Knowledge Representation

Our information age is composed of computer systems that can process and store vast amounts of information. Information comprises data and facts. There is a hierarchical relationship between data, facts, information, and knowledge. The simplest pieces of information are data; from data we can build facts, and from facts we gain information. Knowledge can be defined as the processing of information to enable intelligent decision-making.

Knowledge:

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.

Knowledge is awareness or familiarity gained by experiences of facts, data, and situations.

- Knowledge is understanding of a subject area.

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.

Difference between data, information, and knowledge:

Data: Primitive verifiable facts. Example: name of novels available in a library.

Information: Analyzed data. Example: The novel that is frequently asked by the members of library is “Harry Potter and the Chamber of Secrets”.

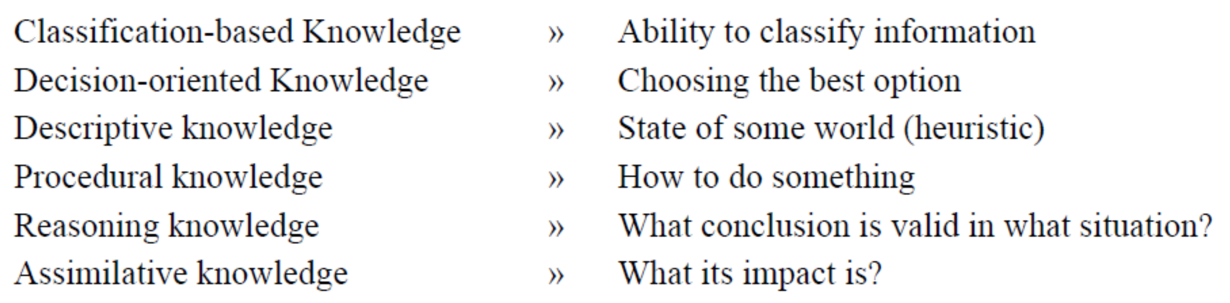
Knowledge: Analyzed information that is often used for further information deduction. Example: Since the librarian knows the name of the novel that is frequently asked by members, s/he will ask for more copies of the novel the next time s/he places an order.

Types of Knowledge

- **Procedural Knowledge**
  - Includes rules, strategies, agendas, procedures.
  - Also known as imperative knowledge
  - Is knowing How to do something
  - Can be directly applied to a task
  - Depends upon the task on which it can be applied
  - Less general
  - Example: How to cook vegetable or how to prepare a particular dish is procedural knowledge.
- **Declarative Knowledge**
  - Includes concepts, objects,facts.
  - Also known as descriptive knowledge
  - Is knowing about something
  - Is expressed in declarative sentences
  - Consists of facts
  - More general than procedural knowledge
  - Example: The first step in cooking a vegetable is chopping it.
  - Example 2: To prepare a dish one needs to gather its ingredients.

- **Structural Knowledge**
  - Includes rule sets, concept relationships, concept-to-object relationships.
  - Is basic to problem solving
  - Describes relationships between concepts like kind of, part of and groupings.
  - Example: Mango is a kind of fruit. Fruit is a kind of crop. Crop information is part of agricultural knowledge.
- **Inexact and Uncertain Knowledge**
  - Includes probabilities, uncertain facts – rules – relationships – evidence.
  - Characterizes situations in which information is imprecise, unavailable, incomplete, random or ambiguous
  - Example: Rumors about something or someone, terms like ‘little’, ‘too much’, ‘warm’, ‘more or less’ etc.
  - **Commonsense Knowledge**
    - Includes default propositions, approximate concepts and theories, general hierarchies and analogies.
    - Denotes vast amount of human knowledge about the world that cannot be stated in precise theories
    - Collection of facts and information that an ordinary person is expected to know
    - Example: The shape and color of an apple, knowledge about human emotions,reflexes etc

Knowledge can also be classified as



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

Knowledge representation is the way to express knowledge in a computer tractable form, so that it can be used to enable our AI agents to perform well.

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.

What to Represent?

Following kinds of knowledge might need to be represented in AI systems

Objects: Physical objects and concepts (e.g., table structure = height, width, depth).

Events: Time element and cause and effect relationships.

Performance: Information on how something is done (the steps) but also the logic or algorithm governing the performance.

Meta-knowledge: Knowledge about knowledge, reliability, and relative importance of facts. For example, if you cram the night before an exam, your knowledge about a subject isn’t likely to last too long

Why do we need Knowledge Representation?

- Unlike human mind, computers cannot acquire and represent knowledge by themselves.
- It is complicated to machine process a knowledge represented in natural language.
- Human knowledge is of different types.
- Knowledge manipulation involves:
    - Knowledge acquisition: gathering, structuring and organizing knowledge.
    - Knowledge storing: putting the knowledge into computer.
    - Knowledge retrieval: getting the knowledge when needed.
    - Reasoning: gives conclusion, inference or explanation

Issues in Knowledge Representation

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?

REPRESENTATION AND MAPPING

In order to solve the complex problems encountered in AI, one needs both large amount of knowledge and some mechanism for manipulating that knowledge to create solutions to new problems. Varieties of ways are there to represent knowledge. Following two points are important in knowledge representation

- Facts: truths in some relevant world. These are the things we want to represent.
- Representation of facts in some chosen formalism. These are the things we will actually be able to manipulate

One way to think of structuring these entities is as two levels:

1. The *knowledge level,* at which facts (including each agent's behaviors and current goals) are described.

2. The *symbol level* , at which the representation of objects at the knowledge level are defined in terms of symbols that can be manipulated by programs

Rather than thinking of one level above top of another, we will focus on facts, on representations, and on the two way mapping that must exist between them. These links are called *representation mapping.*

- ✓ The forward representation mapping maps from facts to representations.
- ✓ The backward representation mapping goes the other way, from representations to facts.

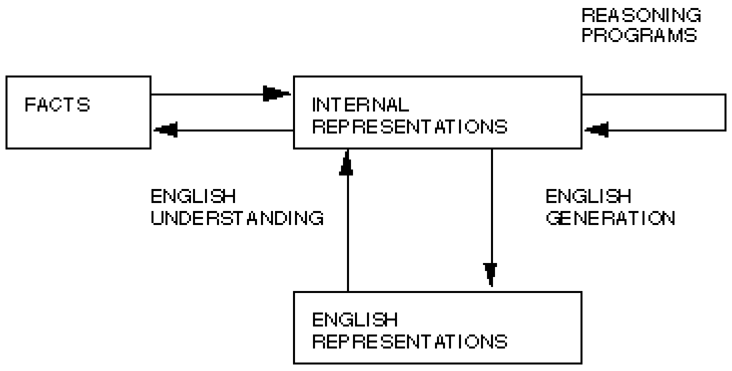


Figure: Mapping between facts and Representations

One representation of fact is so common that it deserves special mention: natural language (particularly English) sentence. Regardless of the representation for facts that we use in a program, we may also need to be concerned with an English representation of those facts in order to facilitate getting information into and out of the system. In this case, we must also have mapping function from English function to representation we are actually going to use and from it back to sentence.

Lets look at simple example using mathematical logic as the representational formalism. Consider the English Sentence:

Spot is a dog.

The fact represented by that English can also be represented in logic as:

dog(Spot)

Suppose that we also have logical representation of the fact that all dogs has tails:

$\forall x : dog(x) \rightarrow hastail(x)$

Then, using deductive mechanism of logic, we may generate new representation object:

hastail(Spot)

Using an appropriate backward mapping fnction, we could then generate the English sentence:

Spot has a tail.

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
- KR system should have the ability to represent all kind of required knowledge.

Inferential Adequacy

- The ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original
- KR system should have ability to manipulate the representational structures to produce new knowledge corresponding to existing structure.

Inferential Efficiency

- The ability to direct the inferential knowledge mechanism 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

Types of Knowledge Representation Systems

There are different types of knowledge representation techniques including followings:

- Semantic Nets
- Frames
- Conceptual Dependencies
- Scripts
- Rule Based Systems (Production System)
- Propositional Logic
- Predicate Logic

Rule Based System (Production System)

Rule-based systems are used as a way to store and manipulate knowledge to interpret information in a useful way. In this approach, idea is to use production rules, sometimes called IF-THEN rules. The syntax structure is

IF <premise> THEN <action>

<premise> - is Boolean. The AND, and to a lesser degree OR and NOT, logical connectives are possible.

<action> - a series of statements

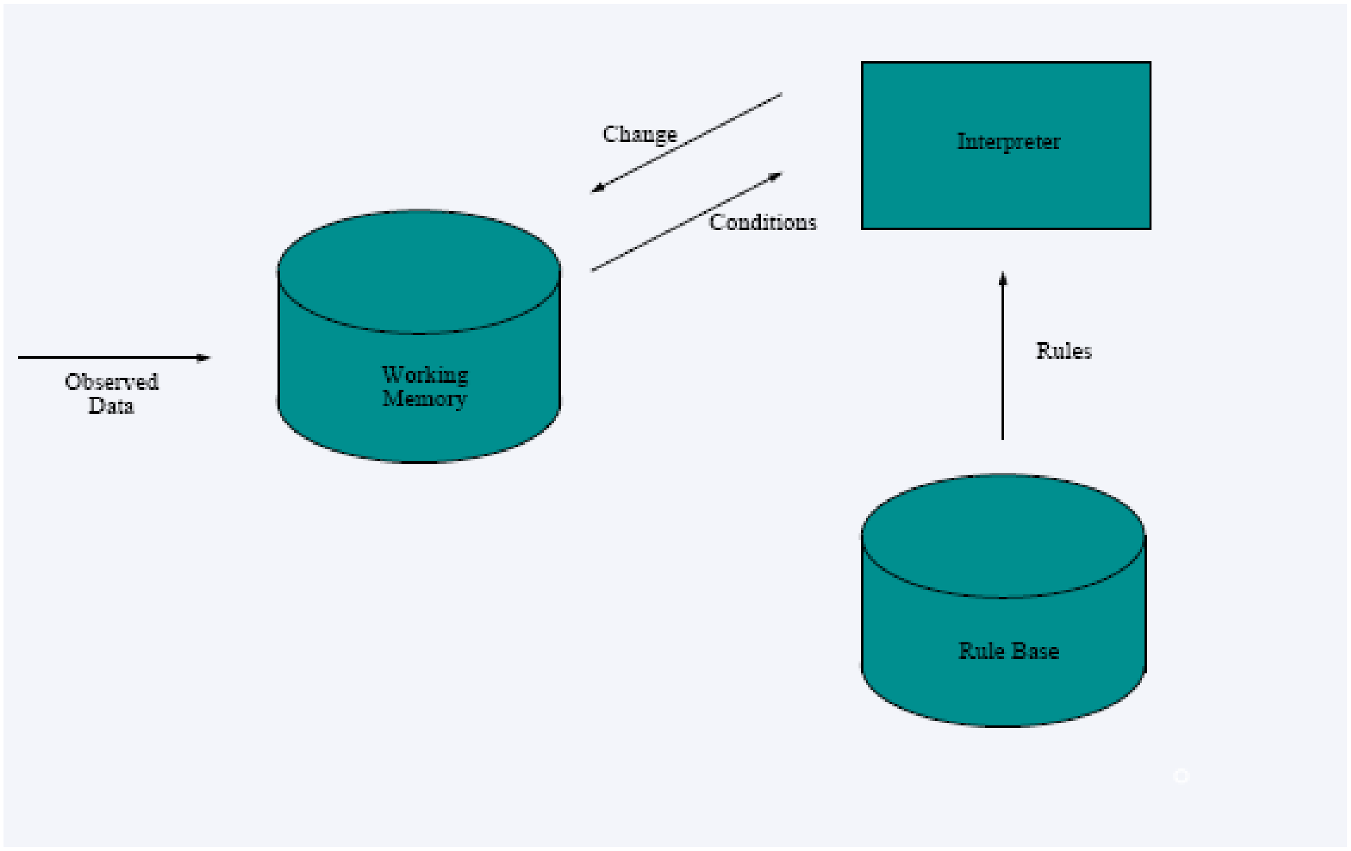
Notes:

• The rule premise can consist of a series of clauses and is sometimes referred to as the antecedent

• The actions are sometimes referred to as the consequent

A typical rule-based system has four basic components:

- A list of rules or **rule base,** which is a specific type of knowledge base.
- An **inference engine** or **semantic reasoner,** which infers information or takes action based on the interaction of input and the rule base.
- Temporary **working memory**
- A **user interface** or other connection to the outside world through which input and output signals are received and sent.



