LOGICAL AGENTS

Lecture-02

Conduct by
Khawja Imran Masud
Lecturer, Department of CSE
Dhaka University of Engineering & Technology, Gazipur
Email: kimasud.cse@duet.ac.bd , eimran.cse@gmail.com

Course related links!

1

- **E-Learning Platform:**

https://elp.duetbd.org

2

- **Course Materials:**

Lecture Slides and Ref. Book & Videos: https://drive.google.com/drive/folders/1pw--AcsR5oNud-jnwu4T4m1d2f89Ro1a?usp=sharing

3

- **Zoom**

Online Class: https://bdren.zoom.us/j/69976591053?pwd=eDFyK3RlTzNLekoxQmt6Z1BDSkg2Zz09

4

- **Resource-1:**

https://dmi.unibas.ch/en/academics/computer-science/courses-in-spring-semester-2020/lecture-foundations-of-artificial-intelligence

5

- **Resource-2:**

https://cms.sic.saarland/ai_20/materials

Outline

01

KNOWLEDGE BASED AGENTS

About logical agents, Architecture of KB agents, A generic knowledge-based agent

Logical Agents

- Agents with some representation of **complex knowledge** about the world (It's environment) & uses **Inference** to derive new information from the knowledge is called logical agents. Logical agents is knowledge based agents.
- **Knowledge based:** It's a set of sentences in a formal language representing facts about the world/environment.
- The idea is that an agent can **represent knowledge** of its world, its **goals** and the **current situation** by **sentences in logic** and decide what to do by **inferring** that a **certain action** or **course of action** is appropriate to **achieve its goals**.

Knowledge based agents

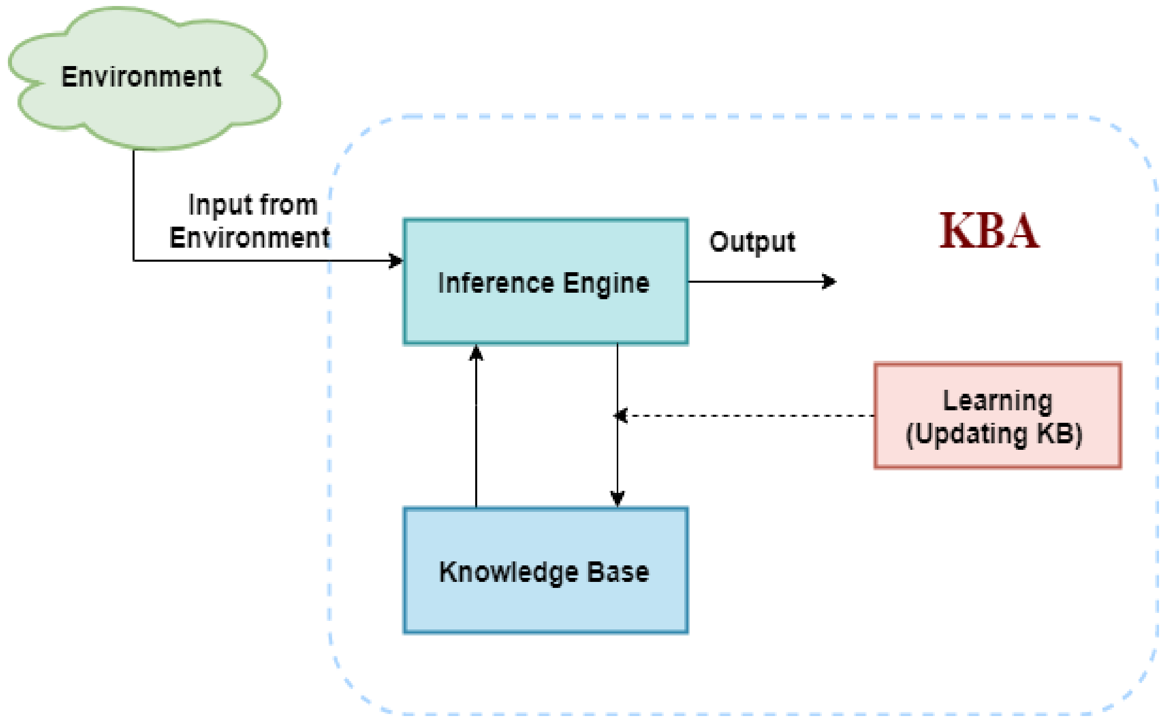
- Intelligent agents need knowledge about the world to choose good actions/decisions.
- Knowledge = *{sentences}* in a knowledge representation language (formal language).
- A **sentence** is an assertion about the world.
- Knowledge-based agents are composed of two main parts:
  - Knowledge-base [**domain-specific** content]
  - Inference system [**domain-independent** algorithms]

Knowledge based agents

- A knowledge-based agent must able to do the following:
  - An agent should be able to represent states, actions, etc.
  - An agent Should be able to incorporate new percepts
  - An agent can update the internal representation of the world
  - An agent can deduce the internal representation of the world
  - An agent can deduce appropriate actions.

The architecture of knowledge-based agent:

- **The knowledge-based agent (KBA)** take input from the environment by perceiving the environment.
- The input is taken by the **inference engine** of the agent and which also communicate with KB to decide as per the knowledge store in KB.
- The **learning element** of KBA regularly updates the KB by learning new knowledge.
- **Knowledge-base** is a central component of a knowledge-based agent, it is also known as KB. The Knowledge-base of KBA stores fact about the world.



Operations Performed by KBA

Following are three operations which are performed by KBA in order to show the intelligent behavior:

- **TELL:** This operation tells the knowledge base what it perceives from the environment.
- **ASK:** This operation asks the knowledge base what action it should perform.
- **Action:** It performs the selected action.

A generic knowledge-based agent

Each time when the function is called, it performs its three operations:

Firstly, it TELLs the KB what it perceives.
Secondly, it asks KB what action it should take
Thirdly, agent program TELLs the KB that which action was chosen.

```
function KB-AGENT(percept): return an action
static: 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
```

MAKE-PERCEPT-SENTENCE generates a sentence as setting that the agent perceived the given percept at the given time.

MAKE-ACTION-QUERY generates a sentence to ask which action should be done at the current time.

MAKE-ACTION-SENTENCE generates a sentence which asserts that the chosen action was executed.

02

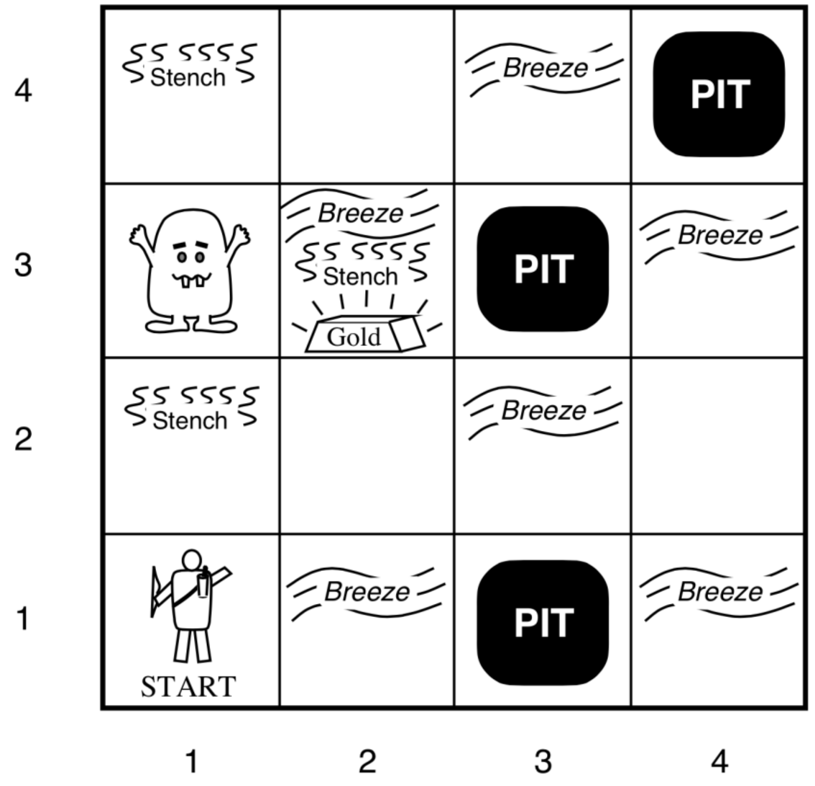
THE WUMPUS WORLD

About Wumpus World, PEAS and Properties of Wumpus World, Exploring Wumpus World, Knowledge-base for Wumpus world

The Wumpus World!

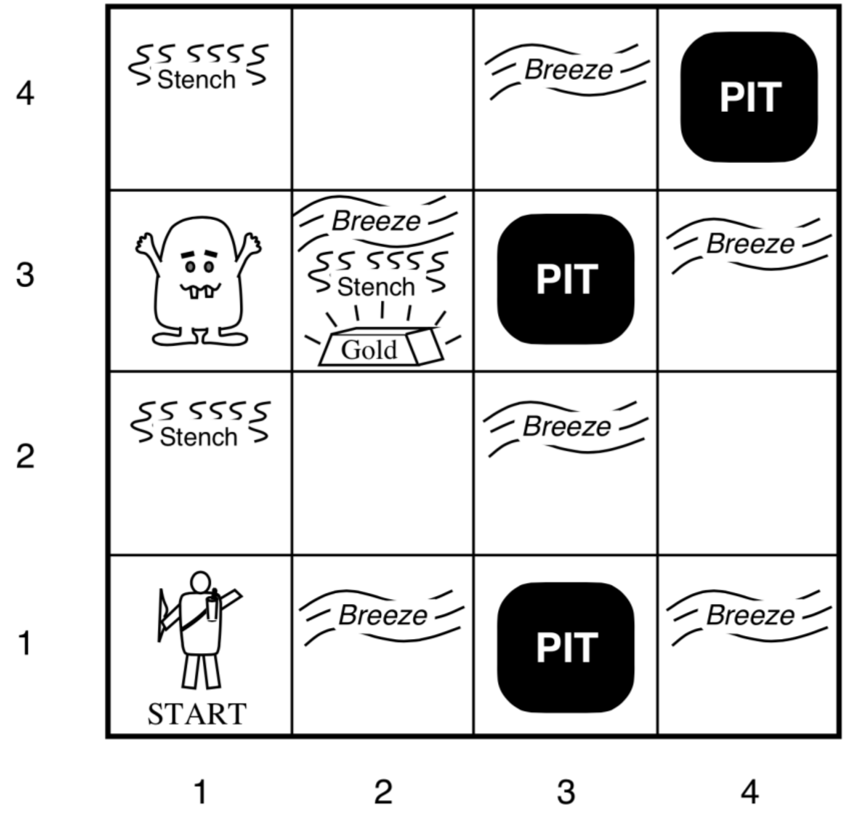
The Wumpus world is a simple world example to illustrate the worth of a knowledge-based agent and to represent knowledge representation. It was inspired by a video game Hunt the Wumpus by Gregory Yob in 1973.

