gives values for all possible assignments:

P(Weather) = (0.72, 0.1, 0.08, 0.1)

Joint probability distribution, for a set of random variables, gives the probability of every atomic event on those random variables.

P(Weather, Cavity) = a 4 ×2 matrix of values.

Weather=	**sunny**	**rain**	**cloudy**	**snow**
Cavity=true	0.144	0.02	0.016	0,02
Cavity=false	0.576	0.08	0.064	0.08

Every question about a domain can be answered by the joint distribution because every event is a sum of sample points.

Conditional Probability:

The conditional probability "P(a|b)" is the probability of "a" given that all we know is "b".

Example: *P(cavity|toothache) = 0.8* means if a patient have toothache and no other information is yet available, then the probability of patient's having the cavity is 0.8.

Definition of conditional probability:

$P(a|b) = P(a \wedge b)/P(b)$ if $P(b) \neq 0$

Product rule gives an alternative formulation:

$P(a \wedge b) = P(a|b)P(b) = p(b|a)P(a)$

Inference using full joint probability distribution:

We use the full joint distribution as the knowledge base from which answers to all questions may be derived. The probability of a proposition is equal to the sum of the probabilities of the atomic events in which it holds.

$P(a) = \Sigma P(ei)$

Therefore, given a full joint distribution that specifies the probabilities of all the atomic events, one can compute the probability of any proposition.

Full Joint probability distribution: an example

We consider the following domain consisting of three Boolean variables: Toothache, Cavity, and Catch (the dentist's nasty steel probe catches in my tooth).

The full joint distribution is the following 2x2x2 table:

	toothache		¬toothache	
	Catch	¬catch	catch	¬catch
Cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

The probability of any proposition can be computed from the probabilities in the table. The probabilities in the joint distribution must sum to 1.

Each cell represents an atomic event and these are all the possible atomic events.

P(cavity or toothache) = P(cavity, toothache, catch) + P(cavity, toothache, ¬catch) + P(cavity, ¬toothache, catch) + P(cavity, ¬toothache, ¬catch) + P(¬cavity, toothache, catch) + P(¬cavity, toothache, ¬catch)
= 0.108+0.012+0.072+0.008+0.016+0.064=0.28

We simply identify those atomic events in which the proposition is true and add up their probabilities

<u>Marginalization or summing out:</u>

Distribution over **Y** can be obtained by summing out all the other variables from any joint distribution containing **Y**. This process is called marginalization.

$$\mathbf{P}(\mathbf{Y}) = \sum P(\mathbf{Y}, z)$$

Examples:

P(cavity) = 0.108 +0.012 +0.072 + 0.008 = 0.2

P(¬Toothache) = 0.072 + 0.008 + 0.144 + 0.576 = 0.8

P(Cavity, ¬Toothache) = 0.072 + 0.008 = 0.08

Calculating Conditional Probability:

P(¬cavity | Toothache) = P(¬cavity ^ Toothache)/ P(Toothache)
= (0.016 + 0.064)/(0.108 + 0.012 + 0.016 + 0.064)
= 0.4

Again let's calculate

P(cavity | Toothache) = P(cavity ^ Toothache)/ P(Toothache)
= (0.108 + 0.012)/(0.108 + 0.012 + 0.016 + 0.064)
= 0.6

Notice that in above two calculations the term 1/ P(Toothache) remain constant no matter which value of cavity is calculated. This constant term is called **normalization constant** for the distribution P(cavity | Toothache), ensuring that it adds up to 1.

Independence:

A and B are independent iff

P(A|B) = P(A) or P(B|A) = P(B) or P(A, B) = P(A)P(B)

Example:

P(Toothache,Catch,Cavity,Weather) = P(Toothache,Catch,Cavity)P(Weather)

Here weather is independent of other three variables.

Bayes' Rule (Theorem) :

$$\boxed{P(b|a) = \frac{P(a|b) * P(b)}{P(a)}}$$

Proof of bays rule:

We know that:

P(a|b) = P(a ^ b)/ P(b)

P(a ^ b) = P(a|b) P(b)……………….(1)

Similarly

P(b|a) = P(a ^ b)/ P(a)

P(a ^ b) = P(b|a) P(a) ……………….(2)

Equating 1 and 2

$P(a|b)\ P(b) = P(b|a)\ P(a)$

i.e. $P(b|a) = P(a|b)\ P(b)/P(a)$

Why is the Bayes' rule is useful in practice?

