Artificial Intelligence

knowledge representation

Dr. Priyadarshan Dhabe,
Ph.D, IIT Bombay,
Assistant Professor in Information Technology

Syllabus-Module 4

- **Knowledge Representation**

Knowledge based agents, Wumpus world, **Propositional logic:** Representation, inference, Reasoning patterns, , Resolution, Forward and backward chaining, **First order Logic:** Representation, inference, Reasoning patterns, , Resolution, Forward and backward chaining,

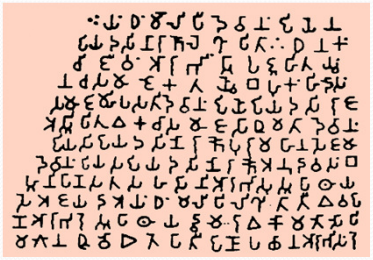
Basics of PROLOG: Representation, Structure, Backtracking

Expert system: Case study of expert system in PROLOG

Knowledge Representation-basics

- **Deals with**
  - **Facts:-** truths in some relevant world. These are the things we want to represent.
  - **Representations of facts:-** in some chosen formalism.
- **Representation of facts** can be done in various ways
  - Plain English/ Mother tongue
  - In Computers we use "**Symbols**"
  - Mathematical logic (Propositional and Predicate Logic)

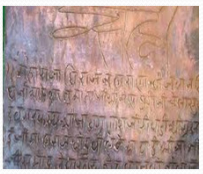
Knowledge Representation-scripts in ancient India



Bramhi script- wikipedia

STORAGE media

Kamal Patra

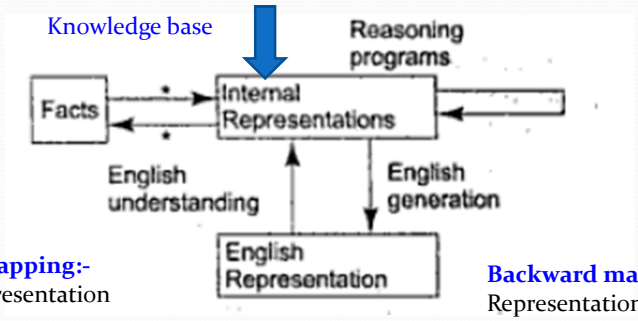


Tamra Patra

Sheela Lekh

Knowledge Representation-basic

- **Mappings** between **facts** and **Representation** also called as "*Representation mappings*"

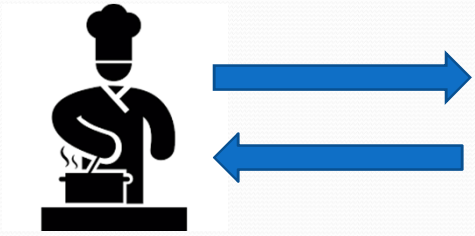


Forward mapping:-
Facts to representation

Backward mapping:-
Representation to Facts

FACT-Representation mapping

FACT REPRESENTATION



I am cooking

Knowledge Representation-basic

- Consider following facts written in English and their representations in ***predicate logic***.

Tommy is a dog.

$dog(Tommy)$

All dogs have tails.

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

Using deductive mechanism we can generate new fact.

$hastail(Tommy)$

Knowledge Based Agents

- That uses Representation of knowledge and reasoning processes to bring knowledge in life.
- The central component of knowledge based agents is "*Knowledge base*" (KB).
- Informally, KB, is set of "*Sentence*" (Sentence is a technical term and is related to *English sentences* but not identical)
- Each *sentence* is expressed in a language called a "*Knowledge Representation Language*"

Knowledge Based Agents

- Operations on **Knowledge Base** (**KB**)
  - **Add** new sentences (standard name is **TELL**)
  - **Query** KB (standard name is **ASK**)
  - **Inference**:- Both TELL and ASK may involve "*inference*"- Deriving new sentence from old (existing) by following systematic process.

Outline of Knowledge Based Agents program

```
function KB-AGENT(percept) returns an action
  persistent: KB, a knowledge base
              t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

Figure 7.1 A generic knowledge-based agent. Given a percept, the agent adds the percept to its knowledge base, asks the knowledge base for the best action, and tells the knowledge base that it has in fact taken that action.

-It takes "Percept" as input and returns "action" as output

- Each time program does the two things. First it "TELL"s (Add) the KB what it perceives. Second, it ASKs (query) KB "*What action it should perform*".
- The second TELL is necessary to know the agent that the action is already taken

Outline of Knowledge Based Agents program

```
function KB-AGENT(percept) returns an action
  persistent: KB, a knowledge base
              t, a counter, initially 0, indicating time
  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

Figure 7.1 A generic knowledge-based agent. Given a percept, the agent adds the percept to its knowledge base, asks the knowledge base for the best action, and tells the knowledge base that it has in fact taken that action.