- Wumpus world is a cave which has 4 X 4 grid of rooms. So there are total 16 rooms which are connected with each other.
- Squares adjacent to Wumpus are smelly and squares adjacent to pit are breezy.
- Glitter if and only if gold is in the same square.
- Shooting kills Wumpus if you are facing it, and emits a horrible scream when it is killed that can be heard anywhere.
- Shooting uses up the only arrow.
- Grabbing picks up gold if in same square.
- Releasing drops the gold in same square.



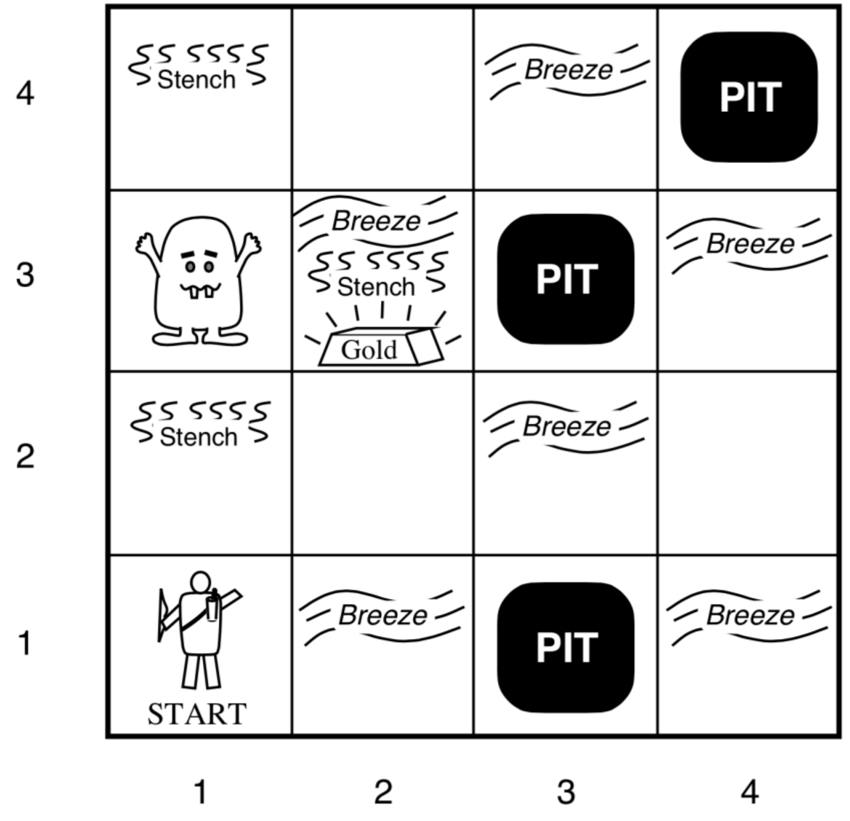
PEAS for the Wumpus World

- **Performance measure:** gold +1000, death (eaten or falling in a pit) -1000, -1 per action taken, -10 for using the arrow. The games ends either when the agent dies or comes out of the cave.
- **Environment:** 4 X 4 grid of rooms. Agent starts in square [1,1] facing to the right. Locations of the gold, and Wumpus are chosen randomly with a uniform distribution from all squares except [1,1].
- **Actuators**: Left turn, Right turn, Forward, Grab, Release, Shoot
- **Sensors**: Breeze, Glitter, Smell, Bump, Scream

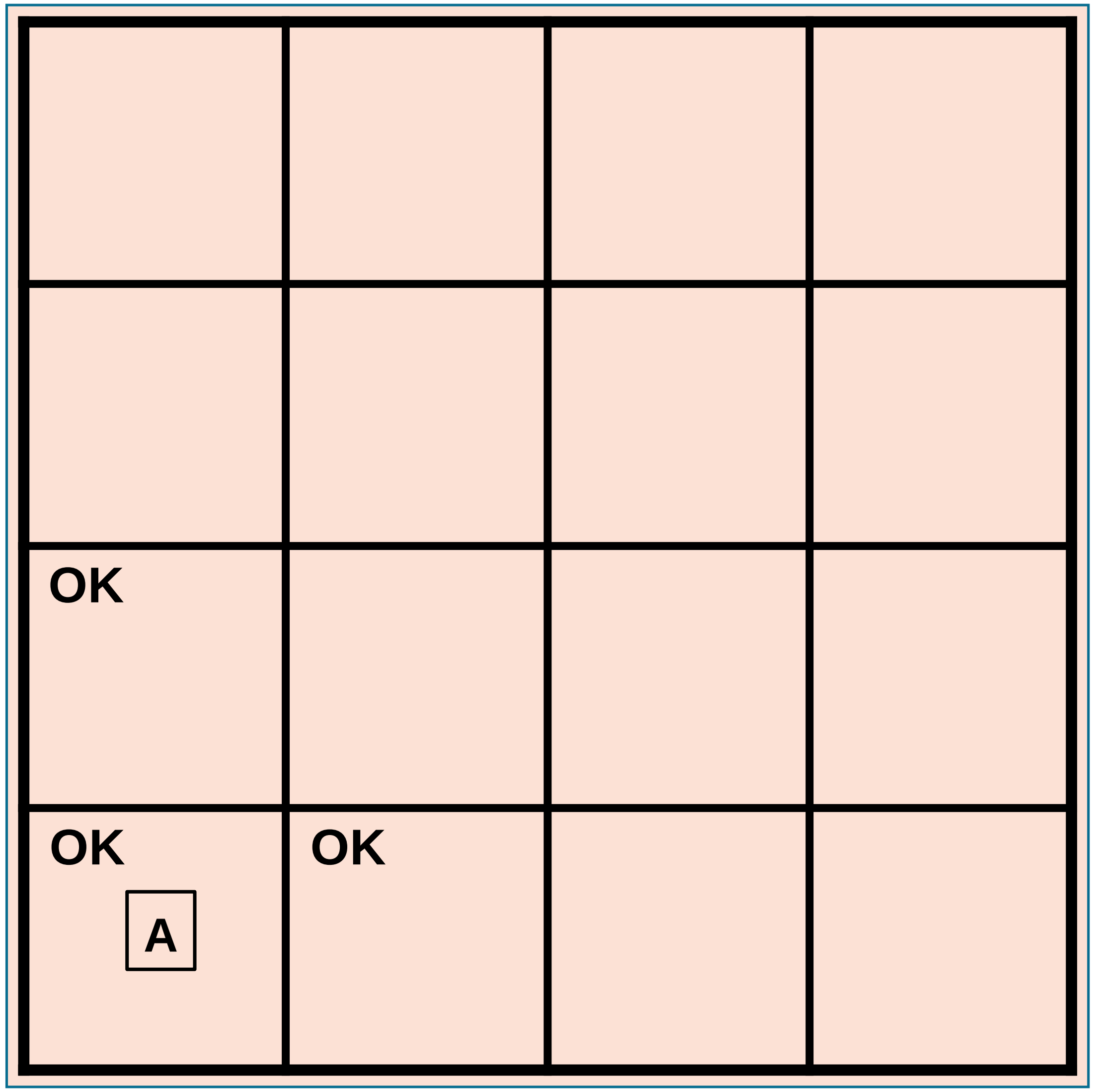


Properties of the Wumpus World

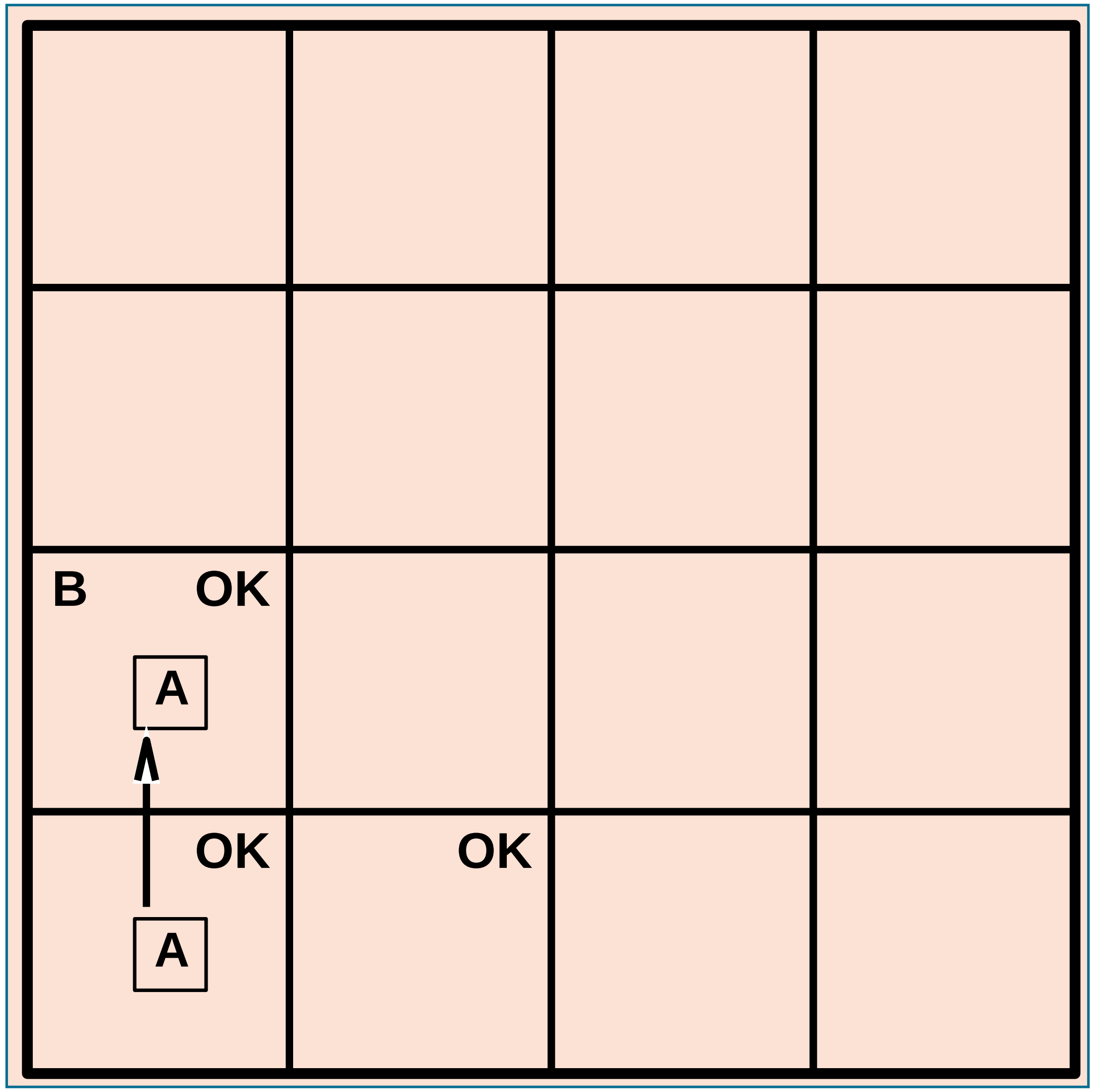
- **Partially observable**: The Wumpus world is partially observable because the agent can only perceive the close environment such as an adjacent room.
- **Deterministic**: It is deterministic, as the result and outcome of the world are already known.
- **Sequential**: The order is important, so it is sequential.
- **Static**: It is static as Wumpus and Pits are not moving.
- **Discrete**: The environment is discrete because there are a finite number of percepts and actions that can be performed within it.
- **Single agent**: The environment is a single agent as we have one agent only and Wumpus is not considered as an agent.