Bayes' rule is useful in practice because there are many cases where we have good probability estimates for three of the four probabilities involved, and therefore can compute the fourth one.

Useful for assessing diagnostic probability from causal probability:

$$P(Cause|Effect) = \frac{P(Effect|Cause)P(Cause)}{P(Effect)}$$

Diagnostic knowledge is often more fragile than causal knowledge.

Example of Bayes' rule:

A doctor knows that the disease meningitis causes the patient to have a stiff neck 70% of the time. The doctor also knows that the probability that a patient has meningitis is 1/50,000, and the probability that any patient has a stiff neck is 1%.

Find the probability that a patient with a stiff neck has meningitis.

Here, we are given;

$p(s|m) = 0.7$

$p(m) = 1/50000$

$p(s) = 0.01$

Now using Bayes' rule;

$P(m|s) = P(s|m)P(m)/P(s) = (0.7*1/50000)/(0.01) = 0.0014$

Uses of Bayes' Theorem :

In doing an expert task, such as medical diagnosis, the goal is to determine identifications (diseases) given observations (symptoms). Bayes' Theorem provides such a relationship.

*P(A | B) = P(B | A) * P(A) / P(B)*

Suppose: *A* = Patient has measles, *B* = has a rash

Then: *P(measles/rash) =* *P(rash/measles) * P(measles) / P(rash)*

The desired diagnostic relationship on the left can be calculated based on the known statistical quantities on the right.

Bayesian networks:

- *A data structure to represent the dependencies among variables and to give a concise specification of any full joint probability distribution*.
- Also called *belief networks* or *probabilistic network* or *casual network* or *knowledge map*.

The basic idea is:

- Knowledge in the world is *modular* -- most events are conditionally independent of most other events.
- Adopt a model that can use a more local representation to allow interactions between events that *only* affect each other.
- Some events may only be *unidirectional* others may be *bidirectional* -- make a distinction between these in model.
- Events may be causal and thus get chained together in a network.

A Bayesian network is a directed acyclic graph which consists of:

- A set of random variables which makes up the nodes of the network.
- A set of directed links (arrows) connecting pairs of nodes. If there is an arrow from node X to node Y, X is said to be a parent of Y.
- Each node Xi has a conditional probability distribution P(Xi| Parents(Xi)) that quantifies the effect of the parents on the node.

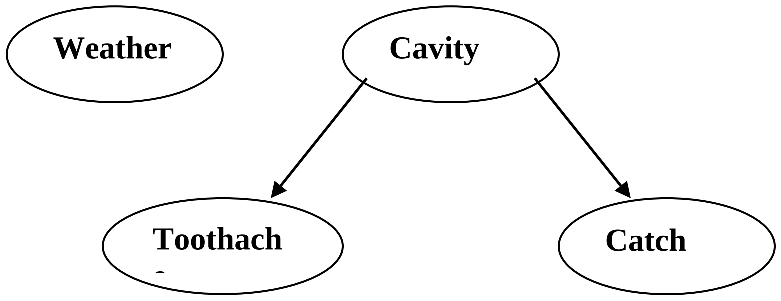
Intuitions:

- A Bayesian network models our incomplete understanding of the causal relationships from an application domain.
- A node represents some state of affairs or event.
- **A link from X to Y means that X has a direct influence on Y.**

Implementation:

- A *Bayesian Network* is a *directed acyclic graph*:
  - A graph where the directions are links which indicate dependencies that exist between nodes.
  - Nodes represent propositions about events or events themselves.
  - Conditional probabilities quantify the strength of dependencies.