MAKE-PPERCEPT-SENTENCE:-

- Takes a percept and a time and returns a sentence (representation) asserting that agent perceived the percept at given time.
- Inputs can be Audio, Video, combinations using various sensors

MAKE-ACTION-QUERY:-

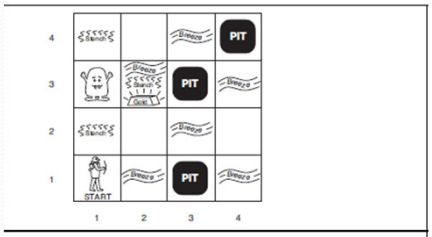
- Takes a time as input and returns a sentence (representation) that asks what action should be performed at that time.

The Wumpus World

Wumpus- a mysterious monster

- "*Hunt the Wumpus game*" (https://www.youtube.com/watch?v=xGVOw8gXl6Y)
- The *Wumpus world* is a cave consisting of rooms connected by passageways. Lurking (hiding) some where in the cave is the Wumpus, a beast that eats anyone who enters its room. The Wumpus can be shot by an agent, but the agent has only one arrow. Some rooms contains bottomless pits (holes) that will trap anyone who wanders into these rooms except Wumpus, since it is too big to fall in. An agent can also find a heap of gold.

The Wumpus World



A typical wumpus world. The agent is in the bottom left corner, facing right.

3. Actuators:- agent can move forward, turn left and right by 90 degrees. Agent dies if enters into a room with pit or wumpus. Action Grab can be used to pick the object in the same square. Action shoot can be used to fire an (only) arrow in the straight direction agent is facing.

PEAS (Performance measure, Environment, Actuators, Sensors) description of task environment

1. Performance measure:- +1000 for picking up the gold, -1000 for falling into pit or being eaten by wumpus, -1 for each action taken and -10 for the using up arrow.
2. Environment:- a 4 x 4 grid of rooms. The agent always start in the square labeled [1,1,], facing the right. The locations of gold and wumpus are chosen randomly from other than start square.

The Wumpus World

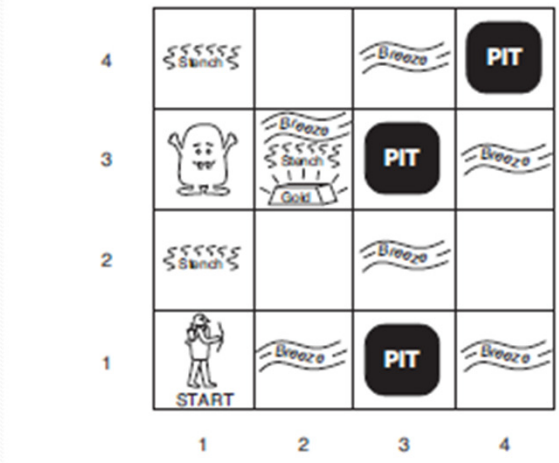
- **Sensors:-** The agent has 5 sensors

1. In the square containing the wumpus and in directly adjacent (not diagonally) squares agent will perceive a **stench** (strong and unpleasant smell).
2. In the square directly adjacent to pit, agent will perceive **breeze** (a gentle wind)
3. In the square where the gold is, agent will perceive a **glitter** (bright reflected light)
4. When agent walks into a wall, it will perceive a **bump** (collision)
5. When the wumpus is killed , it emits a **woeful** (sad) **scream** (loud sound) that can be perceived any-where in the cave

The percepts will be given to the agent in the following form, if there is a stench, breeze but no glitter, bump or scream

[Stench, Breeze, None, None, None]

The Wumpus World



A typical wumpus world. The agent is in the bottom left corner, facing right.

Knowledge based wumpus Agent

- Agent has to retrieve gold safely.
- Initial knowledge Base contains
  - Rules of the environment e.g [1,1] is safe square
- Agents knowledge **evolves** as new percepts arrives and actions are taken.
- The first percept is [None, None, None, None, None], from which agent concludes that its neighboring squares (not diagonal) are safe. Fig. 1 shows this situation. We use sentence "OK" to indicate safety of squares.