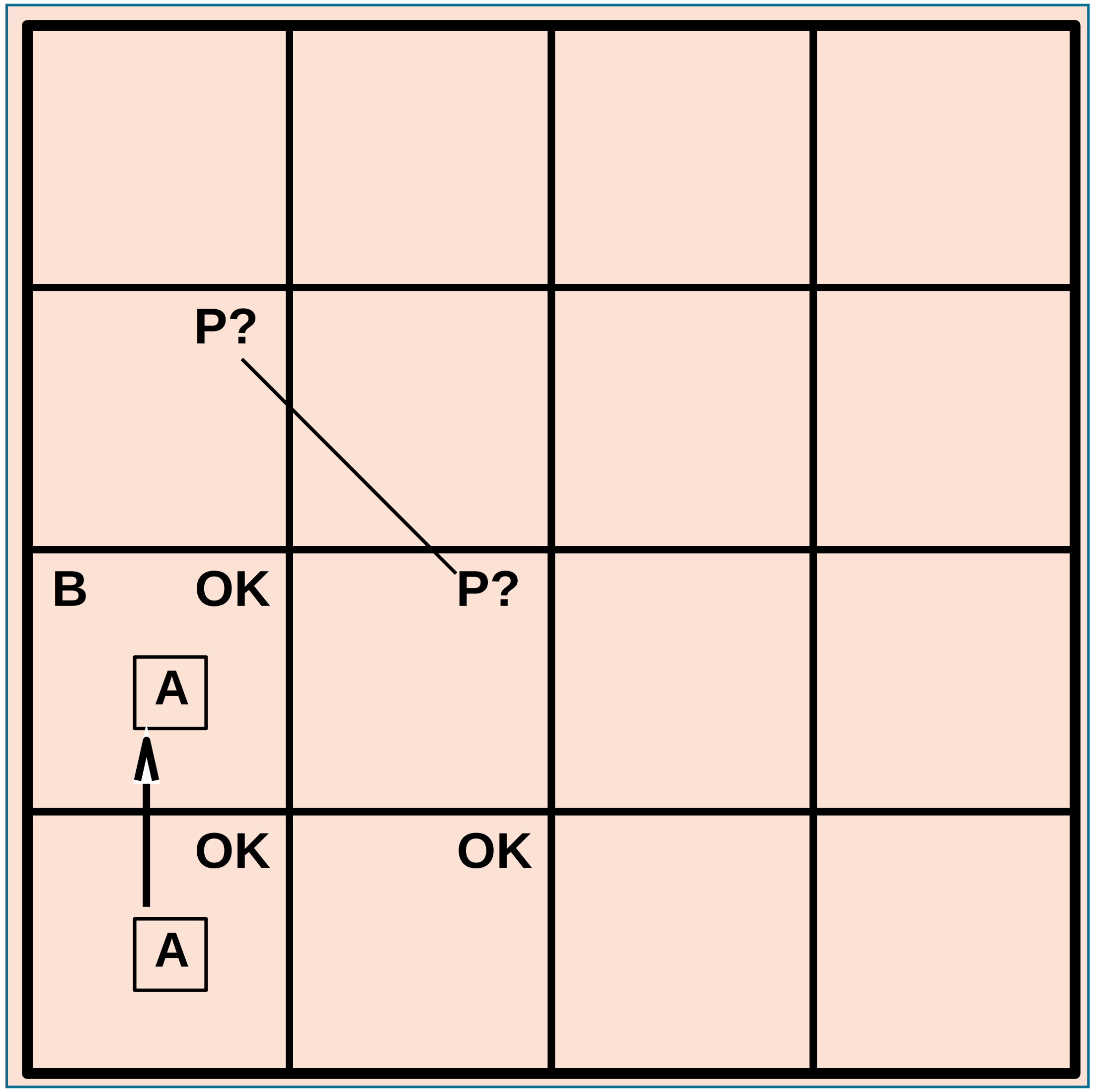
Exploring Wumpus World



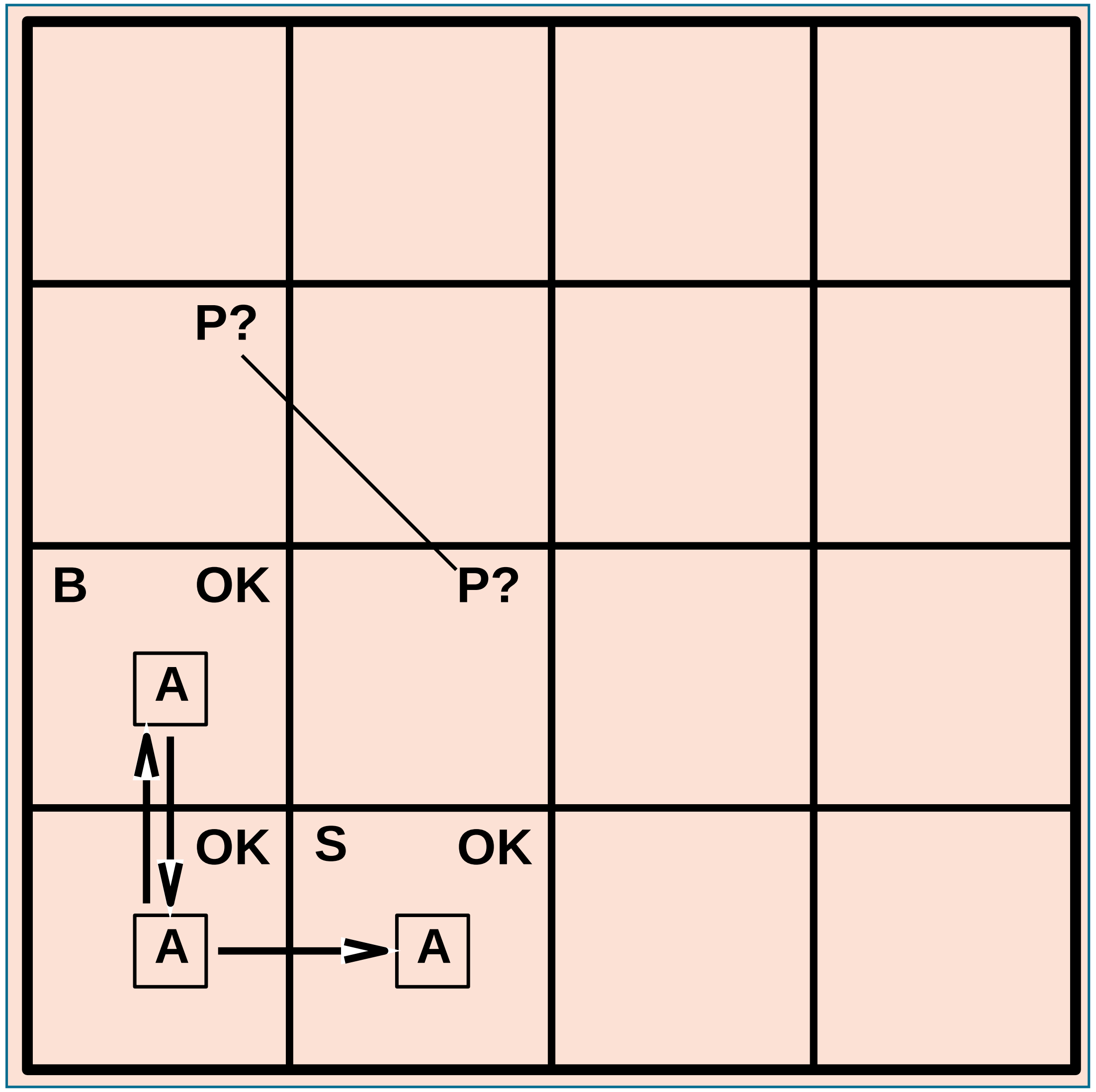
Exploring Wumpus World



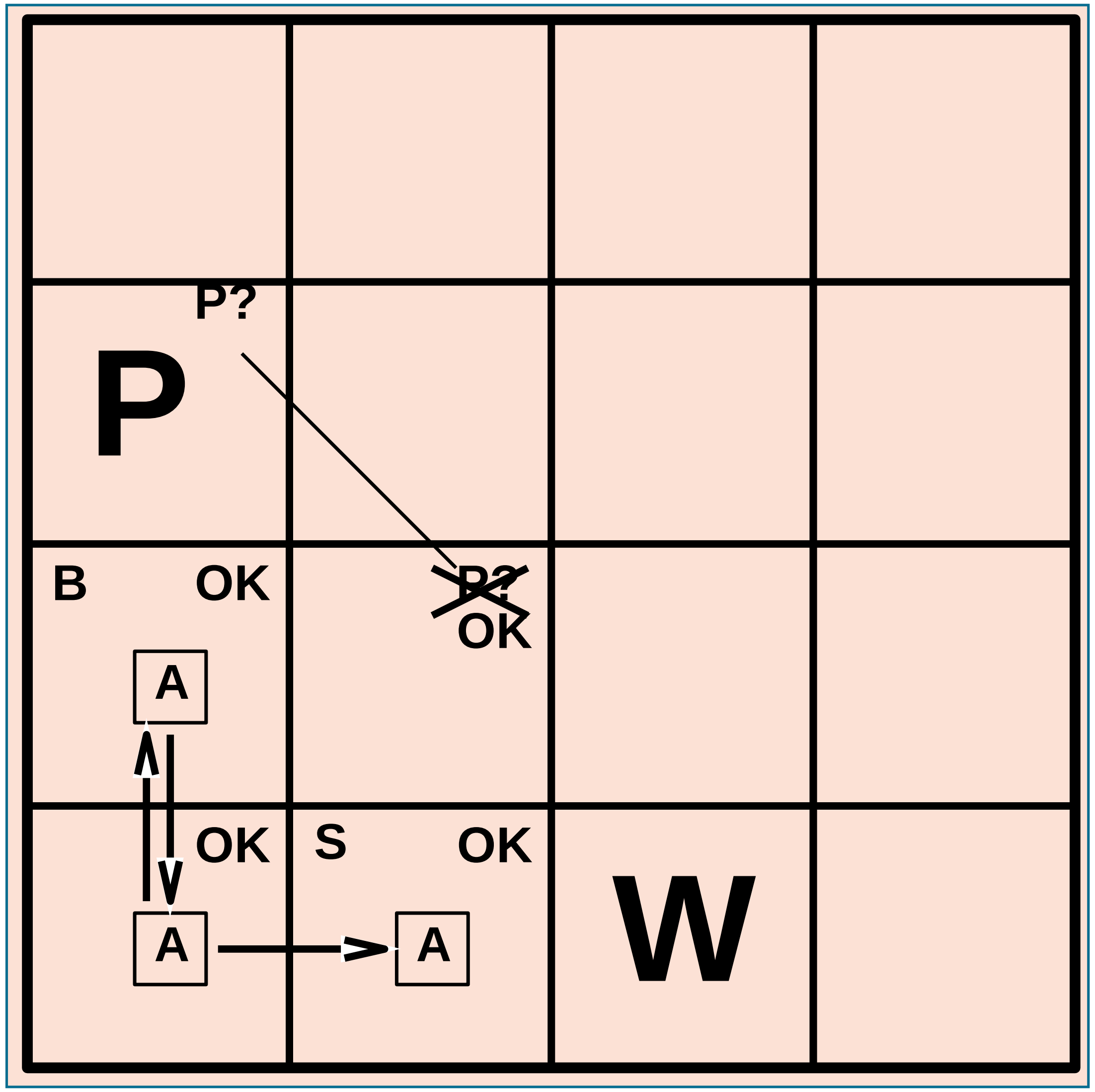
Exploring Wumpus World



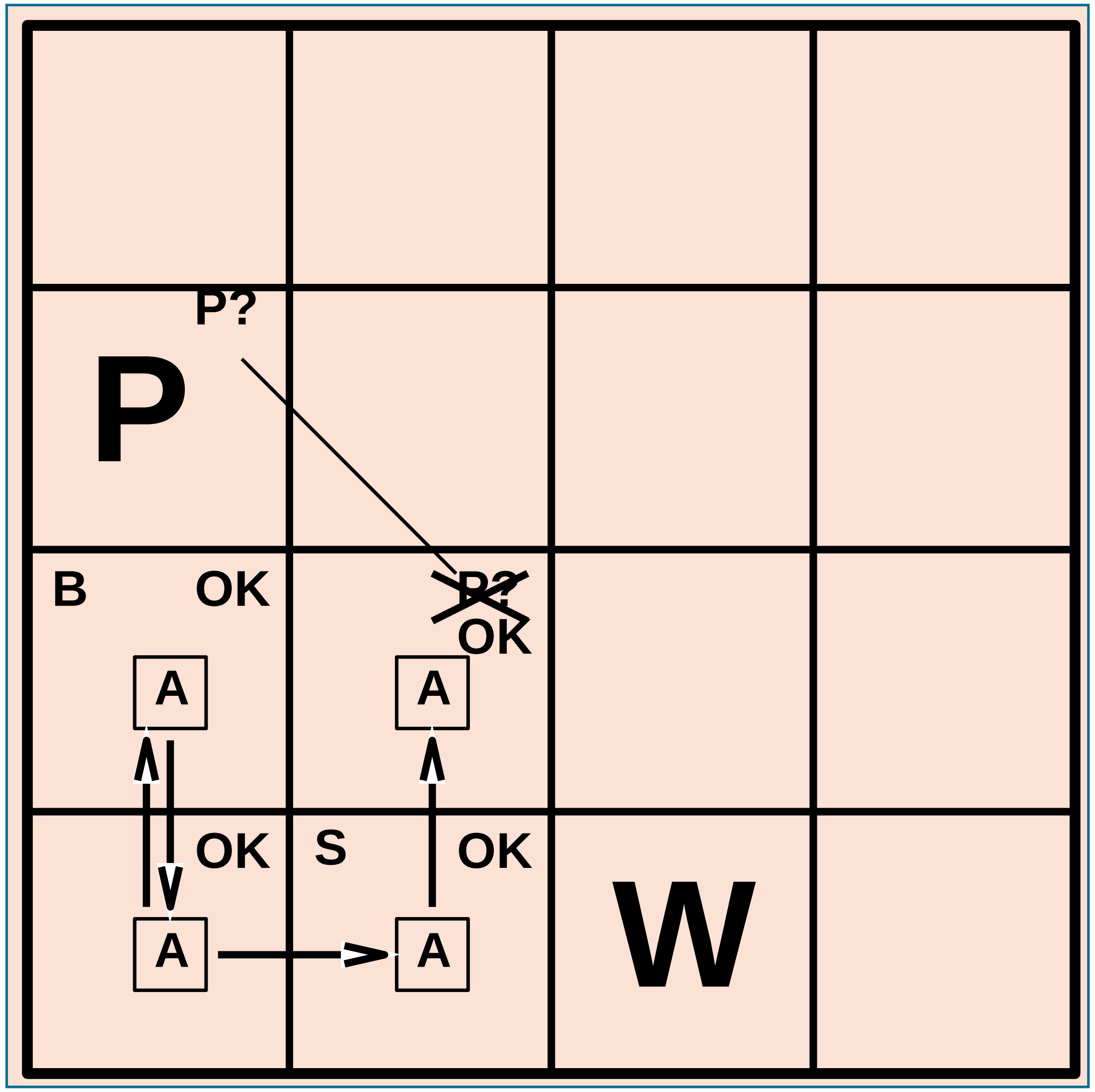
Exploring Wumpus World



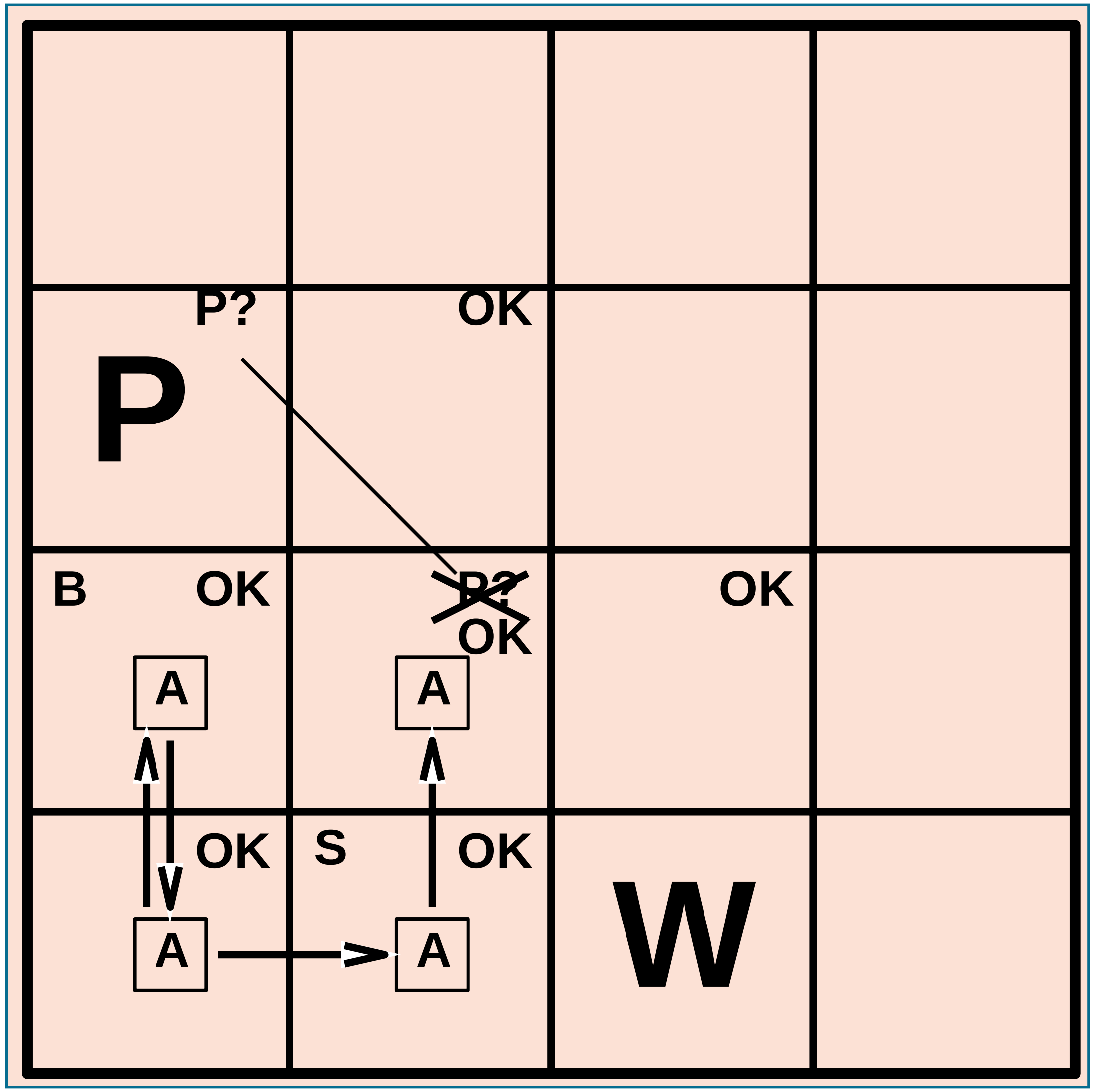
Exploring Wumpus World



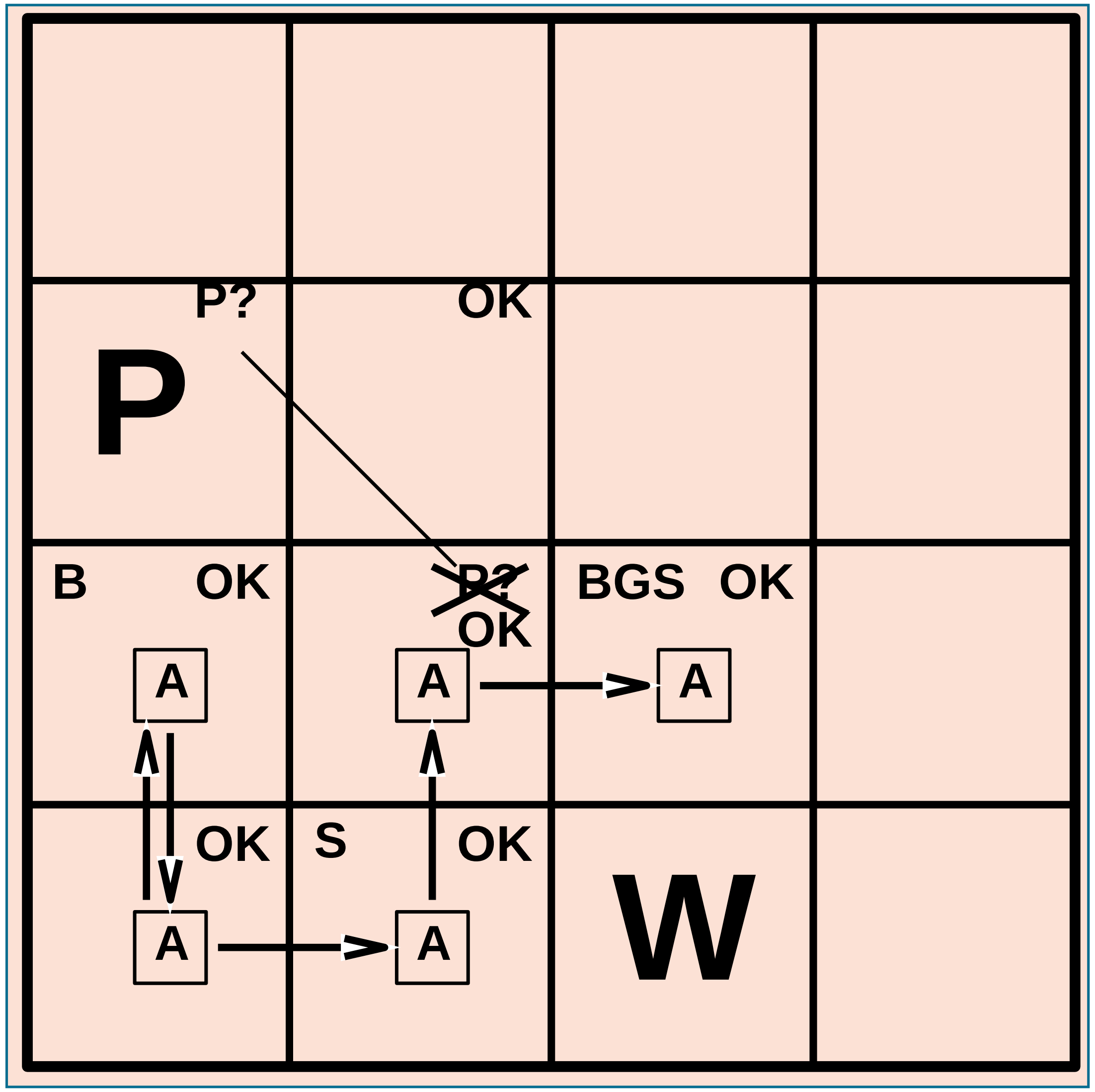
Exploring Wumpus World



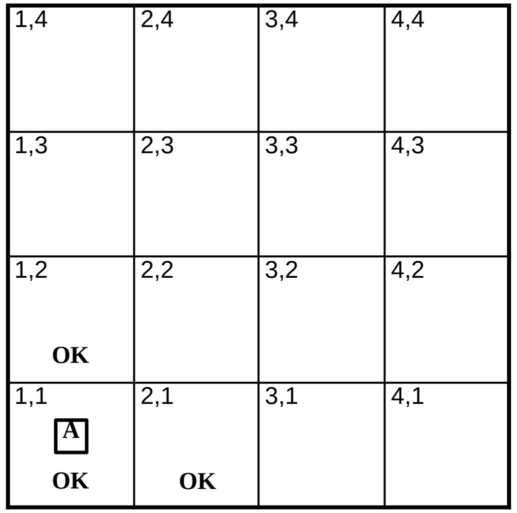