Working Memory contains facts about the world and can be observed directly or derived from a rule. It contains temporary knowledge – knowledge about this problem-solving session. It may be modified by the rules.

It is traditionally stored as <object, attribute, value> triplet.

Rule Base contains rules; each rule is a step in a problem solving process. Rules are persistent knowledge about the domain. The rules are typically only modified from the outside of the system, e.g. by an expert on the domain.

The syntax is a *IF <conditions> THEN <actions>* format.

The conditions are matched to the working memory, and if they are fulfilled, the rule may be fired.

Actions can be:

- ✓ Adding fact(s) to the working memory.

- ✓ Removing fact(s) from the working memory
- ✓ Modifying fact(s) in the working memory.

The **Interpreter** operates on a cycle:

- ✓ **Retrieval:** Finds the rules that match the current Working Memory. These rules are the Conflict Set.
- ✓ **Refinement:** Prunes, reorders and resolves conflicts in the Conflict Set.

- ✓ **Execution:** Executes the actions of the rules in the Conflict Set. Applies the rule by performing the action.

Advantages of rule based approach:

- *Naturalness of Expression:* Expert knowledge can often been seen naturally as rules of thumb.

- *Modularity:* Rules are independent of each other – new rules can be added or revised later. Interpreter is independent from rules.

- *Restricted Syntax:* Allows construction of rules and consistency checking by other programs. Allows (fairly easy) rephrasing to natural language.

Disadvantages (or limitations)

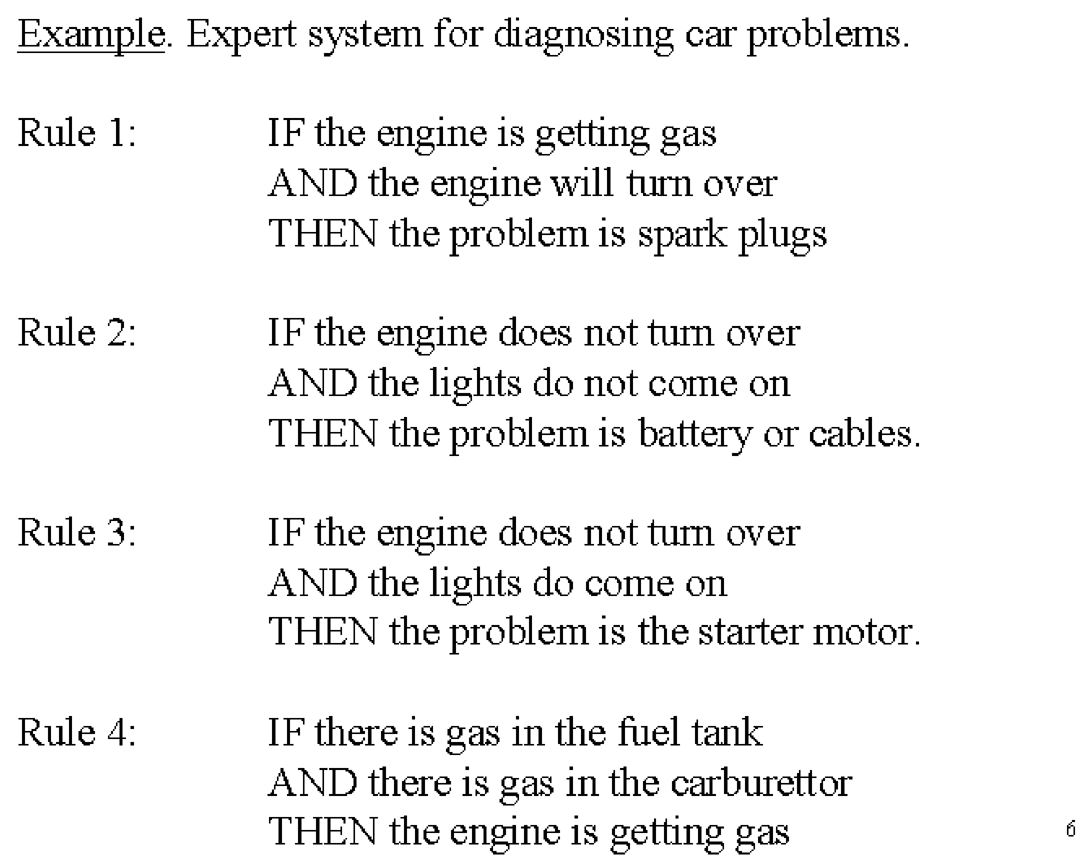
- rule bases can be very large (thousands of rules)

- rules may not reflect the actual decision making

- the only structure in the KB is through the rule chaining

Examples: —If the patient has stiff neck, high fever and an headache, check for Brain Meningitis‖. Then it can be represented in rule based approach as:

IF <FEVER, OVER, 39> AND <NECK, STIFF, YES> AND <HEAD, PAIN, YES> THEN add(<PATIENT,DIAGNOSE, MENINGITIS>)



Example. Expert system for diagnosing car problems.

Rule 1: IF the engine is getting gas
AND the engine will turn over
THEN the problem is spark plugs

Rule 2: IF the engine does not turn over
AND the lights do not come on
THEN the problem is battery or cables.

Rule 3: IF the engine does not turn over
AND the lights do come on
THEN the problem is the starter motor.

Rule 4: IF there is gas in the fuel tank
AND there is gas in the carburettor
THEN the engine is getting gas

6

Frames

- ✓ Object based approach
- ✓ 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.

<u>Disadvantages</u>

- complex

- reasoning (inferencing) is difficult

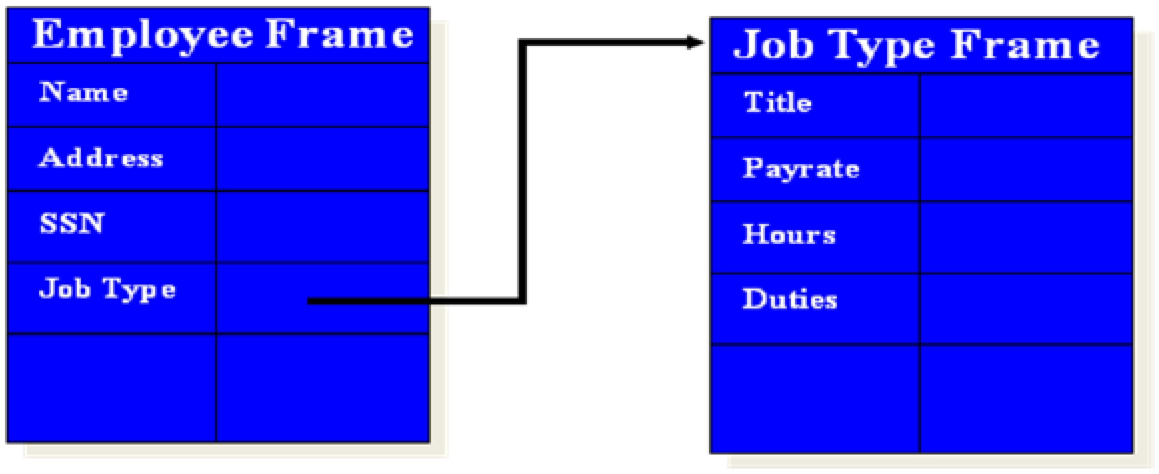
- explanation is difficult, expressive limitation

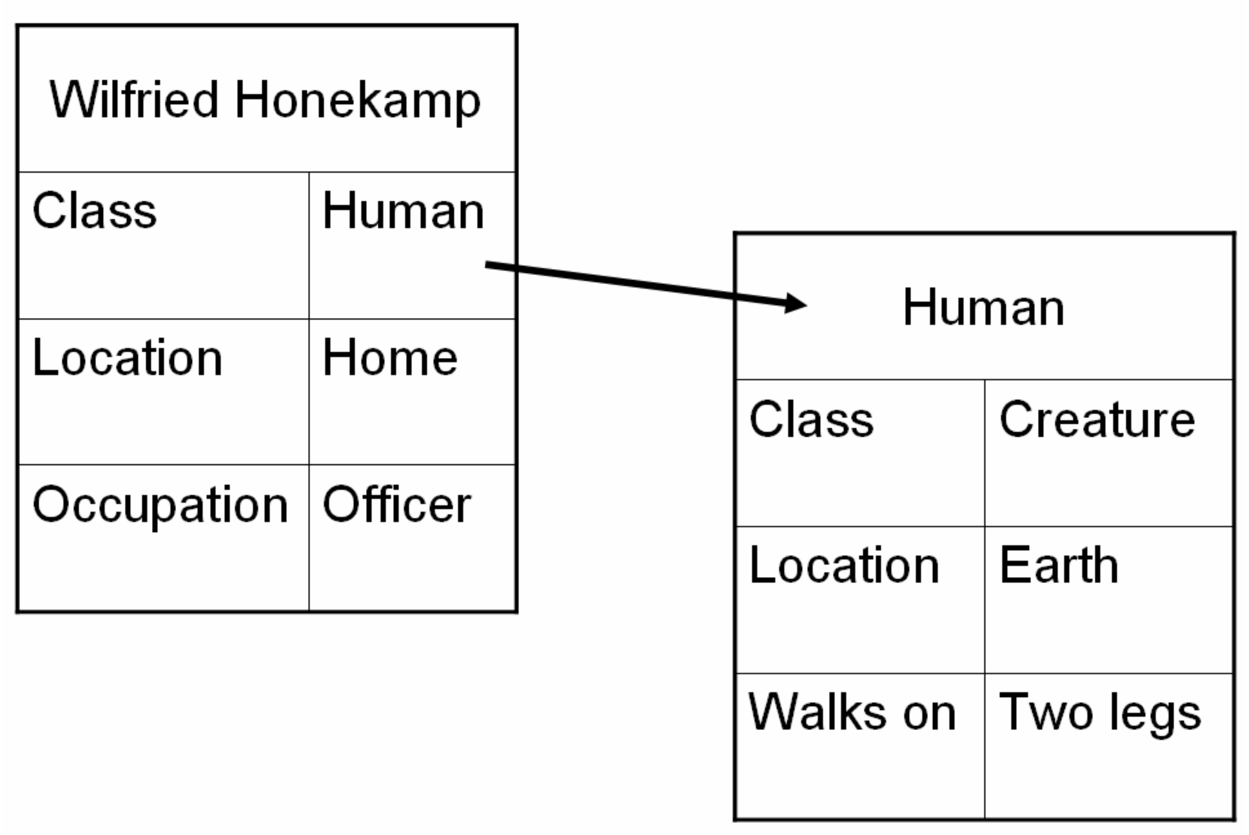
<u>Advantages</u>

- 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:





Semantic Network

- ✓ The meaning of concepts emerges from how it is connected to other concepts.
- ✓ Semantic networks can
    - show natural relationships between objects/concepts
    - be used to represent declarative/descriptive knowledge
- ✓ Knowledge is represented as a collection of concepts, represented by nodes. Thus, semantic networks are constructed using **nodes** linked by directional lines called **arcs**
- ✓ A node can represent a fact description
    - physical object
    - concept
    - event

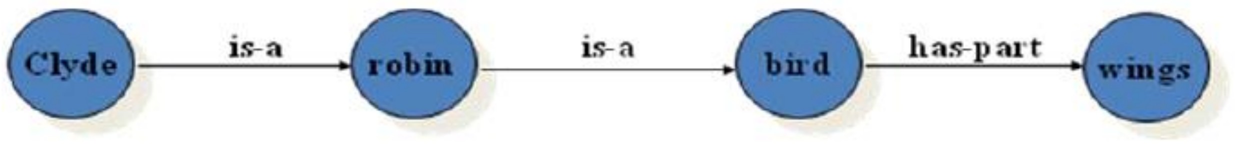
- ✓ An arc (or link) represents relationships between nodes. There are some 'standard' relationship types

  - 'Is-a' (instance relationship): represent class/instance relationships

  - 'Has-a' (part-subpart relationship): identify property relationships

- ✓ Semantic networks are mainly used as an aid to analysis to visually represent parts of the problem domain.
- ✓ One feature of a semantic net is the ability to use the net to deduce new facts

  Example:



Facts:

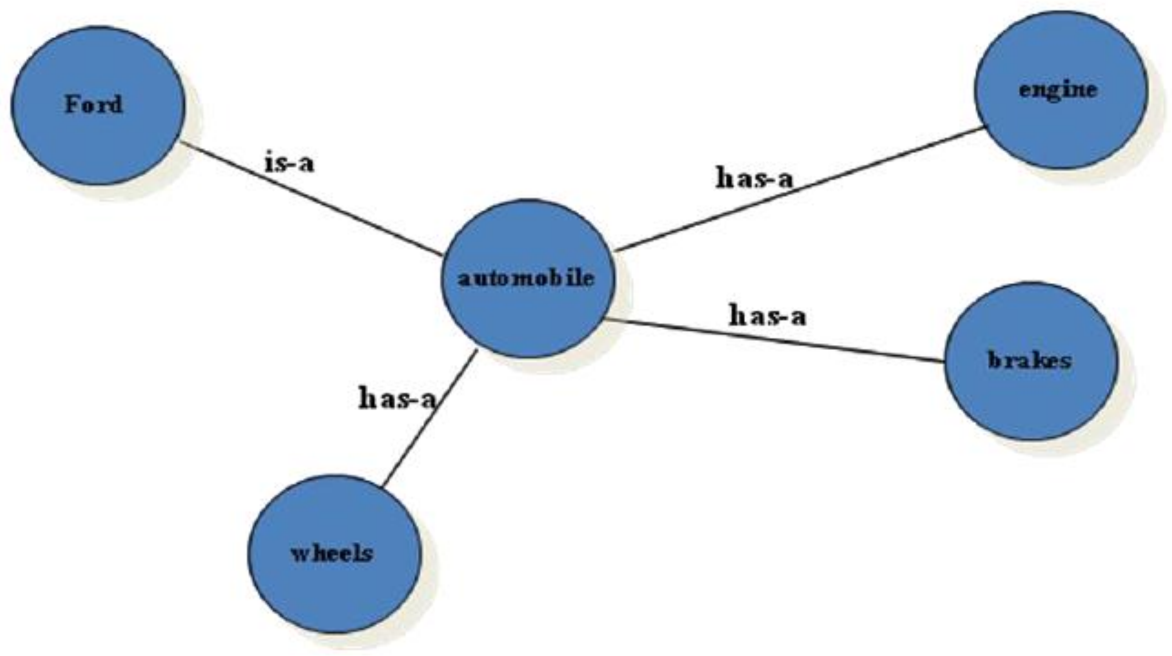
- all robins are birds

- If Clyde is the name of a particular pet robin then

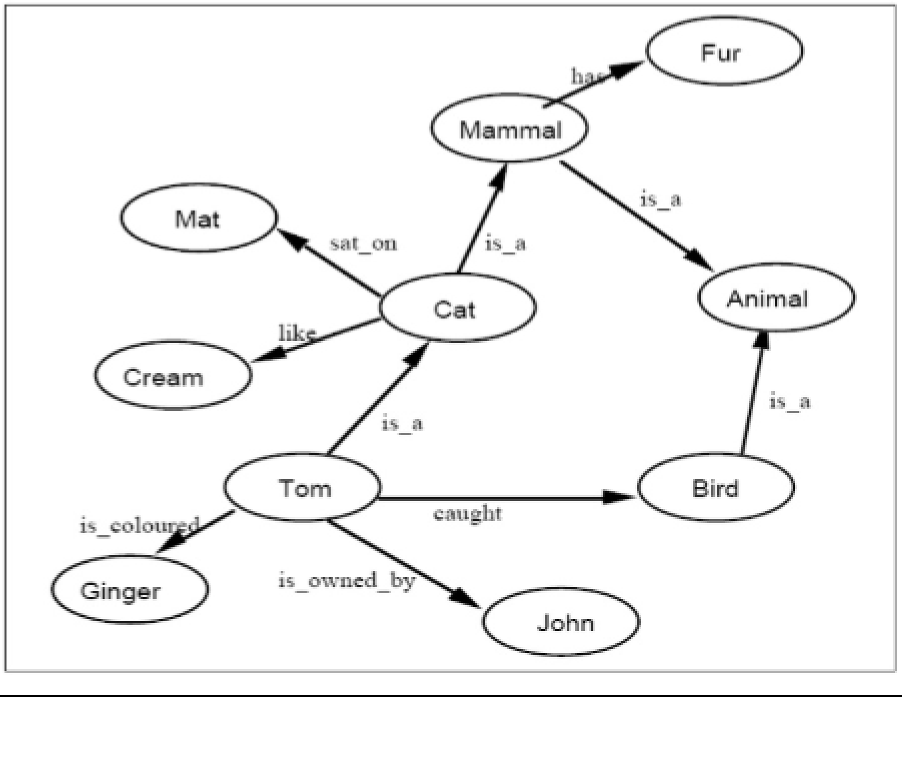
By following the is-a links it is easy to deduce that

- Clyde is a bird

Another example:



Another Example:



Disadvantages of a semantic network

- incomplete (no explicit operational/procedural knowledge)

- no interpretation standard

- lack of standards, ambiguity in node/link descriptions

- not temporal (i.e. doesn't represent time or sequence)

Advantages of a semantic network

- Explicit and easy to understand.

- The net is its own index – quick inference possible.

- Supports default reasoning in finite time.

- Focus on bigger units of knowledge and interconnectedness.

Conceptual Dependency (CD)

- Conceptual Dependency was 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.
- CD:

- focuses on concepts instead of syntax.
- focuses on understanding instead of structure.
- assumes inference is fundamental to understanding.
- introduced idea of a canonical meaning representation.
  - different words and structures represent the same concept.
  - language-independent meaning representation

Tenets of Conceptual Dependency Theory

- The representation of events is separate from the words used to encode them.
- The task of the understander is to represent the events underlying sentences, rather than the sentences themselves.
- Event representations are governed by a set of rules that involve fitting inputs into a predefined representation scheme.
- The rules for filling the slots of the representation are the basis of language understanding.

Conceptual Primitives

- Basic meaning elements that underlie the words that we use.
- Can be combined to represent complex meanings.
- An interlingual representation.

Goal: to represent meaning so that general rules can be applied, without duplicating information.

Theory: a small number of primitive actions can represent any sentence

Conceptual Primitives for Actions

ATRANS
-- Transfer of an abstract relationship. *e.g. give*.
PTRANS
-- Transfer of the physical location of an object. *e.g. go*.
PROPEL
-- Application of a physical force to an object. *e.g. push*.
MTRANS
-- Transfer of mental information. *e.g. tell*.
MBUILD
-- Construct new information from old. *e.g. decide*.

SPEAK
-- Utter a sound. *e.g. say*.
ATTEND
-- Focus a sense on a stimulus. *e.g. listen, watch*.
MOVE
-- Movement of a body part by owner. *e.g. punch, kick*.
GRASP
-- Actor grasping an object. *e.g. clutch*.
INGEST
-- Actor ingesting an object. *e.g. eat*.
EXPEL
-- Actor getting rid of an object from body. *e.g. ????*

Primitive conceptual categories

— Provides building blocks which are the set of allowable dependencies in the concepts in a sentence:

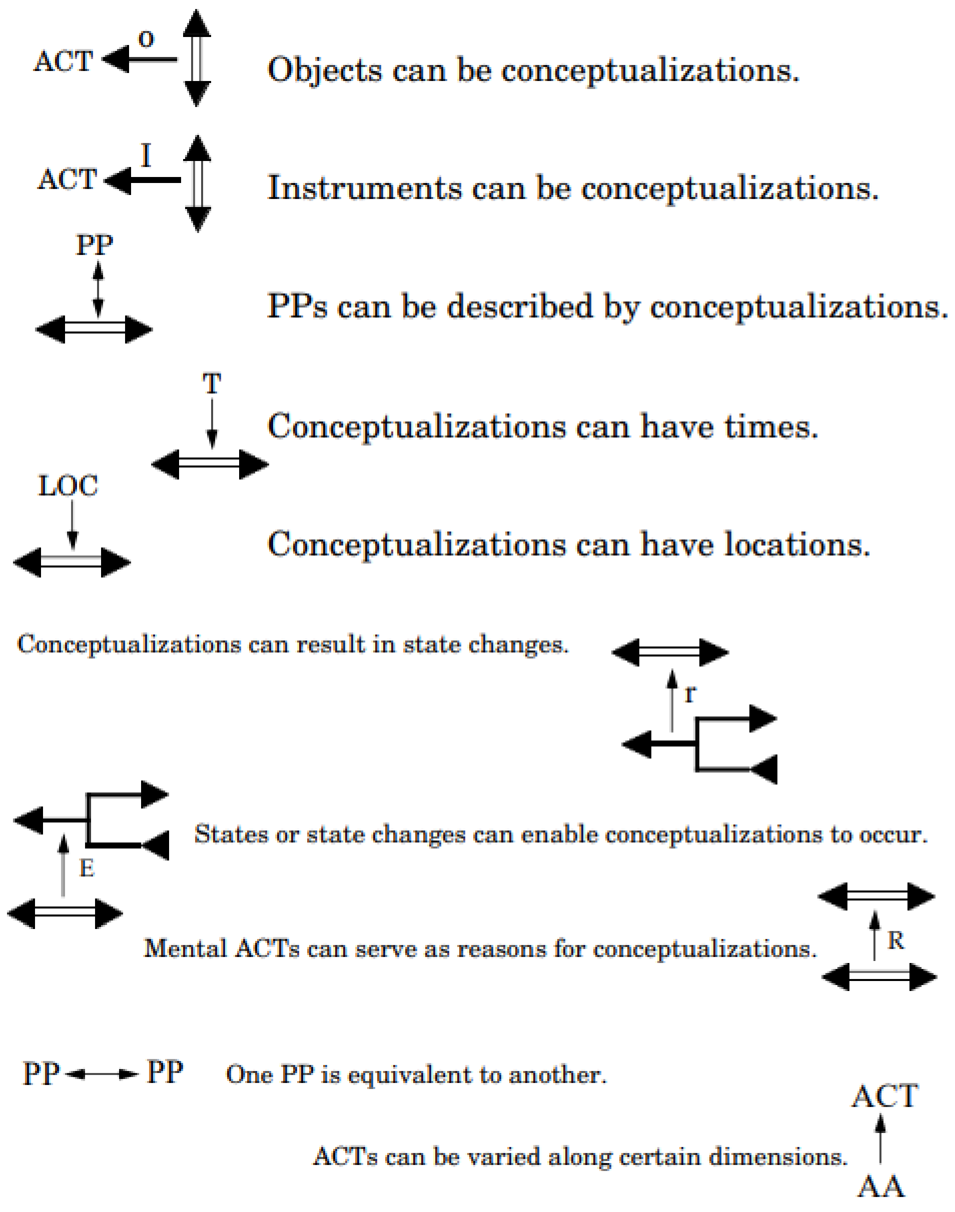
PP
-- Real world objects.
ACT
-- Real world actions.
PA
-- Attributes of objects.
AA
-- Attributes of actions.
T
-- Times.
LOC
-- Locations.