Knowledge based wumpus Agent

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2	3,2	4,2
1,1 A OK	2,1 OK	3,1	4,1

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

Fig. 1

Since there was no **stench** or **breeze** in [1,1], the agent can infer that [1,2] and [2,1] are free from danger and thus marked with OK and will try to move in those squares only.

Knowledge based wumpus Agent

- Let agent decided to move to [2,1].
- Let agent detect a **breeze** in [2,1] i.e there can be pit in [2,2] or [3,1], as shown in fig.2

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 A B OK	3,1 P?	4,1

Fig. 2

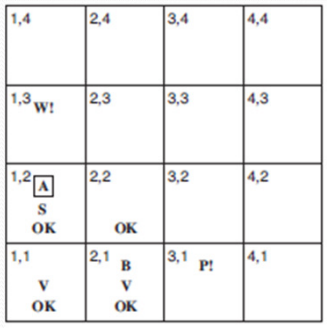
Agent then can go back to [1,1] and then proceed to [1,2]

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

Knowledge based wumpus Agent

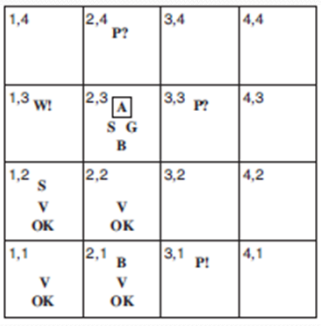
- Let new percept in [1,2] is **stench**., means there can be wumpus near by, but wumpus can not be [1,1] and [2,2], since they are already marked OK (known). Thus, agent can infer that wumpus can be in [1,3] and indicated by w!

Lack of breeze in [1,2] indicates that there is no pit in [2,2] and thus will move to [2,2] and then to [2,3]



Knowledge based wumpus Agent

Let that in [2,3] agent detects glitter, so it must grab the gold.



Conclusion:- In each case where the agent draws a conclusion from the available information, that conclusion is guaranteed to be correct if the available information is correct.

Propositional Logic/Boolean logic

- **What is logic?**
- Logic is the *study of the principles of correct reasoning.*


Aristotle

- It is used for representation of knowledge, reasoning, inference and proof generation.
- It provides syntax of representing facts and semantics for knowing the truth value of a sentence.
- Logic is used to separate truth from falsehood, reasonable from unreasonable beliefs.

Propositional Logic

- **What is a proposition?**
- It is a statement which can either true or false.
- Syntax of propositional logic- defines allowable sentences OR how sentences can be formed?
- **Syntax**
- Atomic sentence:- is individual syntactic element and can be represented with a single proposition symbol.
- Proposition Symbols- are upper case letter like P,Q, R
- Complex sentence:- are constructed from atomic ones, using logical connectives

Propositional Logic-connectives

- **Logical connectives.**
- There are 5 logical connectives
  - NOT , AND, OR, Implication, Bi-conditional

Negation (NOT)

¬ P- is called Negation of P,

A literal is either a atomic sentence (positive literal)

or a negated atomic sentence (negative literal)

Conjunction (AND)

A sentence whose main connective is $\wedge$ is called a Conjunction.

e.g (P $\wedge$ Q) , where P and Q are called conjuncts

Implication $\rightarrow$ OR $\Rightarrow$

A sentence whose connective is $\rightarrow$ is called an implication.

e.g (P $\rightarrow$ Q) , where P is called premise /antecedent

and Q is called as conclusion/consequence

P	Q	$P \rightarrow Q$
F	F	T
F	T	T
T	F	F
T	T	T

Propositional Logic

Bi-conditional ($\Leftrightarrow$) (if and only if)

$P \Leftrightarrow Q$ is same as (P $\Rightarrow$ Q) $\wedge$ (Q $\Rightarrow$ P)

P	Q	$P \Leftrightarrow Q$
F	F	T
F	T	F
T	F	F
T	T	T

Sentence	→	*AtomicSentence* | *ComplexSentence*
AtomicSentence	→	*True* | *False* | *P* | *Q* | *R* | ...
ComplexSentence	→	(*Sentence*) | [*Sentence*]
	|	¬ *Sentence*
	|	*Sentence* ∧ *Sentence*
	|	*Sentence* ∨ *Sentence*
	|	*Sentence* ⇒ *Sentence*
	|	*Sentence* ⇔ *Sentence*
OPERATOR PRECEDENCE	:	¬, ∧, ∨, ⇒, ⇔

Syntax of propositional logic- in Backus –Naur form

Propositional Logic-semantics

- Semantics- Defines rules for determining truth of a sentence wrt to a given model/world/environment

Let $P_{i,j}$ defines Pit in room [i,j] in wumpus word.