Exploring Wumpus World



Exploring Wumpus World

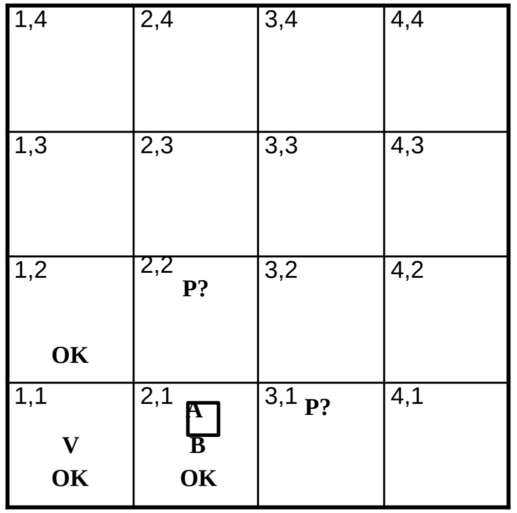


Exploring Wumpus World

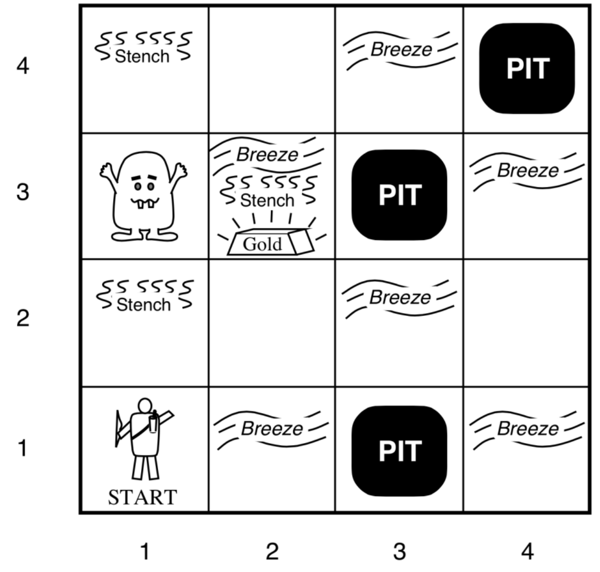


(a)

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



(b)



(a):

Initially, the agent is in the first room or on the square [1,1]. What are the safe moves from (1,1)? Move to (1,2), (2,1), or stay in (1,1)

Suppose agent moves to room [2, 1]