Conceptual Roles

- **Conceptualization:** The basic unit of the conceptual level of understanding.
- **Actor:** The performer of an ACT.
- **ACT:** An action done to an object.
- **Object:** A thing that is acted upon.
- **Recipient:** The reciever of an object as the result of an ACT.
- **Direction:** The location that an ACT is directed toward.
- **State:**The state that an object is in.

Conceptual Syntax Rules

$\text{PP} \Leftrightarrow \text{ACT}$	PPs can perform actions.
$\text{PP} \leftrightarrow \text{PA}$	PPs can be described by an attribute.
$\text{ACT} \xleftarrow{o} \text{PP}$	ACTs can have objects.
$\begin{tikzcd} & & \text{LOC} \\ \text{ACT} & {} \arrow[l, "D"'] \arrow[ur] & \\ & & \text{LOC} \arrow[ul] \end{tikzcd}$	ACTs can have directions.
$\begin{tikzcd} & & \text{PP} \\ \text{ACT} & {} \arrow[l, "R"'] \arrow[ur] & \\ & & \text{PP} \arrow[ul] \end{tikzcd}$	ACTs can have recipients.



Objects can be conceptualizations.

Instruments can be conceptualizations.

PPs can be described by conceptualizations.

Conceptualizations can have times.

Conceptualizations can have locations.

Conceptualizations can result in state changes.

States or state changes can enable conceptualizations to occur.

Mental ACTs can serve as reasons for conceptualizations.

One PP is equivalent to another.

ACTs can be varied along certain dimensions.

Here,

- Arrows indicate the direction of dependency.
- Letters above indicate certain relationships:

o	-- object.
R	-- recipient-donor.
I	-- instrument *e.g. eat with a spoon*.

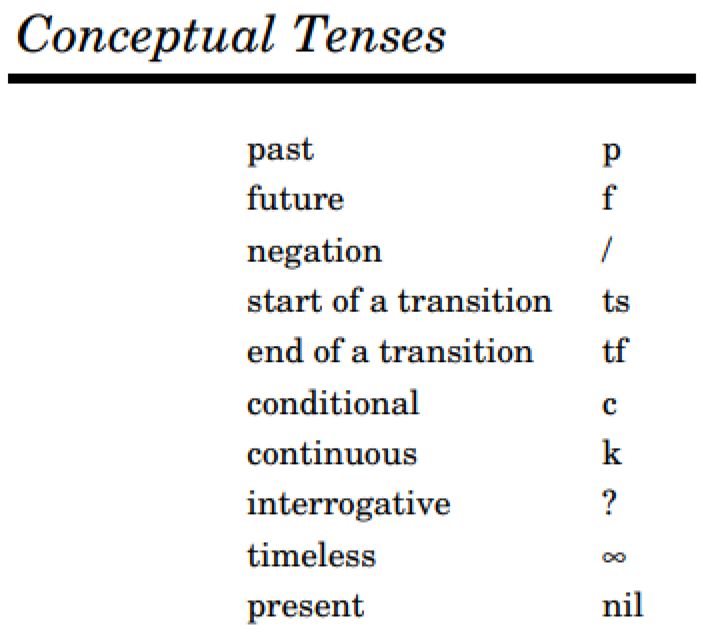
D

-- destination *e.g. going home*.

— Double arrows (⇔) indicate two-way links

— The actions are built from the set of primitive acts

- These can be modified by tense etc.
- The use of tense and mood in describing events is extremely important and schank introduced the following modifiers:



Conceptual Tenses

past	p
future	f
negation	/
start of a transition	ts
end of a transition	tf
conditional	c
continuous	k
interrogative	?
timeless	∞
present	nil

States

- states of objects are described by scales with numerical values.

HEALTH	(range -10 to 10)
dead	-10
gravely ill	-9
sick	-9 to -1
under the weather	-2
all right	0
tip top	+7
perfect health	+10

More States

FEAR	(range -10 to 0)
terrified	-9
scared	-5
anxious	-2
calm	0

MENTAL STATE	(range -10 to +10)
catatonic	-9
depressed	-5
upset	-3
sad	-2
ok	0
pleased	+2
happy	+5
ecstatic	+10

Combinations of States

CONSCIOUSNESS	(range 0 to +10)
unconscious	0
asleep	5
awake	10
"higher drug consciousness"	>10

Some words can be combinations of scales:

shocked =	SURPRISE (6)
	DISGUST (-5)
calm =	SURPRISE (0)
	DISGUST (0)
	FEAR (0)
	ANGER (0)
	CONSCIOUSNESS (> 0)

Some states are not scales ...

... some states take absolute values

LENGTH
COLOR
MASS
SPEED

... some states are just relationships between objects

CONTROL
PART (inalienable possession)
POSS (possession)
OWNERSHIP
CONTAIN
PROXIMITY

Examples

John gave Mary a book.

$$\begin{tikzcd}
\text{John} \arrow[r, Leftrightarrow, "P"] & \text{ATRANS} & \text{book} \arrow[l, "o"'] & \cdot \arrow[l, "R"'] \arrow[r] & \text{Mary} \\
& & & & \text{John} \arrow[ul]
\end{tikzcd}$$

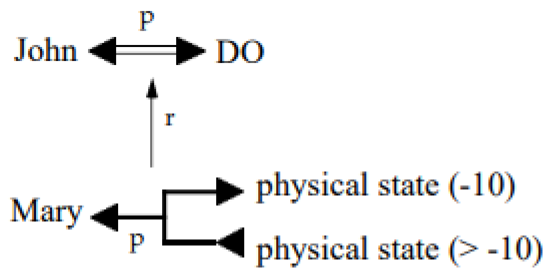
Mary took a book from John.

$$\begin{tikzcd}
\text{Mary} \arrow[r, Leftrightarrow, "P"] & \text{ATRANS} & \text{book} \arrow[l, "o"'] & \cdot \arrow[l, "R"'] \arrow[r] & \text{Mary} \\
& & & & \text{John} \arrow[ul]
\end{tikzcd}$$

John gave Mary a book by handing it to her

$$\begin{tikzcd}
\text{John} \arrow[r, Leftrightarrow, "P"] & \text{ATRANS} & \text{book} \arrow[l, "o"'] & \cdot \arrow[l, "R"'] \arrow[r] & \text{Mary} & \cdot \arrow[l, "I"'] & \text{John} \arrow[d, Leftrightarrow, "P"] \\
& & & & \text{John} \arrow[ul] & & \text{MOVE} \\
& & & & & & \text{hand} \arrow[u, "o"'] \\
& & & & & & \cdot \arrow[u, "D"'] \arrow[dr] \\
& & & & & \text{Y} \arrow[ur] & & \text{Mary}
\end{tikzcd}$$

John killed Mary.



This representation is used in natural language processing in order to represent them earning of the sentences in such a way that inference we can be made from the sentences. It is independent of the language in which the sentences were originally stated. CD representations of a sentence is built out of primitives , which are not words belonging to the language but are conceptual , these primitives are combined to form the meaning s of the words. As an example consider the event represented by the sentence.

Advantages of CD:

- Using these primitives involves fewer inference rules.
- Many inference rules are already represented in CD structure.
- The holes in the initial structure help to focus on the points still to be established.

Disadvantages of CD:

- Knowledge must be decomposed into fairly low level primitives.
- Impossible or difficult to find correct set of primitives.
- A lot of inference may still be required.
- Representations can be complex even for relatively simple actions. Consider:

  Dave bet Frank five pounds that Wales would win the Rugby World Cup.

  Complex representations require a lot of storage

Applications of CD:

MARGIE

(*Meaning Analysis, Response Generation and Inference on English*) -- model natural language understanding.

SAM

(*Script Applier Mechanism*) -- Scripts to understand stories. See next section.

PAM

(*Plan Applier Mechanism*) -- Scripts to understand stories.

Schank *et al.* developed all of the above.

Scripts

- A script is a structured representation describing a stereotyped sequence of events in a particular context.
- They were originally proposed as a means of providing contextual information to support natural language understanding.
- Scripts use a frame-like structure to represent the commonly occurring experience like going to the movies, eating in a restaurant, shopping in a supermarket, or visiting an ophthalmologist.
- Thus, a script is a structure that prescribes a set of circumstances that could be expected to follow on from one another.
- 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.

- .A script is composed of several components

  Components of a Script :

  - **Entry condition:** These are basic condition which must be fulfilled before events in the script can occur.
  - **Results:** Condition that will be true after events in script occurred.
  - **Props:** Slots representing objects involved in events
  - **Roles:** These are the actions that the individual participants perform.
  - **Track:** Variations on the script. Different tracks may share components of the same scripts.
  - **Scenes:** The sequence of events that occur.
- Describing a script, **special symbols of actions** are used. These are:

Symbol	**Meaning**	**Example**
ATRANS	transfer a relationship	give
PTRANS	transfer physical location of an object	go
PROPEL	apply physical force to an object	push
MOVE	move body part by owner	kick
GRASP	grab an object by an actor	hold
INGEST	taking an object by an animal eat	drink
EXPEL	expel from animal’s body	cry
MTRANS	transfer mental information	tell
MBUILD	mentally make new information	decide
CONC	conceptualize or think about an idea	think
SPEAK	produce sound	say
ATTEND	focus sense organ	listen

- Example: Bank Robbery

  This might involve:

  - Getting a gun.
  - Hold up a bank.
  - Escape with the money.

Here the *Props* might be

- Gun, *G*.
- Loot, *L*.
- Bag, *B*
- Get away car, *C*.

The *Roles* might be:

- Robber, *R*.
- Cashier, *M*.
- Bank Manager, *O*.
- Policeman, *P*.