Our existing simple world of variables *toothache, cavity, catch* & *weather* is represented as:



Weather is independent of the other variables

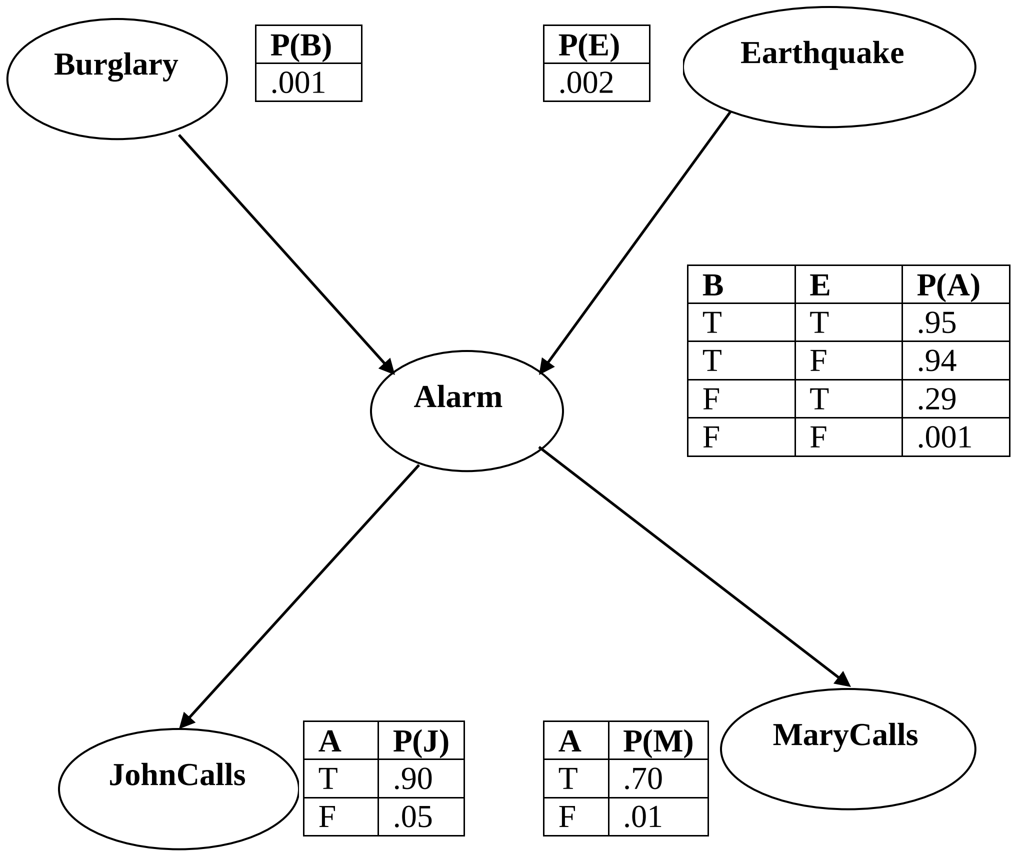
Example:

Sample Domain:

You have a burglar alarm installed in your home. It is fairly reliable at detecting a burglary, but also responds on occasion to minor earthquakes. You also have two neighbors, John and Mary, who have promised to call you at work when they hear the alarm. John always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too. Mary, on the other hand, likes rather loud music and sometimes misses the alarm altogether.

We would like have to estimate the probability of a burglary with given evidence who has or has not call.

Variables: Burglary, Earthquake, Alarm, JohnCalls, MaryCalls



P(B)
.001

P(E)
.002

B	E	P(A)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J)
T	.90
F	.05

A	P(M)
T	.70
F	.01

The probabilities associated with the nodes reflect our representation of the causal relationships.

A Bayesian network provides a complete description of the domain in the sense that one can compute the probability of any state of the world (represented as a particular assignment to each variable).

Example: What is the probability that the alarm has sounded, but neitherburglary nor an earthquake has occurred, and both John and Mary call?

$P(j, m, a, \neg b, \neg e) = P(j|a)\ P(m|a)\ P(a|, \neg b, \neg e)\ P(\neg b)\ P(\neg e)$

$= 0.90*0.70*0.001*0.999*0.998 = 0.00062$

Fuzzy Logic and Fuzzy Rule Base System

Fuzzy Logic:

Fuzzy logic is a form of many-valued logic**; it deals with** reasoning **that is approximate rather than fixed and exact.** Compared to traditional binary sets (where variables may take on true or false values) fuzzy logic variables may have a truth value that ranges in degree between 0 and 1.

Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. Furthermore, when linguistic variables are used, these degrees may be managed by specific functions.

Fuzzy logic is based on the **observation that people make decisions based on imprecise and non-numerical information**. Fuzzy models or sets are mathematical means of representing **vagueness and imprecise information** (hence the term fuzzy). These models have the capability of recognizing, representing, manipulating, interpreting, and utilizing data and information that are vague and lack certainty.

More specifically, fuzzy logic may be viewed as an attempt at formalization/mechanization of two remarkable human capabilities. First, the capability to converse, reason and make rational decisions in an environment of imprecision, uncertainty, incompleteness of information, conflicting information, partiality of truth and partiality of possibility – in short, in an environment of **imperfect information**. And second, the capability to perform a wide variety of physical and mental tasks **without any measurements and any computations**.

Linguistic Variables:

By a ***linguistic variable we*** mean a variable whose values are words or sentences in a natural or artificial language. A numerical variable takes numerical values: Age = 65. While, a linguistic variables takes linguistic values: Age is old

A linguistic values is a fuzzy set. It has a LABEL and a MEANING. The LABEL is a Symbol, Sentence in a Language while MEANING is the Fuzzy Subset of a Universe of Discourse

All linguistic values form a term set:

T(age) = {young, not young, very young, ...middle aged, not middle aged, ...old, not old, very old, more or less old, ...not very young and not very old, ...}

Knowledge Representation in Fuzzy Logic:

Knowledge in fuzzy in represented using the fuzzy logic propositions.

A fuzzy logic proposition, P is a statement involving some concept without clearly defined boundaries. Linguistic statements that tend to express subjective ideas and that can be interpreted slightly differently by various individuals typically involve fuzzy propositions. Most natural language is fuzzy, in that it involves vague and imprecise terms. Statements describing a person's height or weight or assessments of people's preferences about colors or menus can be used as examples of fuzzy propositions. The truth value assigned to P∼ can be any value on the interval [0, 1]. The assignment of the truth value to a proposition is actually a mapping from the interval [0, 1] to the universe U of truth values, *T*.

$$T : u \in \mathrm{U} \longrightarrow (0, 1).$$

As in classical binary logic, we assign a logical proposition to a set in the universe of discourse. Fuzzy propositions are assigned to fuzzy sets. Suppose proposition P is assigned to fuzzy set A; then, the truth value of a proposition, denoted *T* (P), is given by

$$T(\mathrm{P}) = \mu_{\mathrm{A}}(x), \text{ where } 0 \le \mu_{\mathrm{A}} \le 1$$

It indicates that the degree of truth for the proposition P: $x \in$A is equal to the membership grade of x in the fuzzy set A.

The logical connectives of negation, disjunction, conjunction, and implication are also defined for a fuzzy logic. For two simple propositions: proposition P∼ defined on fuzzy set A and proposition Q defined on fuzzy set B;

.

Negation

$$T(\overline{\underset{\sim}{\mathrm{P}}}) = 1 - T(\underset{\sim}{\mathrm{P}}).$$

Disjunction

$$\underset{\sim}{\mathrm{P}} \vee \underset{\sim}{\mathrm{Q}} : x \text{ is } \underset{\sim}{\mathrm{A}} \text{ or } \underset{\sim}{\mathrm{B}} \quad T(\underset{\sim}{\mathrm{P}} \vee \underset{\sim}{\mathrm{Q}}) = \max(T(\underset{\sim}{\mathrm{P}}), T(\underset{\sim}{\mathrm{Q}})).$$

Conjunction

$$\underset{\sim}{\mathrm{P}} \wedge \underset{\sim}{\mathrm{Q}} : x \text{ is } \underset{\sim}{\mathrm{A}} \text{ and } \underset{\sim}{\mathrm{B}} \quad T(\underset{\sim}{\mathrm{P}} \wedge \underset{\sim}{\mathrm{Q}}) = \min(T(\underset{\sim}{\mathrm{P}}), T(\underset{\sim}{\mathrm{Q}})).$$