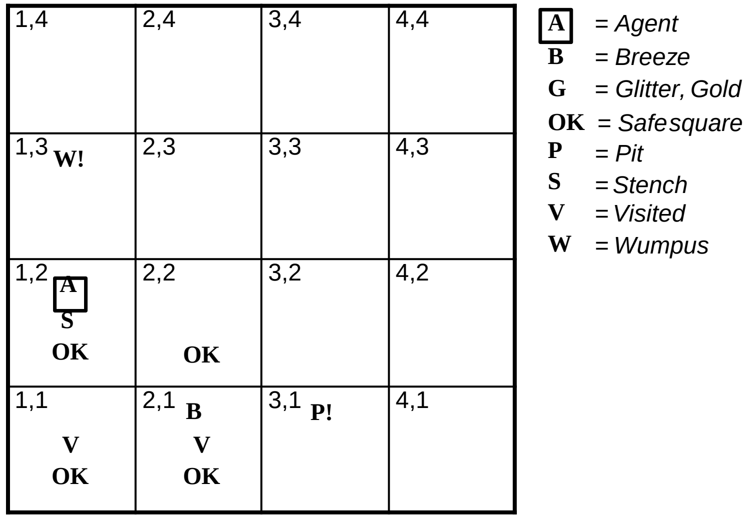
(b):

What are the safe moves from (2,1)?

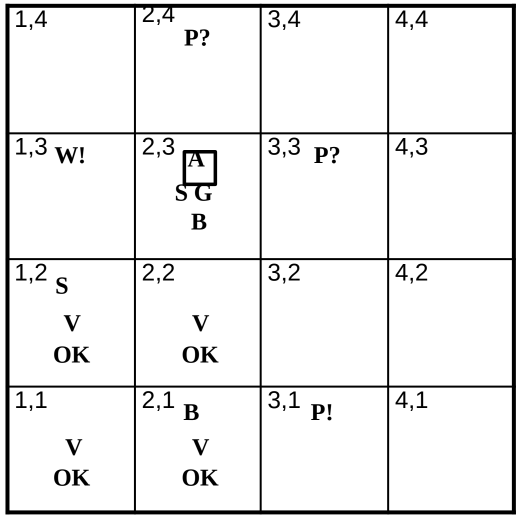
B in (2,1) ⇒ P in (2,2) or (3,1) or (1,1)

Now agent will stop and think and will not make any harmful move. The agent will go back to the [1, 1] room

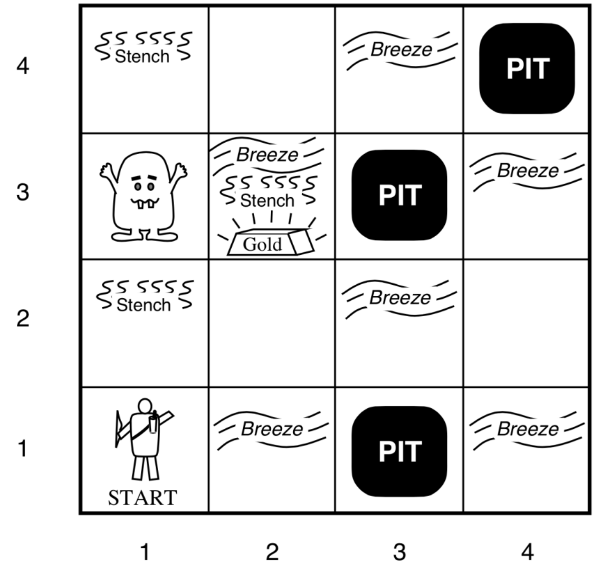
Exploring Wumpus World



(c)



(d)



(c):

Now agent will move to the room [1,2] which is OK.