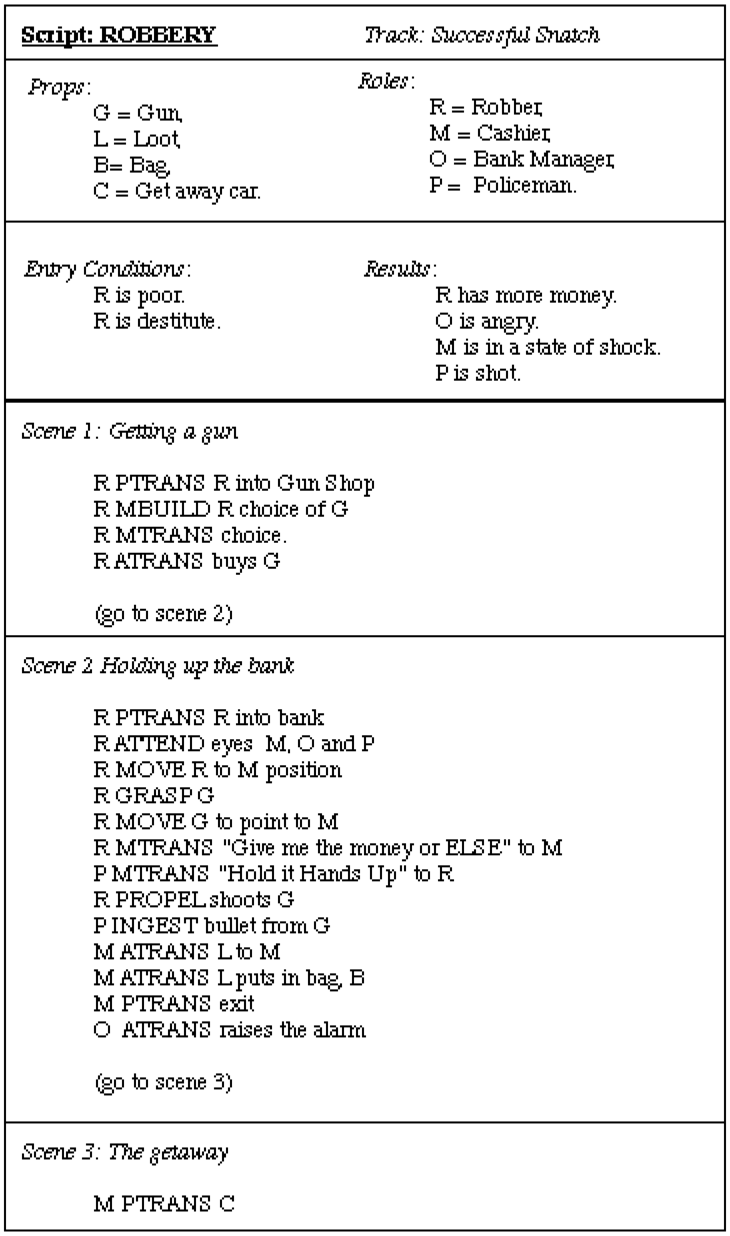
The *Entry Conditions* might be:

- *R* is poor.
- *R* is destitute.

The *Results* might be:

- *R* has more money.
- *O* is angry.
- *M* is in a state of shock.
- *P* is shot.

There are 3 scenes: obtaining the gun, robbing the bank and the getaway.



Script: ROBBERY *Track: Successful Snatch*

Props:
G = Gun,
L = Loot,
B= Bag,
C = Get away car.

Roles:
R = Robber,
M = Cashier,
O = Bank Manager,
P = Policeman.

Entry Conditions:
R is poor.
R is destitute.

Results:
R has more money.
O is angry.
M is in a state of shock.
P is shot.

Scene 1: Getting a gun

R PTRANS R into Gun Shop
R MBUILD R choice of G
R MTRANS choice.
R ATRANS buys G

(go to scene 2)

Scene 2 Holding up the bank

R PTRANS R into bank
R ATTEND eyes M, O and P
R MOVE R to M position
R GRASP G
R MOVE G to point to M
R MTRANS "Give me the money or ELSE" to M
P MTRANS "Hold it Hands Up" to R
R PROPEL shoots G
P INGEST bullet from G
M ATRANS L to M
M ATRANS L puts in bag, B
M PTRANS exit
O ATRANS raises the alarm

(go to scene 3)

Scene 3: The getaway

M PTRANS C

Task: Write Scripts for Restaurant Visit

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.

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.

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 **predicate logic**

Entailment:

Entailment means that one thing follows from another:

$$KB \models \alpha$$

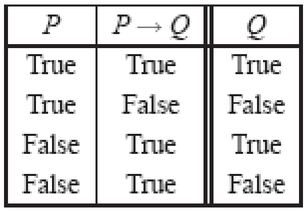
Knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true

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.

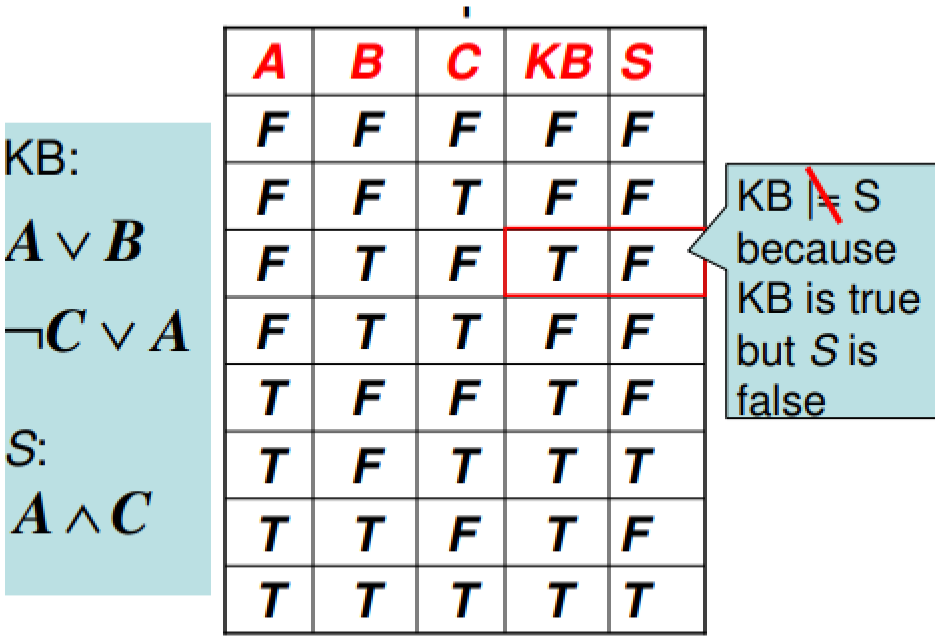
Example:



P	$P \to Q$	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}.

See the following examples:



KB:

$A \vee B$

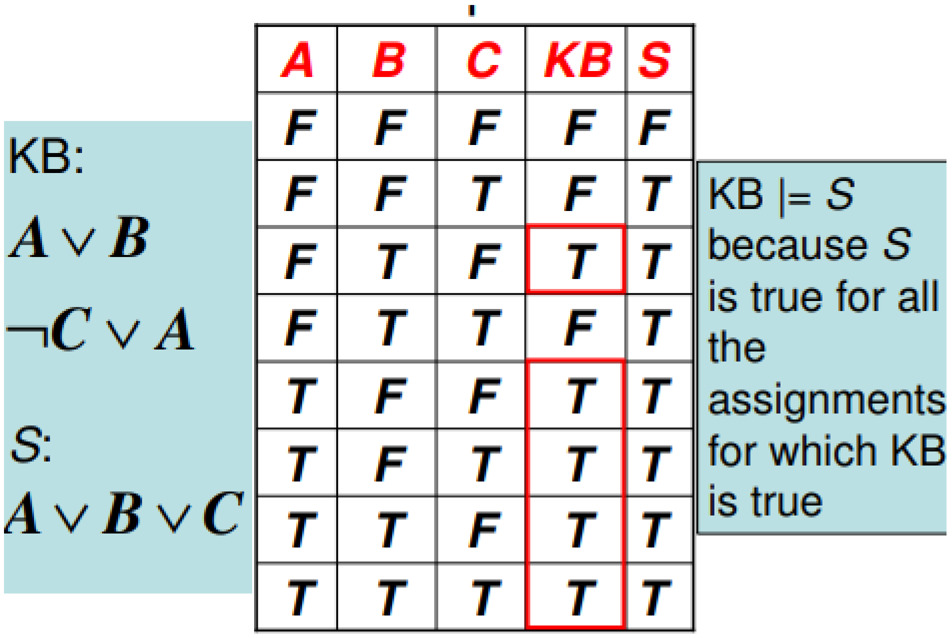
$\neg C \vee A$

S:

$A \wedge C$

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

KB $\not\models$ S because KB is true but *S* is false



KB:

$A \vee B$

$\neg C \vee A$

S:

$A \vee B \vee C$

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

KB \|= S because S is true for all the assignments for which KB is true

Note: We do not care about those models that evaluate KB to False. The result of evaluating S for these models is irrelevant

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.

- Assignment of true/false value to each of the symbol forms a model.

If a sentence α is true in model m, we say that m satisfies α or sometimes m is a model of α. We use the notation

M(α) to mean the set of all models of α.

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.

- Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions.
- A proposition is a declarative statement which is either true or false.
- It is a technique of knowledge representation in logical and mathematical form

Atomic and Compound propositions

Atomic Proposition: Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.

2+2 is 4, it is an atomic proposition as it is a true fact.

"The Sun is cold" is also a proposition as it is a false fact.

Compound proposition: Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives

"It is raining today, and street is wet."

"Ankit is a doctor, and his clinic is in Mumbai

Syntax of propositional logic:

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/compound sentences are constructed from simpler sentences using logical connectives. There are five logical connectives in common use:

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

1. **Negation:** A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.
2. **Conjunction:** A sentence which has ∧ connective such as, **P ∧ Q** is called a conjunction.
   Example: Rohan is intelligent and hardworking. It can be written as,
   P= Rohan is intelligent,
   Q= Rohan is hardworking. → P∧ Q.

3. **Disjunction:** A sentence which has ∨ connective, such as **P ∨ Q**. is called disjunction, where P and Q are the propositions.
   Example: "Ritika is a doctor or Engineer",
   Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as **P ∨ Q**.
4. **Implication:** A sentence such as P → Q, is called an implication. Implications are also known as if-then rules. It can be represented as
   If it is raining, then the street is wet.
   Let P= It is raining, and Q= Street is wet, so it is represented as P → Q
5. **Biconditional:** A sentence such as **P↔ Q is a Biconditional sentence, example If I am breathing, then I am alive**
   P= I am breathing, Q= I am alive, it can be represented as P ↔ Q.

The order of precedence in propositional logic is from (highest to lowest): ¬ , ^ , ∨, ⇒, ⇔.

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 ∨ S2 is a sentence (disjunction)

If S1 and S2 are sentences, S1 → S2 is a sentence (implication)

If S1 and S2 are sentences, S1↔S2 is a sentence (biconditional)

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^ Q)

- If it is raining, then I am indoors (P → Q)
- It is raining if I am indoors (Q→ P)
- It is raining if and only if I am indoors (P↔ Q)
- It is not raining (¬P)

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

Formal grammar for propositional logic can be given as below:

Sentence → AtomicSentence | ComplexSentence

AtomicSentence → True | False | P | Q | R | . . .

ComplexSentence → (Sentence) | [Sentence]

| ¬Sentence

| Sentence ∧ Sentence

| Sentence ∨ Sentence

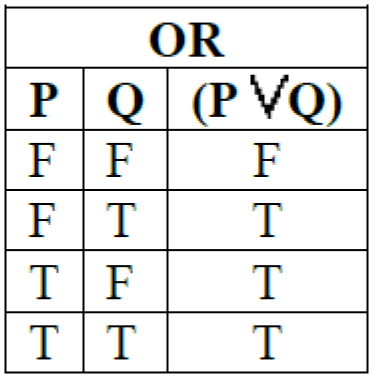
| Sentence ⇒ Sentence

| Sentence ⇔ Sentence

Truth Table:

A proposition in general contains a number of variables. For example (P ∨ 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

Semantics:

The semantics defines the rules for determining the truth of a sentence with respect to a particular model. In propositional logic, a model simply fixes truth value—true or false—for every proposition symbol.

The semantics for propositional logic must specify how to compute the truth value of any sentence, given a model. This is done recursively. All sentences are constructed from atomic sentences and the five connectives; therefore, we need to specify how to compute the truth of atomic sentences and how to compute the truth of sentences formed with each of the five connectives.

Atomic sentences are easy:

- True is true in every model and False is false in every model.
- The truth value of every other proposition symbol must be specified directly in the model.

For complex sentences, we have five rules, which hold for any sub sentences P and Q in any model m (here "iff" means "if and only if"):

- ¬P is true iff P is false in m.
- P ∧ Q is true iff both P and Q are true in m.
- P ∨ Q is true iff either P or Q is true in m.
- P ⇒ Q is true unless P is true and Q is false in m.
- P ⇔ Q is true iff P and Q are both true and both false in m.