Implication (Zadeh, 1973)

$$\underset{\sim}{\mathrm{P}} \to \underset{\sim}{\mathrm{Q}} : x \text{ is } \underset{\sim}{\mathrm{A}}, \text{ then } x \text{ is } \underset{\sim}{\mathrm{B}}$$

$$T(\underset{\sim}{\mathrm{P}} \to \underset{\sim}{\mathrm{Q}}) = T(\overline{\underset{\sim}{\mathrm{P}}} \vee \underset{\sim}{\mathrm{Q}}) = \max(T(\overline{\underset{\sim}{\mathrm{P}}}), T(\underset{\sim}{\mathrm{Q}})).$$

Fuzzy Rule Based System

In FRBS, it consists of fuzzy rules.

IF premise (antecedent), THEN conclusion (consequent).

The premises are fuzzy elements and conclusion are also fuzzy.

Rule 1:	IF condition C^1, THEN restriction R^1
Rule 2:	IF condition C^2, THEN restriction R^2
⋮	
Rule r:	IF condition C^r, THEN restriction R^r

The fuzzy rule-based system is most useful in modeling some complex systems that can be observed by humans because they make use of linguistic variables as their antecedents and consequents; as described here these linguistic variables can be naturally represented by fuzzy sets and logical connectives of these sets.

In a broad sense, fuzzy rule-based systems are rule-based systems, where fuzzy sets and fuzzy logic are used as tools for representing different forms of knowledge about the problem at hand, as well as for modeling the interactions and relationships existing between its variables.

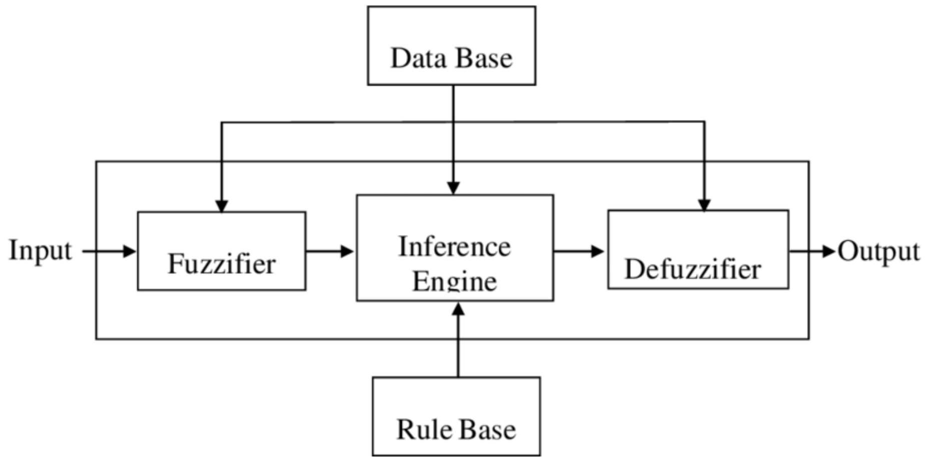


Fig: Block diagram of fuzzy rule based system

The fuzzifier converts real numbers of inputs into fuzzy sets. Fuzzy sets are functions that map a value that might be a member of the set to a number between zero and one indicating its actual degree of membership.

The knowledge base includes a fuzzy rule-base and a database. Membership functions of the linguistic terms are contained in the database. The rule base consists of if then rules, which represents the relationship between input and output variables.

The inference engine performs inference of the fuzzy rules using fuzzy inference techniques. **The defuzzifier converts fuzzy value back to real world crisp values.**

For example;

IF *temperature is hot*
THEN *fan speed is fast*

IF *temperature is hot* ***AND*** *humidity is high*
THEN *fan speed is fast*

IF *temperature is hot* ***OR*** *humidity is high*
THEN *fan speed is fast*

IF *temperature is* ***NOT*** *hot*
THEN *fan speed is slow*

Here the linguistic variables hot, fast, high, slow are described using fuzzy sets. The crisp input will be mapped to fuzzy sets of premises i. e. temperature and the fuzzy consequences will be mapped to fuzzy sets of speed. Then using fuzzy inference techniques like Mamdani Principle, inference of the rules will be done. The fuzzy outputs will be defuzzified to some crisp output using defuzzifucation methods.