If knowledge base has three proposition symbols $P_{1,2}$, $P_{2,2}$ and $P_{3,1}$ then one possible model can be $m_1 = \{P_{1,2}=F, P_{2,2}=F, P_{3,1}=T\}$ there can be $2^3 = 8$ such possible models.

Truth tables are used to determine the truth of sentence. Find truth value of an atomic sentence and then compute truth value of complex sentences based on them.

Knowledge base in Propositional Logic

- KB consists of set of *sentences*. Thus, KB logically is *conjunction* of those sentences (s1 to sn).
- If we start with empty KB and do **TELL**(KB,s1),....,**TELL**(KB, sn), then we have

$$KB = S1 \wedge S2 \wedge ..... \wedge Sn$$

Simple Wumpus world Knowledge base in Propositional Logic

Let $P_{i,j}$ is true if there is pit in room [i,j]

and $B_{i,j}$ is true if there is breeze in room [i,j]

KB then contains following sentences , each one is labeled for convenience

$R1: \neg P_{1,1}$ (no pit in room [1,1])

R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ (pit is in room [1,2] or [2,1] iff breeze is in [1,1])

R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R 4: $\neg B_{1,1}$

$R5: B_{2,1}$

KB

KB then consists of sentences R1 to R5 or it can also be considered as a single sentence

$$R1 \wedge R2 \wedge \ldots \wedge R5$$

Inference in Propositional Logic

- **Entailment** :- (implies) (logically follows)

Logical entailment between sentences.

We say that sentence β *follows logically* from α

$$\alpha \models \beta$$

Formal definition:- $\alpha \models \beta$, iff, in every model in which α is true β is also true.

α can be an atomic senetence or complete KB

We can say that β can be inferred from α

Logical equivalences in Propositional Logic

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

Equivalence

Equivalence

We say that α and β are equivalent and written as

$\alpha \equiv \beta$

iff $\alpha \models \beta$ and $\beta \models \alpha$

Validity and satisfiability in Propositional Logic

- **Validity**

  A sentence is valid if it is true in all models

  e.g $P \vee \neg P$

  They are also called as tautologies

- **Satisfiability**

  A sentence α is satisfiable if it is true in some models.

  If α is true in model m , the we say that m satisfy α

Reasoning patterns in Propositional Logic

- **Standard patterns of inferences**
  (also called as inference rules)

Rule1. Modus Ponen (best known Rule)

If $\alpha \Rightarrow \beta$ and α are given, then

sentence β can be inferred.

e.g (WumpusAhead $\wedge$ WumpusAlive) $\Rightarrow$ shoot and (WumpusAhead $\wedge$ WumpusAlive) are given then shoot can be inferred

If an **implication** is given and its **antecedent** is true then its **consequence** must also be true

Reasoning patterns in Propositional Logic

Rule 2- And-Elimination

if $\alpha \wedge \beta$ are given then we can infer either α or β

i.e from conjunction any conjuct canbe inferred

From (WumpusAhead $\wedge$ WumpusAlive) we can infer WumpusAlive

If a **conjunction** is true then we can infer any one of the **conjunct** as true

Reasoning patterns in Propositional Logic

- All logical equivalences can be used as **inference rules**

If $\alpha \Leftrightarrow \beta$ is given then we can infer

$(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$, since

$(\alpha \Leftrightarrow \beta) \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ ⟸ Bi-conditional elimination rule

OR

If $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ is given then we can infer

$\alpha \Leftrightarrow \beta$

Not all rules can be used in both the ways

e.g if $\alpha \Rightarrow \beta$ and β is given we can not infer α

Use of inference rules in Propositional Logic in wumpus world

KB contains R1 to R5 and want to prove that $\neg P_{1,2}$

Using bi-conditional elimination rule on R2 we get R6

R6: $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Applying And-Elimination rule on R6 to obtain

R7: $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Using contrapositive rule if $\alpha \Rightarrow \beta$ then $(\neg\beta \Rightarrow \neg\alpha)$

$R8: \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$

Applying modus ponens with R8 the fact $\neg B_{1,1}$ to obtain

R9: $\neg(P_{1,2} \vee P_{2,1})$

Applying De-morgans law we obtain

R10: $\neg P_{1,2} \wedge P_{2,1}$

By applying And-Elimination we obtain

$\neg P_{1,2}$

Proof: sequence of application of inference rules is called Proof.