Each model specifies true/false for each proposition symbol

Rules for evaluating truth with respect to a model:

Truth Table showing the evaluation of semantics of complex sentences:

$\neg$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

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 β are logically equivalent if they are true in the same set of models. We write this as $\alpha \equiv \beta$. For example, we can easily show (using truth tables) that P ∧ Q and Q ∧ P are logically equivalent; other equivalences are shown below. These equivalences play much the same role in logic as arithmetic identities do in ordinary mathematics. An alternative definition of equivalence is as follows: any two sentences α and β are equivalent only if each of them entails the other:

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$.

Standard logical equivalences.

(The symbols α, β, and γ stand for arbitrary sentences of propositional logic)

$(\alpha \land \beta) \equiv (\beta \land \alpha)$ **commutativity of ∧**

$(\alpha \lor \beta) \equiv (\beta \lor \alpha)$ **commutativity of ∨**

$((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma))$ **associativity of ∧**

$((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma))$ **associativity of ∨**

$\neg(\neg \alpha) \equiv \alpha$ **double-negation elimination**

$(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha)$ **contraposition**

$(\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta)$ **implication elimination**

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$ **biconditional elimination**

$\neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta)$ **De Morgan**

$\neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta)$ **De Morgan**

$(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma))$ **distributivity of ∧ over ∨**

$(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma))$ **distributivity of ∨ over ∧**

Validity:

A sentence is valid if it is true in all models. For example, the sentence $P \vee \neg P$ is valid. Valid sentences are also known as tautologies—they are necessarily true. Valid sentences are also known as **tautologies**. Every valid sentence is logically equivalent to True.

Tautology is a proposition which is always true. (eg. $P \vee \neg P$)

Contradiction is notation in formal logic which is always false. (eg. $P \wedge \neg P$)

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

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:

All men are mortal

Socrates is a man

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.

Satisfiability:

A sentence is **satisfiable** if it is true in, or satisfied by, some model

e.g., $A \vee B$, C

A sentence is unsatisfiable if it is true in no models

e.g., $A \neg \wedge A$

The problem of determining the satisfiability of sentences in propositional logic—the SAT problem—was the first problem proved to be NP-complete. Many problems in computer science are really satisfiability problems. For example, all the constraint satisfaction problems in previous chapter ask whether the constraints are satisfiable by some assignment.

Validity and satisfiablity are related concepts.

- α is valid iff $\neg\alpha$ is unsatisfiable
- α is satisfiable iff $\neg\alpha$ is not valid.

We also have the following useful result:

$\alpha \models \beta$ *if and only if the sentence* $(\alpha \wedge \neg\beta)$ *is unsatisfiable*

Proving β from α by checking the unsatisfiability of $(\alpha \wedge \neg\beta)$ corresponds exactly to the standard mathematical proof technique of reduction and absurdum (literally, “reduction to an absurd thing”). It is also called **proof by refutation or proof by contradiction**. One assumes a sentence β to be false and shows that this leads to a contradiction with known axioms α. This contradiction is exactly what is meant by saying that the sentence $(\alpha \wedge \neg\beta)$ is unsatisfiable

Satisfiability is connected to inference via the following:

$$KB \models \alpha \text{ if and only if } (KB \wedge \neg\alpha) \text{ is unsatisfiable}$$

Well-formed formula

Well-formed formula is a finite sequence of symbols from a given alphabet that is part of a formal language. A formal language can be identified with the set of formulas in the language. A formula is a syntactic object that can be given a semantic meaning by means of an interpretation.

Propositional logic uses a symbolic "language" to represent the logical structure, or form, of a compound proposition. Like any language, this symbolic language has rules of syntax—grammatical rules for putting symbols together in the right way. Any expression that obeys the syntactic rules of propositional logic is called a well-formed formula, or WFF of Propositional logic.

A well-formed formula (or wff) is any formula that is capable of generated by some conbination of following formation rules:

1. Every propositional variable/Letter of Propositioal Logic (PL) (eg. P,Q,R) is a wff.
2. If P is a wff, then ¬(P) is a wff.
3. If P and Q are wffs, then (P∧Q) is a wff.
4. If P and Q are wffs, then (P ∨ Q) is a wff.
5. If P and Q are wffs, then (P→Q) is a wff.
6. If P and Q are wffs, then (P ↔ Q) is a wff.
7. Nothing else is wff except that can be fromed by repeated use of rule 1-6.

Example of wffs in PL

- A
- (A→B)
- ¬((P ∨¬(Q)))
- ¬((P ∧¬(Q)))

Followings are not wffs in PL

- P¬

- PQ
- ∨¬(Q)

Inference rules in Propositional Logic

- ✓ Inference is the process of deriving new sentence from exisiting sentences
- ✓ In logic, a rule of inference, inference rule or transformation rule is a logical form consisting of a function which takes premises, analyzes their syntax, and returns conclusion
- ✓ Inference rules can be applied to derive a proof—a chain of conclusions that leads to the desired goal
- ✓ All of the logical equivalences can be used as inference rules
- ✓ The best-known rule is called **Modus Ponens** (Latin for mode that affirms) and is written

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

The notation means that, whenever any sentences of the form ⇒ β and α are given, then the sentence β can be inferred.

- ✓ Another useful inference rule is **And-Elimination**, which says that, from a conjunction, any of the conjuncts can be inferred:

$$\frac{\alpha \wedge \beta}{\alpha}$$

- ✓ Here are some more rules:

(AI) And-introduction

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

(OI) Or-introduction

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \cdots \vee \alpha_n}$$

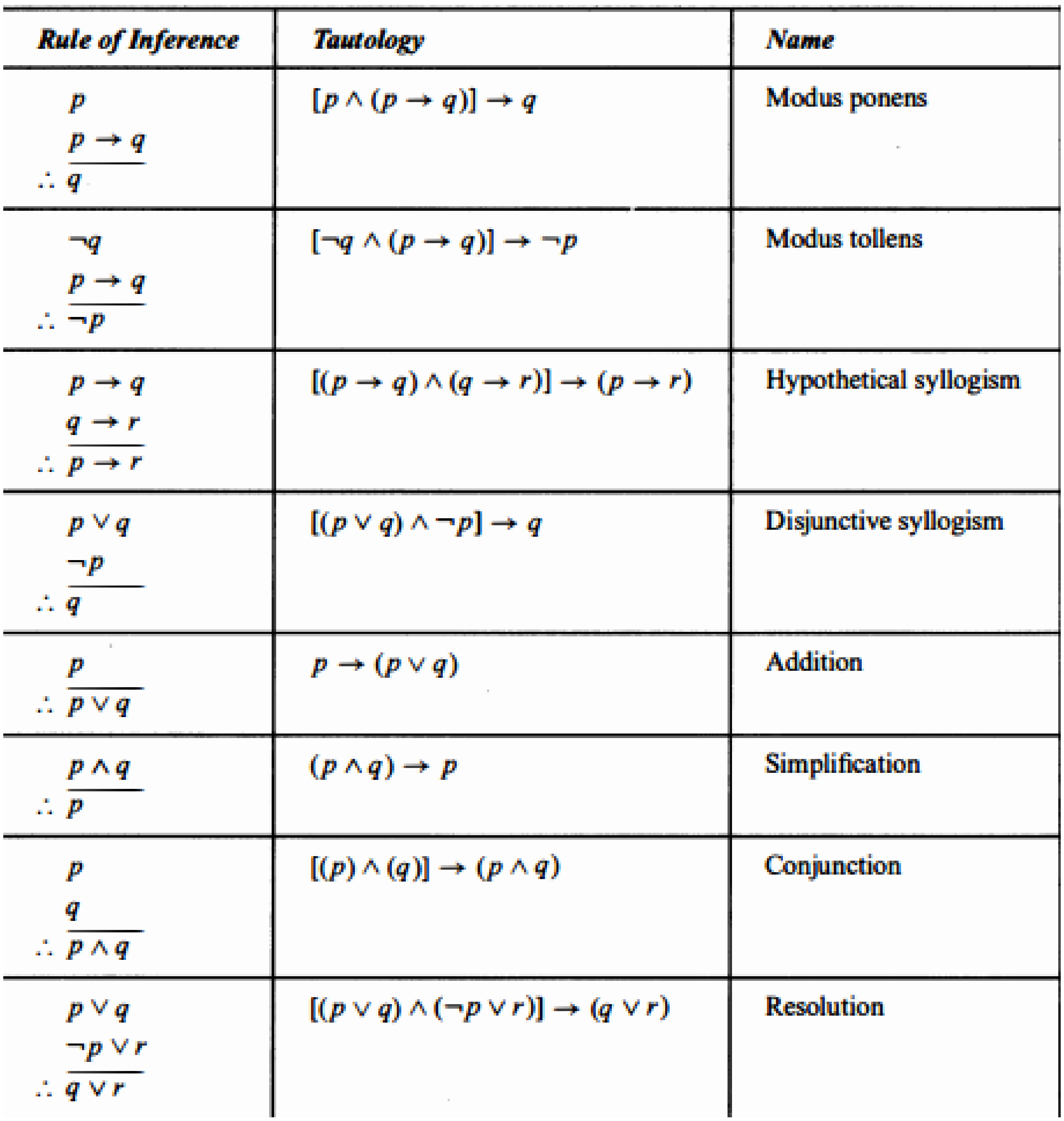
(AE) And-elimination:

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n}{\alpha_i}$$

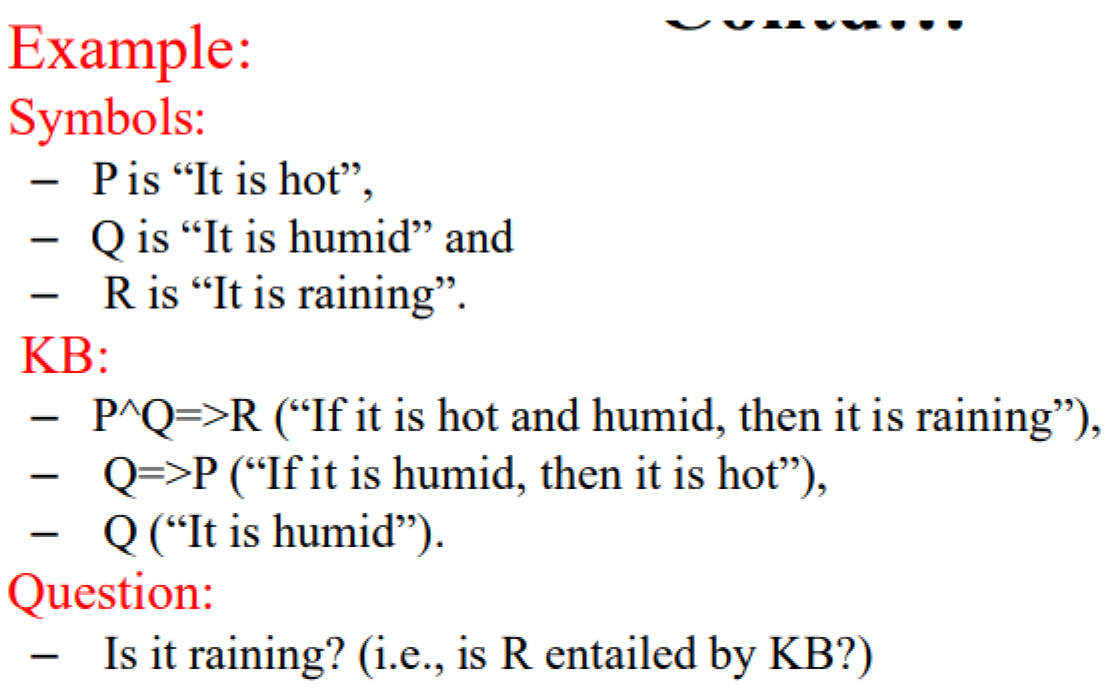
(NE) Negation-elimination

$$\frac{\neg\neg\alpha}{\alpha}$$

✓ Following table lists some other rules:



Rule of Inference	***Tautology***	***Name***
$\begin{array}{l} p \\ p \to q \\ \hline \therefore q \end{array}$	$[p \wedge (p \to q)] \to q$	Modus ponens
$\begin{array}{l} \neg q \\ p \to q \\ \hline \therefore \neg p \end{array}$	$[\neg q \wedge (p \to q)] \to \neg p$	Modus tollens
$\begin{array}{l} p \to q \\ q \to r \\ \hline \therefore p \to r \end{array}$	$[(p \to q) \wedge (q \to r)] \to (p \to r)$	Hypothetical syllogism
$\begin{array}{l} p \vee q \\ \neg p \\ \hline \therefore q \end{array}$	$[(p \vee q) \wedge \neg p] \to q$	Disjunctive syllogism
$\begin{array}{l} p \\ \hline \therefore p \vee q \end{array}$	$p \to (p \vee q)$	Addition
$\begin{array}{l} p \wedge q \\ \hline \therefore p \end{array}$	$(p \wedge q) \to p$	Simplification
$\begin{array}{l} p \\ q \\ \hline \therefore p \wedge q \end{array}$	$[(p) \wedge (q)] \to (p \wedge q)$	Conjunction
$\begin{array}{l} p \vee q \\ \neg p \vee r \\ \hline \therefore q \vee r \end{array}$	$[(p \vee q) \wedge (\neg p \vee r)] \to (q \vee r)$	Resolution



Example:

Symbols:

- P is “It is hot”,
- Q is “It is humid” and
- R is “It is raining”.

KB:

- P^Q=>R (“If it is hot and humid, then it is raining”),
- Q=>P (“If it is humid, then it is hot”),
- Q (“It is humid”).

Question:

- Is it raining? (i.e., is R entailed by KB?)

Steps		**Reason**
1	**Q**	**Premise**
2	**Q ⇒P**	**Premise**
3	**P**	**Modus Ponens (1,2)**
4.	**(P ∧ Q) ⇒ R**	**Premise**
5	**(P ∧ Q)**	**And Introduction (1,3)**
6	**R**	**Modus Ponens (4,5)**

So, R is entailed by the KB and we can conclude it is raining

Monotonicity:

In logic, it refers to the fact that a valid argument cannot be made invalid, nor an invalid argument made valid, by adding new premises. More precisely, monotonicity in logic is the property of obeying the extension theorem of semantic entailment.

The set of entailed sentences can only increase as information is added to the knowledge base. For any sentences α and β,

if KB |= α then KB ∧ β |= α .

Monotonicity means that inference rules can be applied whenever suitable premises are found in the knowledge base—the conclusion of the rule must follow regardless of what else is in the knowledge base.

Inference using Resolution

Some basic terminologies:

Literal: A literal is an atomic formula (atom) or its negation (eg. P, ¬P)

Clause: A disjunction of literals (eg. A∨B, A∨¬B, A∨B∨¬C)

Knowledge Base (KB) : A set of sentences (collection of knowledges)

- Each sentence is called sentence when it is not derived from other sentence
- If a question is asked to KB, the answer should follow from what has been told the KB previously.

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 each $l_i$is a literal and $l_i$and m are complementary literals (i.e., one is the negation of the other)

Generalized resolution rule:

The unit resolution rule can be generalized to the full resolution rule. Generalized resolution rule has the follwing form:

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

where $l_i$and $m_j$are complementary literals

Generalized resolution rule takes two clauses and produces a new clause containing all the literals of the two original clauses except the two complementary literals.

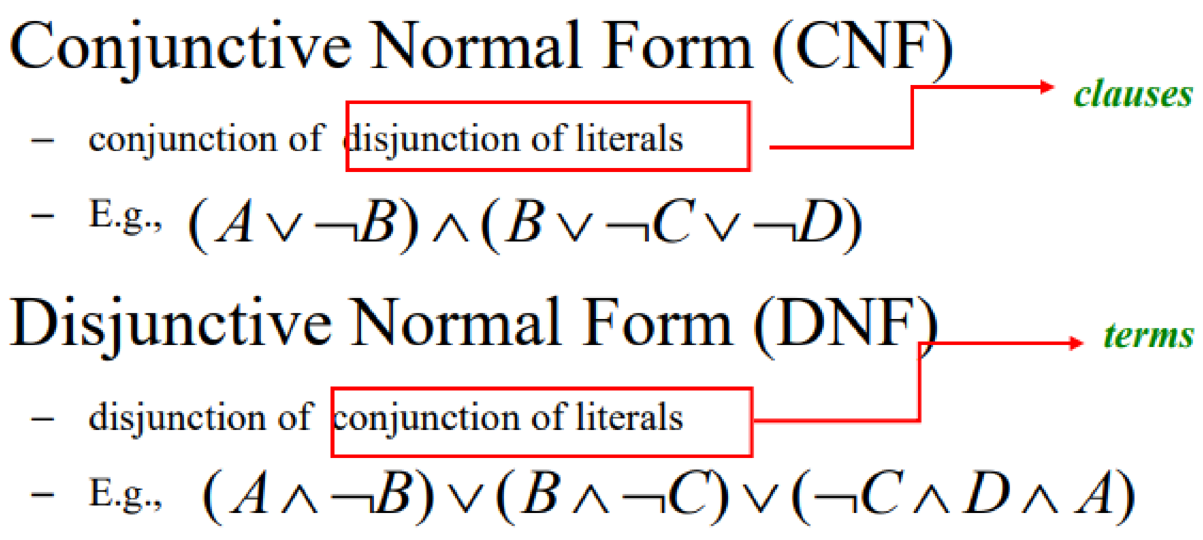
For example:

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

Normal forms:

There are two major normal forms of statements in propositional logic. They are :

- **Conjunctive Normal Form (CNF)**
- **Disjunctive Normal Form (DNF)**



Conjunctive Normal Form (CNF)

- conjunction of disjunction of literals → clauses
- E.g., $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$

Disjunctive Normal Form (DNF) → terms

- disjunction of conjunction of literals
- E.g., $(A \land \neg B) \lor (B \land \neg C) \lor (\neg C \land D \land A)$

Conjunctive Normal Form (CNF)

- Conjunction of disjunction of literals (clause)

Eg. $(A \lor B) \land (A \lor \neg B) \land (A \lor B \lor \neg C) \land D$

A sentence in CNF that contains only k literals per clause is said to be in k-CNF

$(A \lor B) \land (A \lor \neg B)$ is 2-CNF

Disjunctive Normal Form (DNF)

- Disjunction of conjunctions

Eg. $(A \land B) \lor (A \land \neg B) \lor (A \land B \land \neg C)$

Resolution Uses CNF (Conjunctive normal form)

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 (it cannot be used to enumerate true sentences.)

Conversion to CNF

The resolution rule applies only to clauses (that is, disjunctions of literals), so it would seem to be relevant only to knowledge bases and queries consisting of clauses. How, then, can it lead to a complete inference procedure for all of propositional logic? The answer is that every sentence of propositional logic is logically equivalent to a conjunction of clauses. A sentence expressed as a conjunction of clauses is said to be in conjunctive normal form or CNF.

Algorithm:

1. Eliminate ↔rewriting P↔Q as (P→Q)∧(Q→P)
2. Eliminate →rewriting P→Q as ¬P∨Q
3. Use De Morgan‘s laws to push ¬ inwards:

    - rewrite ¬(P∧Q) as ¬P∨¬Q

    - rewrite ¬(P∨Q) as ¬P∧¬Q

4. Eliminate double negations: rewrite ¬¬P as P
5. Use the distributive laws to get CNF:

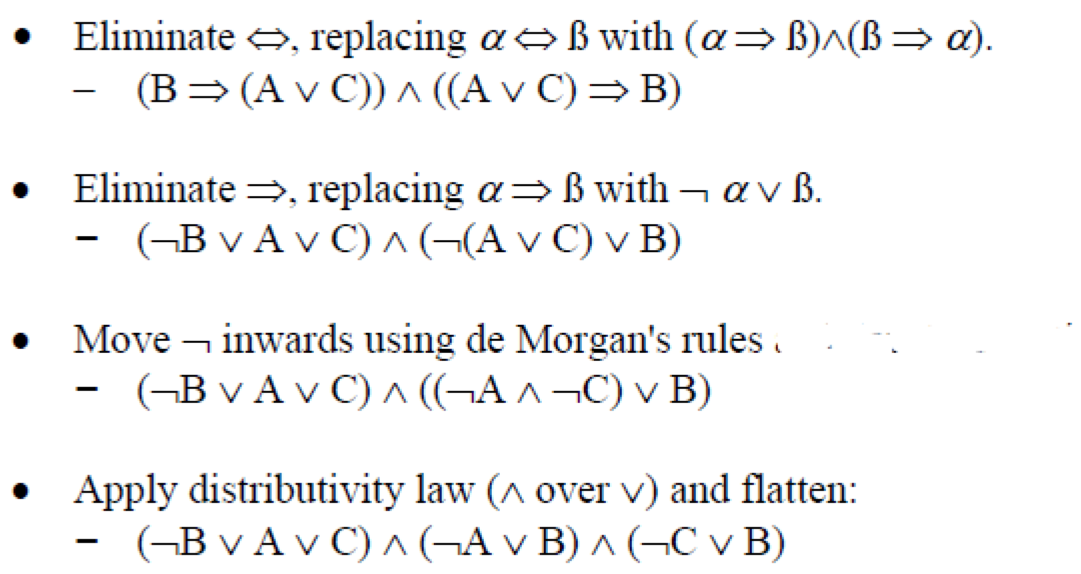
    - rewrite (P∧Q)∨R as (P∨R)∧(Q∨R)

6. Flatten :
    - rewrite (P∧Q) ∧ R as P∧Q ∧ R

    - rewrite (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 $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
  - $(B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B)$
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$.
  - $(\neg B \vee A \vee C) \wedge (\neg(A \vee C) \vee B)$
- Move $\neg$ inwards using de Morgan's rules
  - $(\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B)$
- Apply distributivity law ($\wedge$ over $\vee$) and flatten:
  - $(\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B)$

Exercise: Translate following into propositional logic first and then convert into CNF

"If James does not die then Mary will not get any money and Jame's family will be happy"

Solution: Let's Suppose,

P= James dies.

Q= Mary will get money.

R= Jame's family will be happy.

The propositional sentence of given English sentence is :

$$\neg \boldsymbol{P} \rightarrow (\neg \boldsymbol{Q} \wedge \boldsymbol{R})$$

Now Converting into CNF,

- Eliminating *implication* $(\rightarrow)$

$$\neg(\neg \boldsymbol{P}) \vee (\neg \boldsymbol{Q} \wedge \boldsymbol{R})$$

- Eliminating double negation

$$\boldsymbol{P} \vee (\neg \boldsymbol{Q} \wedge \boldsymbol{R})$$

- Using Distributive law

$$(\mathbf{P} \vee \neg \boldsymbol{Q}) \wedge (\mathbf{P} \vee \boldsymbol{R})$$

Which is in CNF.