- S in (1,2) ⇒ W in (1,1) or (2,2) or (1,3)
- Survived in (1,1) and no S in (2,1) ⇒ W in (1,3)
- No B in (1,2) ⇒ P in (3,1)

So (2,2) is safe, and we will mark it OK, and the agent moves further in [2,2].

(d):

At room [2,2], here no stench and no breezes present so let's suppose agent decides to move to [2,3].

At room [2,3] agent perceives glitter, so it should grab the gold and climb out of the cave.

03

PROPOSITIONAL LOGIC

Logical Representation, Propositional Logic, Logical Connectives, Rules of Inference, Knowledge-base for Wumpus World

Logical Representation

Logical representation is a **language** with some concrete rules which deals with **propositions** and has **no ambiguity** in representation. Logical representation means drawing a conclusion based on various conditions. It consists of precisely defined **syntax** and **semantics** which supports the sound **inference**. Each sentence can be translated into logics using **syntax** and **semantics**.

Logical representation can be categorized into mainly two logics:

- Propositional Logics
- Predicate logics

Logical Representation

- **Syntax**: Syntaxes are the rules which decide how we can construct legal sentences in the logic. It determines which symbol we can use in knowledge representation.
- **Semantics**: Semantics are the rules by which we can interpret the sentence in the logic. Semantic also involves assigning a meaning to each sentence.
- **Inference**: a procedure to derive a new sentence from other ones.
- **Knowledge based:** It's a set of sentences in a formal language representing facts about the world/environment. It's denoted with **KB**.
- **Logical entailment**: It's a relationship between sentences. It means that a sentence follows logically from other sentences. For example, α is a sentence which logically infer from KB.

$$KB \models \alpha$$

Propositional 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. For example:
  - The Sun rises from West (False proposition)
  - 5 is a prime number. (True proposition)
- Propositions can be either true or false, but it cannot be both.
- Propositional logic is also called Boolean logic as it works on 0 and 1.
- In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.

Propositional Logic

There are two types of Propositions:

1. Atomic Propositions
2. 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. For example:

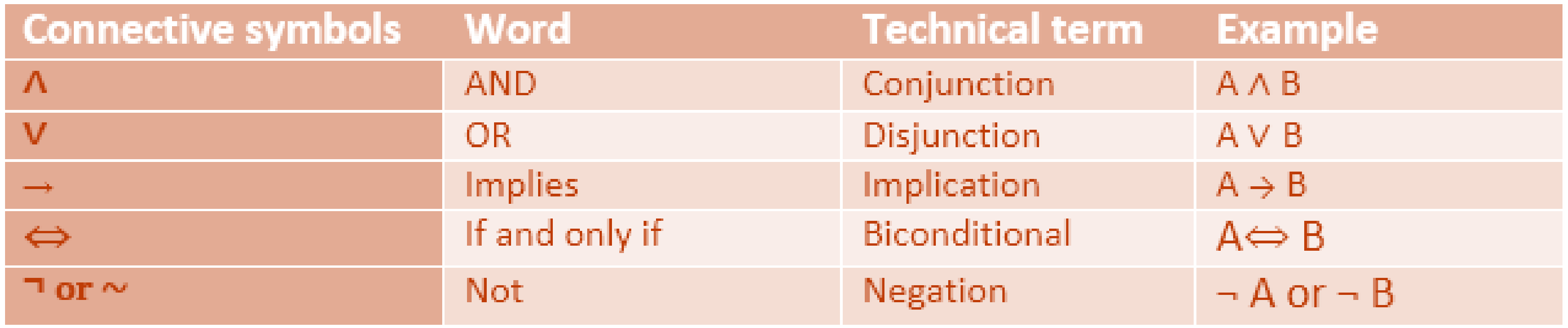
  a) 2+2 is 4, it is an atomic proposition as it is a **true** fact.
  b) "The Sun is cold" is also a proposition as it is a **false** fact.

- **Compound proposition**: Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives. For example:

  a) "It is raining today, and street is wet."
  b) "Ankit is a doctor, and his clinic is in Mumbai."

Logical Connectives:

Logical connectives are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives. There are mainly five connectives, which are given as follows:



Connective symbols	Word	Technical term	Example
∧	AND	Conjunction	A ∧ B
∨	OR	Disjunction	A ∨ B
→	Implies	Implication	A → B
⇔	If and only if	Biconditional	A⇔ B
¬ or ~	Not	Negation	¬ A or ¬ B

Negation

A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.

p	$\neg p$
T	F
F	T

Conjunction:

A sentence which has ∧ connective such as, P ∧ Q is called a conjunction.

Example: Rohan is intelligent and hardworking.

Here, P= Rohan is intelligent and Q= Rohan is hardworking.

So we can write it as **P∧ Q.**

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Disjunction:

A sentence which has V connective, such as P V Q. is called disjunction, where P and Q are the propositions.

Example: "Ritika is a doctor or Engineer",

Here, P= Ritika is Doctor and Q= Ritika is Doctor

So we can write it as **P V Q.**

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Implication:

A sentence such as P → Q, is called an implication. Implications are also known as if-then rules.

Example: 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

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Bi-conditional:

A sentence such as P⇔ Q is a Bi-Conditional sentence,

Example: If I am breathing, then I am alive

Let, P= I am breathing and Q= I am alive,

So, it can be represented as **P ⇔ Q.**

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Precedence of connectives

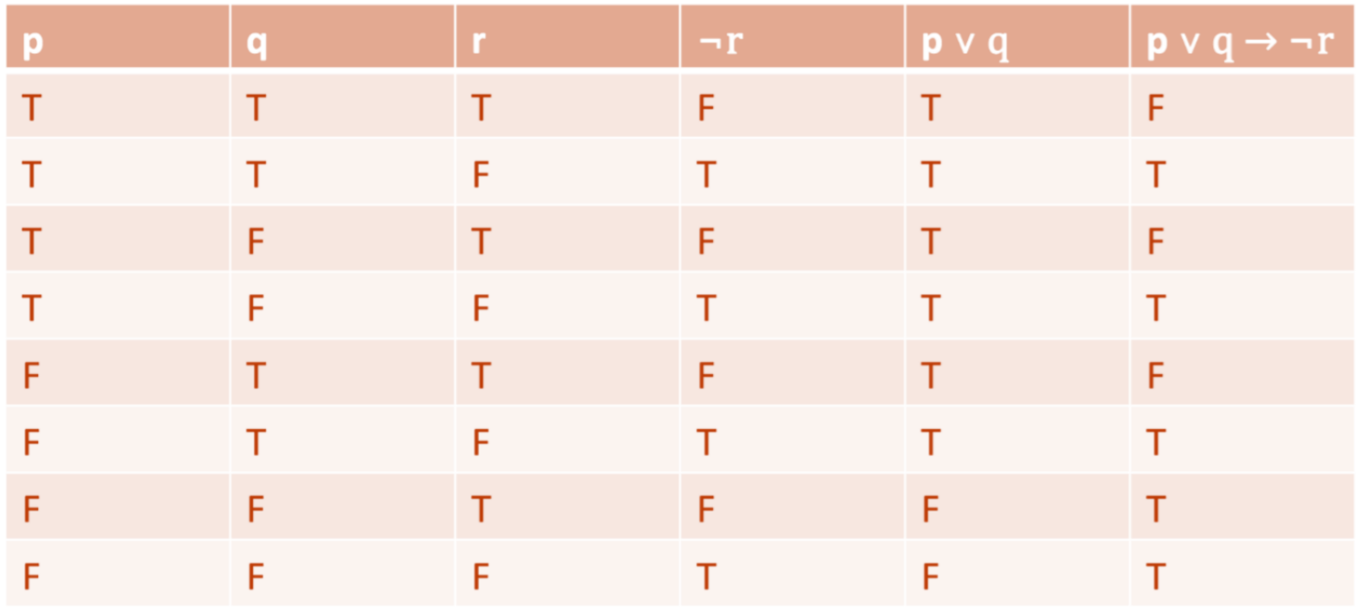
- Just like arithmetic operators, there is a precedence order for propositional connectors or logical operators. This order should be followed while evaluating a propositional problem. Following is the list of the precedence order for operators:

Precedence	**Operators**
First Precedence	Parenthesis
Second Precedence	Negation
Third Precedence	Conjunction(AND)
Fourth Precedence	Disjunction(OR)
Fifth Precedence	Implication
Six Precedence	Biconditional

Note: *For better understanding use parenthesis to make sure of the correct interpretations. Such as* $\neg R \vee Q$*, It can be interpreted as* $(\neg R) \vee Q$.

Building propositions

- We can build a proposition composing three propositions P, Q, and R. This truth table is made-up of 8n Tuples as we have taken three proposition symbols.



p	q	r	$\neg r$	$p \vee q$	$p \vee q \rightarrow \neg r$
T	T	T	F	T	F
T	T	F	T	T	T
T	F	T	F	T	F
T	F	F	T	T	T
F	T	T	F	T	F
F	T	F	T	T	T
F	F	T	F	F	T
F	F	F	T	F	T

Logical equivalence

- Two propositions are said to be logically equivalent if and only if the columns in the truth table are identical to each other.
- Let's take two propositions A and B, so for logical equivalence, we can write it as A⇔B. In below truth table we can see that column for ¬A∨ B and A→B, are identical hence A is Equivalent to B

p	q	$\neg p$	$\neg p \vee q$	$p \rightarrow q$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Properties of Operators

Commutativity:

- $P \land Q = Q \land P$, or
- $P \lor Q = Q \lor P$.

Associativity:

- $(P \land Q) \land R = P \land (Q \land R)$,
- $(P \lor Q) \lor R = P \lor (Q \lor R)$

Identity element:

- $P \land True = P$,
- $P \lor True = True$.