Proof generation and searching in Propositional Logic

- Generating **proof** is equivalent of **searching** through space and finding the solution. At each step set of all applicable ***inference rules*** can be generated and the most optimal rule is selected, continue till we get the final conclusion.
- **Proof procedures:-**
  - **Forward:-** start with a fact in KB and end in fact to be proved
  - **Backward:-** Opposite of forward
- Thus, searching algorithms can also be used to generate proofs.

A->B->C->D **D<- C<- B<-A**

Resolution in Propositional Logic

- ***Resolution*** is an *inference RULE*.
- It generates proof by contradiction
- At each step takes two clauses (parent clauses) as input which have complimentary literals and produces a new clause containing all the literals of two clauses except complementary literals. ($\neg P$ and P)
- Resolution process continues till a resolvant is either NULL or Contradiction

To prove that KB|=α (α logically follows or inferred from KB), Resolution shows that

KB|=$\neg\alpha$ is unsatisfiable by prooving a contradiction.

Resolution in Propositional Logic

Resolution converts (KB|=¬α) into Conjuctive Normal Form (CNF)

CNF is conjunction of disjuncts i.e $(P \vee Q) \wedge (R \vee S)$....

Convert the fact R2 from wumpus world into CNF (B: Breeze, P: Pit)

R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1.Eliminate bi-conditional using $\alpha \Leftrightarrow \beta$ can be replaced by $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2.Eliminate implication using $\alpha \Rightarrow \beta \equiv (\neg \alpha \vee \beta)$

$(\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})) \wedge (\neg (P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. CNF requires ¬ to appear only in literals, so we move ¬ inwards using equivalence rules

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4.Distributing $\vee$ over $\wedge$

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

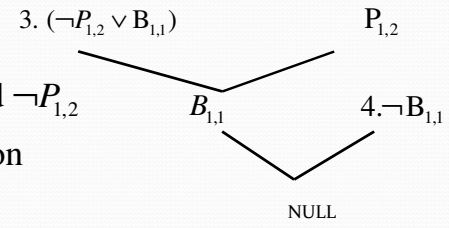
(Is Required CNF)

Example 1: Resolution in Propositional Logic

Consider the KB and $\neg P_{1,2}$

prove using resolution

1. $(\neg P_{2,1} \vee B_{1,1})$
2. $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$
3. $(\neg P_{1,2} \vee B_{1,1})$
4. $\neg B_{1,1}$



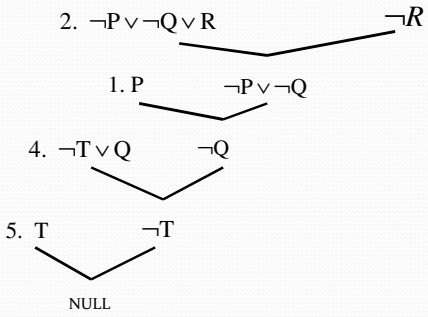
Thus, $\neg P_{1,2}$ is proved

Example 2: Resolution in Propositional Logic

Using KB prove that R is True

Given axioms	Converted to a clause form	
P	P	(1)
$(P \wedge Q) \Rightarrow R$	$\neg P \vee \neg Q \vee R$	(2)
$(S \vee T) \Rightarrow Q$	$\neg S \vee Q$	(3)
	$\neg T \vee Q$	(4)
T	T	(5)

Reference:- Rich and Knight



Since resolvant is a null a contradiction is obtained and thus R is proven true

Forward and Backward chaining in Propositional Logic

- Real world KBs often contain only clauses of restricted kind called Horn Clauses.
- A Horn clause is a disjunction of literals of which *at most one is positive*.
- $(\neg P \vee Q \vee \neg R)$ is Horn clause but $(P \vee Q \vee \neg R)$is not
- Using Horn clauses Inference can be drawn using forward chaining and backward chaining in a very efficient manner.

Forward and Backward chaining in Propositional Logic

- **Forward chaining.**
  - Determines if a single propositional symbol q can be entailed by KB of Horn clauses. It begins with the known facts (positive literals). If all premises of an implication are known then its conclusion is added to set of facts. This process continues till q is added to KB or until no inference can be made.
- **Backward chaining**
  - It works backward from q. If q is known to be true no work is needed. Otherwise algorithm finds those implications in KB that concludes q. If all premises of implications can be proved true (by backward chaining) then q is true.
  - It is a form of goal-directed reasoning.
- Forward reasoning is used for generation of facts and backward is for proof.

A->B->C->D **D<- C<- B<-A**

NOTE

- Read from slides of Predicate logic