Exercise: Convert into propositional logic and then into CNF

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

Solution: Let q, r, and s represent “You can ride the roller coaster,” “You are under 4 feet tall,”

and “You are older than 16 years old,” respectively. Then the sentence can be translated to

$$(r \land \neg s) \rightarrow \neg q.$$

(Now convert to CNF)

Resolution algorithm

– Convert KB into CNF

– Add negation of sentence to be entailed into KB i.e. ($KB \land \neg \alpha$, here α is to be entailed)

– 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

or,

– Two clauses resolve to entail the empty clause.

Hence KB entails α

Example: Consider the knowledge base given as:

$KB = (B \Leftrightarrow (A \lor C)) \land \neg B$

Prove that ¬ A can be inferred from above KB by using resolution.

Solution:

At first, convert KB into CNF

$B \Rightarrow (A \lor C)) \land ((A \lor C) \Rightarrow B) \land \neg B$

$(\neg B \lor A \lor C) \land (\neg(A \lor C) \lor B) \land \neg B$

$(\neg B \lor A \lor C) \land ((\neg A \land \neg C) \lor B) \land \neg B$

$(\neg B \lor A \lor C) \land (\neg A \lor B) \land (\neg C \lor B) \land \neg B$

Add negation of sentence to be inferred from KB into KB

Now KB contains following sentences all in CNF

$(\neg B \lor A \lor C)$

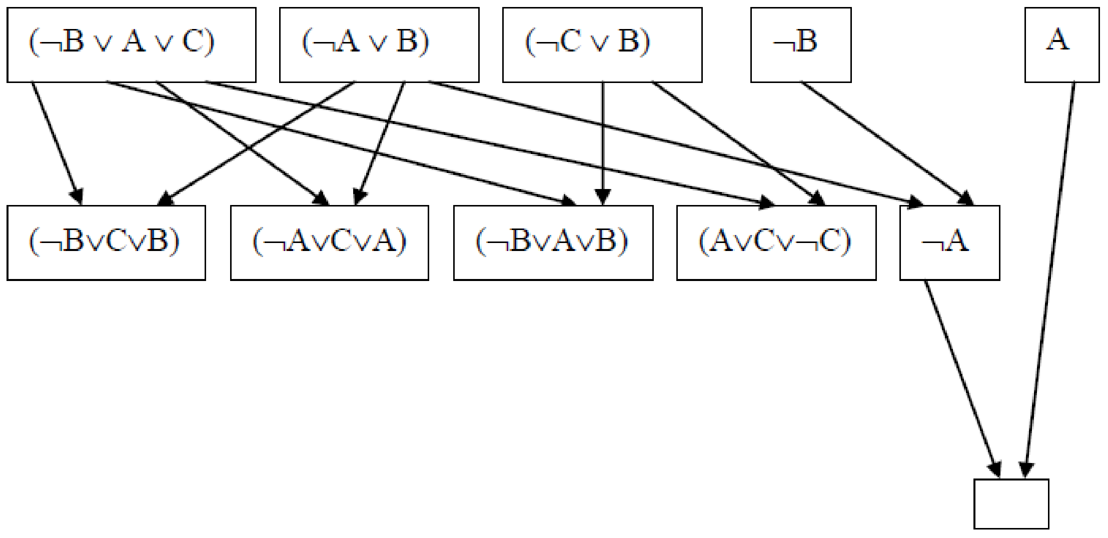
$(\neg A \lor B)$

$(\neg C \lor B)$

$\neg B$

A (negation of conclusion to be proved)

Now use Resolution algorithm



Resolution: More Examples

1. **KB= {(G∨H)→(¬J ∧¬K), G}. Show that KB ⊢ ¬J**

Solution:

Clausal form of (G∨H)→(¬J ∧¬K) is

{¬G∨¬J, ¬H ∨¬J, ¬G∨¬K, ¬H ∨¬K}

1. ¬G∨¬J [Premise]

2. ¬H ∨¬J [Premise]

3. ¬G∨¬K [Premise]

4. ¬H ∨¬K [Premise]

5. G [Premise]

6. J [¬ Conclusion]

7. ¬G [1, 6 Resolution]

8. _ [5, 7 Resolution]

Hence KB entails ¬J

2. **KB= {*P*→ ¬ *Q*, ¬ *Q*→*R*}. Show that KB ⊢ *P*→*R***

Solution:

1. ¬P∨¬Q [Premise]

2. Q∨R [Premise]

3. P [¬ Conclusion]

4. ¬R [¬ Conclusion]

5. ¬Q [1, 3 Resolution]

6. R [2, 5 Resolution]

7. _ [4, 6 Resolution]

Hence, KB ⊢ P→R

3. . ⊢ ((P∨Q)∧¬P)→Q

Clausal form of ¬(((P∨Q)∧¬P)→Q) is {P∨Q, ¬P, ¬Q}

1. P∨Q [¬ Conclusion]

2. ¬P [¬ Conclusion]

3. ¬Q [¬ Conclusion]

4. Q [1, 2 Resolution]

5. _ [3, 4 Resolution]

Exercise: Use resolution to show that hypotheses "*It is not raining or Sangita has her umbrella*," "*Sangita does not have umbrella or she does not get wet*," and " *It is raining or Sangita does not get wet*" imply that "*Sangita does not get wet*."

Solution:

Let P = "It is raining."

Q = "Sangita has her umbrella."

R = "Sangita gets wet"

Sentences in KB are:

1. $\neg P \vee Q$
2. $\neg Q \vee \neg R$
3. $P \vee \neg R$

Sentence to prove: $\neg R$

Here every sentences are in CNF.

Now adding negation of conclusion into CNF

4. R

Applying resolution in 1 and 2

5. $\neg P \vee \neg R$

Applying resolution in 3 and 5

6. $\neg R$

Applying resolution in 4 and 6

7. _

Hence KB entails $\neg R$

Definite clause and Horn Clause

The completeness of resolution makes it a very important inference method. In many practical situations, however, the full power of resolution is not needed. Some real-world knowledge bases satisfy certain restrictions on the form of sentences they contain, which enables them to use a more restricted and efficient inference algorithm.

One such restricted form is the **definite clause**, which is a disjunction of literals of which *exactly one is positive.*

For example, the clause ($\neg$P ∨ Q ∨$\neg$R) is a definite clause, whereas ($\neg$P ∨ Q, ∨ R) is not.

Slightly more general is the **Horn clause**, which is a disjunction of literals of which *at most one is positive.*

- All definite clauses are Horn clauses.
- Horn clause with no positive literals is called **goal clause**
- If you resolve two Horn clauses, you get back a Horn clause

Horn clause and definite clause are the forms of sentences, which enables knowledge base to use a more restricted and efficient inference algorithm. Logical inference algorithms use forward and backward chaining approaches, which require KB in the form of the first-order definite clause.

Definite clause: A clause which is a disjunction of literals with exactly one positive literal is known as a definite clause or strict horn clause.

Horn clause: A clause which is a disjunction of literals with at most one positive literal is known as horn clause. Hence all the definite clauses are horn clauses.

Example: $(\neg p \vee \neg q \vee k)$. It has only one positive literal k.

It is equivalent to $p \wedge q \rightarrow k$.

Three important properties of Horn clause are:

1. Can be written as an implication
2. Inference can be done through forward chaining and backward chaining.
3. Deciding entailment can be done in time that is linear in the size of the knowledge base

Forward and backward chaining

Forward Chaining

It begins from known facts (positive literals) in the knowledge base. If all the premises of an implication are known, then its conclusion is added to the set of known facts

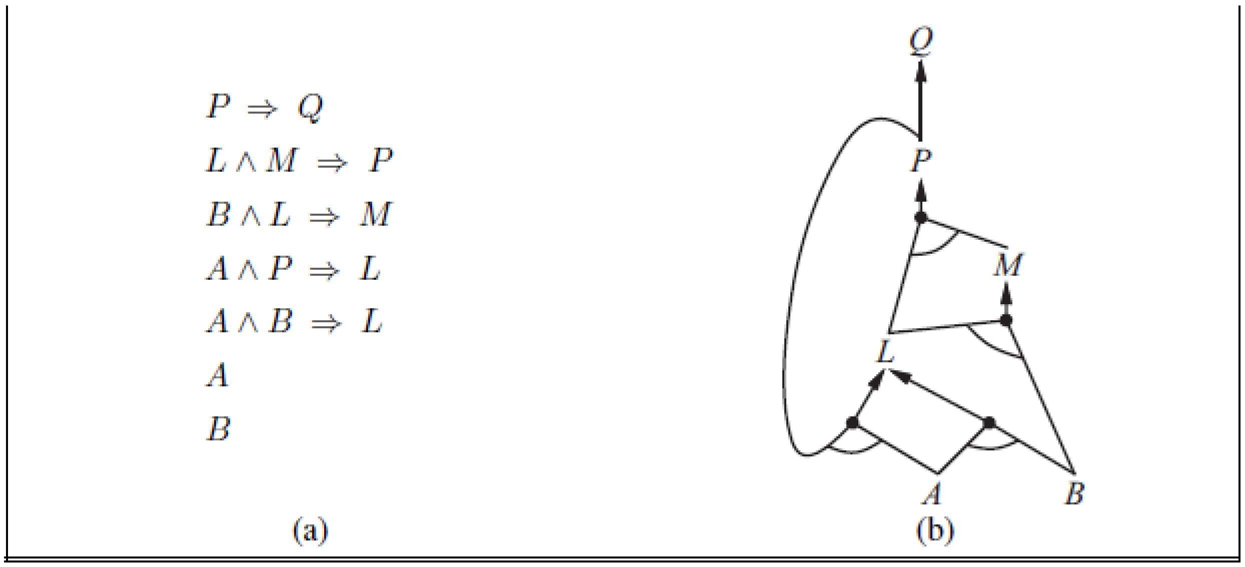

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B
(a)
Q
P
M
L
A
B
(b)

Figure (a) A set of Horn clauses. (b) The corresponding AND–OR graph.

In AND–OR graphs, multiple links joined by an arc indicate a conjunction—every link must be proved—while multiple links without an arc indicate a disjunction—any link can be proved. It is easy to see how forward chaining works in the graph. The known leaves (here, A and B) are set, and inference propagates up the graph as far as possible. Wherever a conjunction appears, the propagation waits until all the conjuncts are known before proceeding.

- Forward chaining is also known as a forward deduction or forward reasoning method when using an inference engine. Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.
- The Forward-chaining algorithm starts from known facts, triggers all rules whose premises are satisfied, and add their conclusion to the known facts. This process repeats until the problem is solved.
- Forward Chaining→ Conclude from "A" and "A implies B" to "B".

Example:

It is raining. (p)

If it is raining, the street is wet.(p->q)

The street is wet. (q) [Conclusion]

Properties of Forward-Chaining:

- It is a down-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward-chaining approach is also called as *data-driven* as we reach to the goal using available data.
- Forward -chaining approach is commonly used in the expert system

Backward Chaining

Backward-chaining is also known as a backward deduction or backward reasoning method when using an inference engine. A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

- Backward Chaining→ Conclude from "B" and "A implies B" to "A".
  Example

The street is wet. (q) [Conclusion]

If it is raining, the street is wet.(p->q)

It is raining. (p)

For example, for following KB ,Prove that Q can be inferred

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

<u>Solution:</u>

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>Properties of backward chaining:</u>

- It is known as a top-down approach.
- Backward-chaining is based on modus ponens inference rule.
- In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.
- It is called a *goal-driven approach,* as a list of goals decides which rules are selected and used.

- Backward -chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.

Limitations of Propositional logic:

- We cannot represent relations ALL, some, or none with propositional logic. Example:
    - All the girls are intelligent.
    - Some apples are sweet.
- Propositional logic has limited expressive power.
- In propositional logic, we cannot describe statements in terms of their properties or logical relationships.