Distributive:

- $P \land (Q \lor R) = (P \land Q) \lor (P \land R)$.
- $P \lor (Q \land R) = (P \lor Q) \land (P \lor R)$.

DE Morgan's Law:

- $\neg (P \land Q) = (\neg P) \lor (\neg Q)$
- $\neg (P \lor Q) = (\neg P) \land (\neg Q)$.

Double-negation elimination:

- $\neg (\neg P) = P$.

Tautology

- Tautology is a proposition which is always true. For example: $(p \longrightarrow q) \leftrightarrow (\sim q \longrightarrow \sim p)$ is a tautology.

p	q	$p \rightarrow q$	$\sim q$	$\sim p$	$\sim q \longrightarrow \sim p$	$(p \rightarrow q) \leftrightarrow (\sim q \longrightarrow \sim p)$
T	T	T	F	F	T	T
T	F	F	T	F	F	T
F	T	T	F	T	T	T
F	F	T	T	T	T	T

Contradiction

- Contradiction is a proposition which is always false. For example: p Λ~p is a contradiction.

p	~p	p Λ~p
T	F	F
F	T	F

Contingency

- Contingency is a proposition which is neither a tautology or a contradiction. For example: $(p \rightarrow q) \longrightarrow (p \land q)$ is a contingency.

p	q	$p \rightarrow q$	$p \land q$	$(p \rightarrow q) \longrightarrow (p \land q)$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	F
F	F	T	F	F

Variations in Conditional Statement

A conditional statement consists of two parts, a **hypothesis** in the “if” clause and a **conclusion** in the “then” clause. For instance, “If it rains, then they cancel school.”

"It rains" is the hypothesis.

"They cancel school" is the conclusion.

- **Contrapositive:** The proposition $\sim q \rightarrow \sim p$ is called contrapositive of $p \rightarrow q$. The contrapositive of "If it rains, then they cancel school" is "*If they do not cancel school, then it does not rain*."
- **Converse**: The proposition $q \rightarrow p$ is called the converse of $p \rightarrow q$. The converse of "If it rains, then they cancel school" is "*If they cancel school, then it rains.*"
- **Inverse**: The proposition $\sim p \rightarrow \sim q$ is called the inverse of $p \rightarrow q$. The inverse of “If it rains, then they cancel school” is *“If it does not rain, then they do not cancel school.”*

Rules of Inference

In artificial intelligence, we need intelligent computers which can create new logic from old logic or by evidence, so generating the **conclusions** from **evidence** and **facts** is termed as **Inference**. Types of Inference rules:

1. Modus Ponens: The Modus Ponens rule is one of the most important rules of inference, and it states that if P and P → Q is true, then we can infer that Q will be true. It can be represented as:

Notation for Modus ponens: $\frac{P \to Q, \quad P}{\therefore Q}$

Example:

Statement-1: "If I am sleepy then I go to bed" ==> P→ Q

Statement-2: "I am sleepy" ==> P

Conclusion: "I go to bed." ==> Q

Hence, we can say that, if P→ Q is true and P is true then Q will be true.

Proof by Truth table:

P	Q	$P \to Q$
0	0	0
0	1	1
1	0	0
1	1	1 ←

Rules of Inference

2. Modus Tollens: The Modus Tollens rule state that if P→ Q is true and ¬ Q is true, then ¬ P will also true. It can be represented as:

Notation for Modus Tollens: $\dfrac{P \to Q, \;\; \sim Q}{\sim P}$

Example:

Statement-1: "If I am sleepy then I go to bed" ==> P→ Q

Statement-2: "I do not go to the bed."==> ~Q

Statement-3: Which infers that "I am not sleepy" => ~P

Proof by Truth table:

P	Q	$\sim P$	$\sim Q$	$P \to Q$
0	0	1	1	1 ←
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1

Rules of Inference

3. Hypothetical Syllogism: The Hypothetical Syllogism rule state that if P→R is true whenever P→Q is true, and Q→R is true. It can be represented as the following notation:

$$\begin{array}{l} P\to Q \\ Q\to R \\ \underline{P\to R} \end{array}$$

Example:

Statement-1: If you have my home key then you can unlock my home. P→Q

Statement-2: If you can unlock my home then you can take my money. Q→R

Conclusion: If you have my home key then you can take my money. P→R

Proof by truth table:

P	Q	R	$P \to Q$	$Q \to R$	$P \to R$	
0	0	0	1	1	1	←
0	0	1	1	1	1	←
0	1	0	1	0	1	
0	1	1	1	1	1	←
1	0	0	0	1	1	
1	0	1	0	1	1	
1	1	0	1	0	0	
1	1	1	1	1	1	←

Rules of Inference

4. Disjunctive Syllogism or Unit Resolution: The Disjunctive syllogism rule state that if PVQ is true, and ¬P is true, then Q will be true. It can be represented as:

Notation of Disjunctive syllogism: $\frac{P \vee Q, \quad \neg P}{Q}$

Example:

Statement-1: Today is Sunday or Monday. ==>PVQ

Statement-2: Today is not Sunday. ==> ¬P

Conclusion: Today is Monday. ==> Q

Proof by truth-table:

P	Q	$\neg P$	$P \vee Q$
0	0	1	0
0	1	1	1 ←
1	0	0	1
1	1	0	1

Rules of Inference

5. Addition: The Addition rule is one the common inference rule, and it states that If P is true, then PVQ will be true.

Notation of Addition: $\frac{P}{P \vee Q}$

Example:

Statement: I have a vanilla ice-cream. ==> P

Statement-2: I have Chocolate ice-cream.

Conclusion: I have vanilla or chocolate ice-cream. ==> (PVQ)

Proof by Truth-Table:

P	Q	$P \vee Q$
0	0	0
1	0	1 $\leftarrow$
0	1	1
1	1	1 $\leftarrow$

Rules of Inference

6. Simplification or And Elimination Rule: The simplification rule state that if P∧ Q is true, then Q or P will also be true. It can be represented as:

Notation of Simplification rule: $\frac{P \wedge Q}{Q}$ Or $\frac{P \wedge Q}{P}$

Proof by Truth-Table:

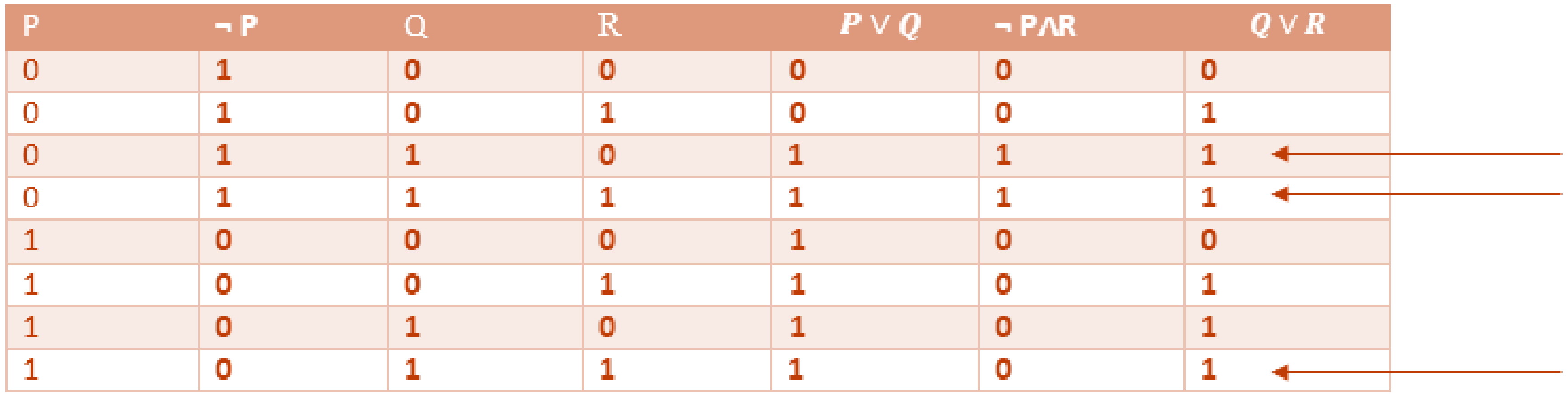
P	Q	$P \wedge Q$
0	0	0
1	0	0
0	1	0
1	1	1 $\longleftarrow$

Rules of Inference

7. Resolution: The Resolution rule state that if **PVQ** and **¬ PɅR** is true, then **QVR** will also be true. It can be represented as

Notation of Resolution $\dfrac{P\vee Q,\ \ \neg\, P\wedge R}{Q\vee R}$

Proof by Truth-Table:



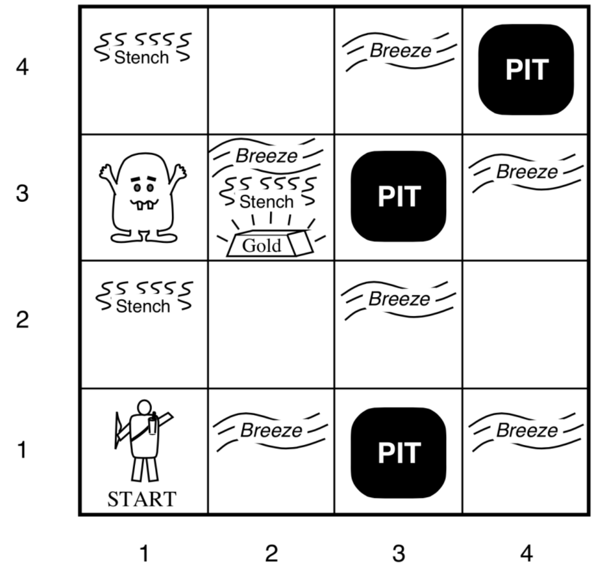
P	¬ P	Q	R	$P \vee Q$	¬ PɅR	$Q \vee R$
0	1	0	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	0	1

Knowledge-base for Wumpus World

We will create a **knowledge base** for the wumpus world, and will derive some proves for the Wumpus-world using propositional logic. To build a knowledge base for wumpus world, we will use some rules and atomic propositions. We need symbol [i, j] for each location in the wumpus world, where i is for the location of rows, and j for column location.

Atomic proposition variable for Wumpus world:

- Let $P_{i,j}$ be true if there is a Pit in the room [i, j].
- Let $B_{i,j}$ be true if agent perceives breeze in [i, j].
- Let $W_{i,j}$ be true if there is wumpus in the square[i, j].
- Let $S_{i,j}$ be true if agent perceives stench in the square [i, j].
- Let $V_{i,j}$ be true if that square[i, j] is visited.
- Let $G_{i,j}$ be true if there is gold (and glitter) in the square [i, j].
- Let $OK_{i,j}$ be true if the room is safe.



Knowledge-base for Wumpus World

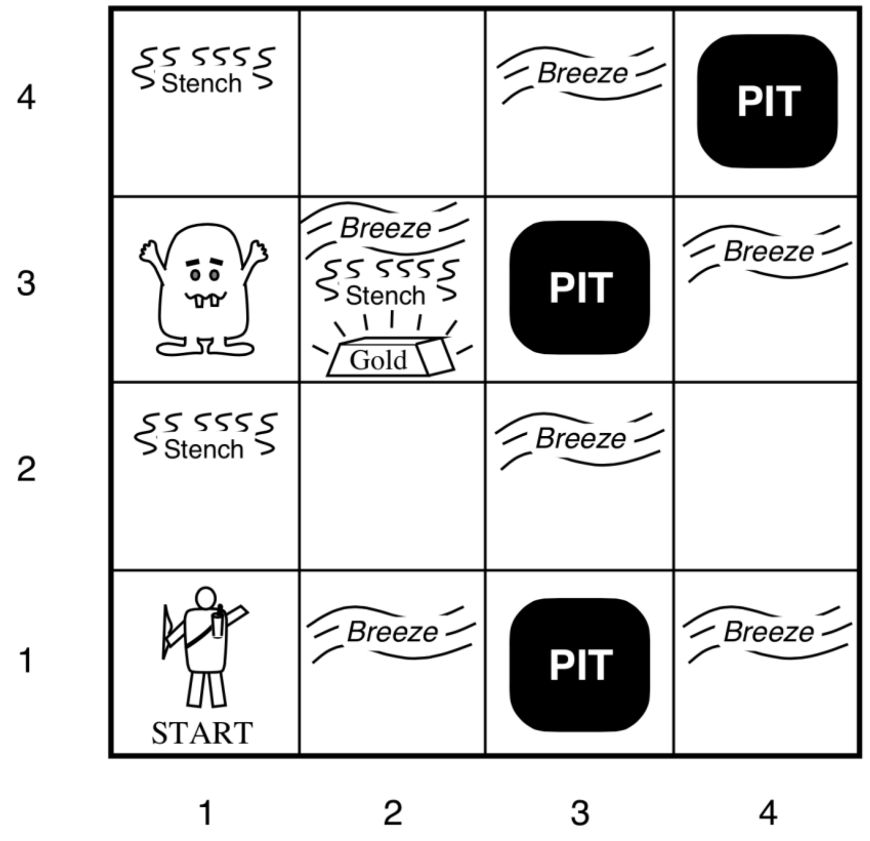
Some Propositional Rules for the wumpus world:

(R1) $\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

(R2) $\neg S_{21} \rightarrow \neg W_{11} \wedge \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$

(R3) $\neg S_{12} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{22} \wedge \neg W_{13}$

(R4) $S_{12} \rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$



Knowledge-base for Wumpus World

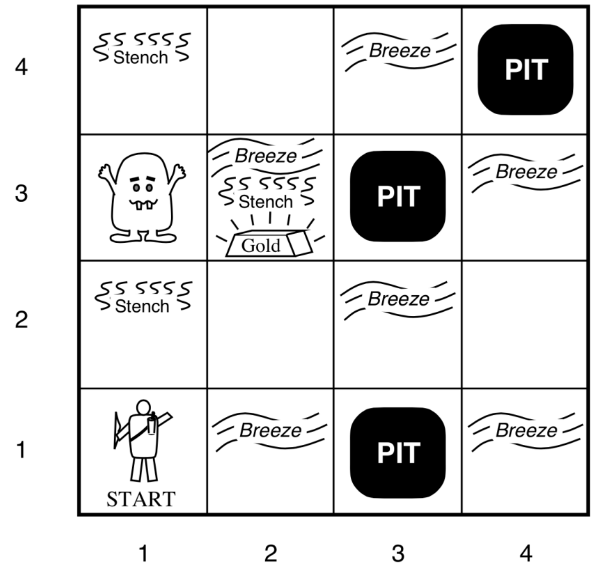
Representation of Knowledgebase for Wumpus World: Following is the Simple KB for wumpus world when an agent moves from room [1, 1], to room [2,1]:

$\neg W_{11}$	$\neg S_{11}$	$\neg P_{11}$	$\neg B_{11}$	$\neg G_{11}$	V_{11}	OK_{11}
$\neg W_{12}$	----	$\neg P_{12}$	-----	----	$\neg V_{12}$	OK_{12}
$\neg W_{21}$	$\neg S_{21}$	$\neg P_{21}$	B_{21}	$\neg G_{21}$	V_{21}	OK_{21}

Here in the first row, propositional variables for room[1,1] is showing that room does not have wumpus($\neg$ W_{11}), no stench ($\neg S_{11}$), no Pit($\neg P_{11}$), no breeze($\neg B_{11}$), no gold ($\neg G_{11}$), visited (V_{11}), and the room is Safe(OK_{11}).

In the second row, propositional variables for room [1,2], is showing that there is no wumpus, stench and breeze are unknown as an agent has not visited room [1,2], no Pit, not visited yet, and the room is safe.

In the third row, propositional variable for room[2,1], which is showing that there is no wumpus($\neg$ W21), no stench ($\neg S_{21}$), no Pit ($\neg P_{21}$), Perceives breeze(B_{21}), no glitter($\neg G_{21}$), visited (V_{21}), and room is safe (OK_{21}).



Knowledge-base for Wumpus World

Prove that Wumpus is in the room (1, 3):

We can prove that wumpus is in the room (1, 3) using propositional rules which we have derived for the wumpus world and using inference rule.

(R1) $\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

(R2) $\neg S_{21} \rightarrow \neg W_{11} \wedge \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$

(R3) $\neg S_{12} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{22} \wedge \neg W_{13}$

(R4) $S_{12} \rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$

**Apply Modus Ponens with ¬S11 and R1:

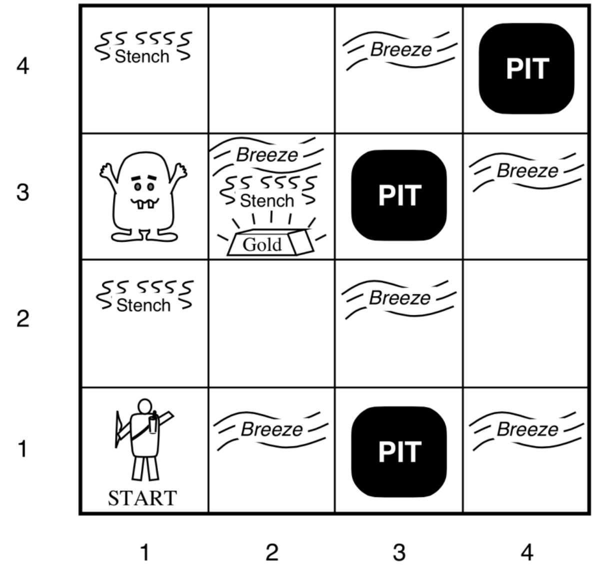
We will firstly apply MP rule with R1 which is ¬S11 → ¬ W11 ^ ¬ W12 ^ ¬ W21, and ¬S11 which will give the output ¬ W11 ^ W12 ^ W12.

$$\begin{tikzcd}
\boxed{\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}} \arrow[dr] & & \boxed{\neg S_{11}} \arrow[dl] \\
& \boxed{\neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}} &
\end{tikzcd}$$

**Apply And-Elimination Rule:

After applying And-elimination rule to ¬ W11 ∧ ¬ W12 ∧ ¬ W21, we will get three statements:

¬ W11, ¬ W12, and ¬W21.



Knowledge-base for Wumpus World

Prove that Wumpus is in the room (1, 3):

**Apply Modus Ponens to ¬S21, and R2:

Now we will apply Modus Ponens to ¬S21 and R2 which is ¬S21 → ¬ W21 ∧¬ W22 ∧ ¬ W31, which will give the Output as ¬ W21 ∧ ¬ W22 ∧¬ W31

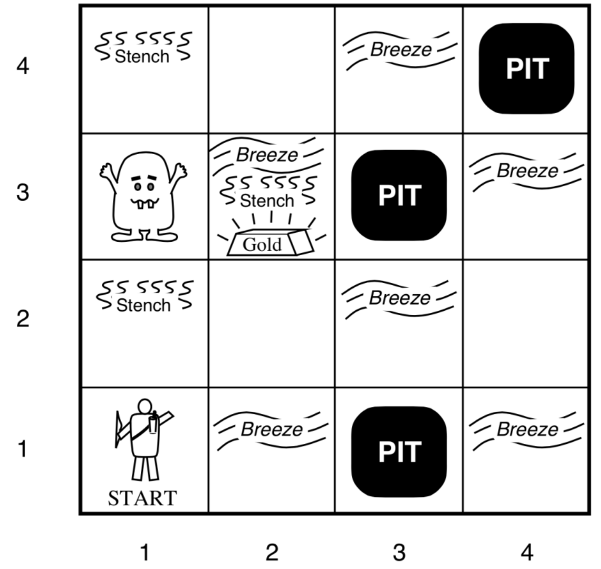
$$\begin{tikzcd}
\neg S_{21} \rightarrow \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31} \arrow[d] & \neg S_{21} \arrow[dl] \\
\neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31} &
\end{tikzcd}$$

(R1) $\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

(R2) $\neg S_{21} \rightarrow \neg W_{11} \wedge \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$

(R3) $\neg S_{12} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{22} \wedge \neg W_{13}$

(R4) $S_{12} \rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$



**Apply And-Elimination Rule:

Now again apply And-elimination rule to ¬ W21 ∧ ¬ W22 ∧¬ W31, We will get three statements:

¬ W21, ¬ W22, and ¬ W31.

Knowledge-base for Wumpus World

Prove that Wumpus is in the room (1, 3):

**Apply MP to S12 and R4: Apply Modus Ponens to S12 and R4 which is S12 → W13 ∨ W12 ∨ W22 ∨ W11, we will get the output as W13∨ W12 ∨ W22 ∨ W11.

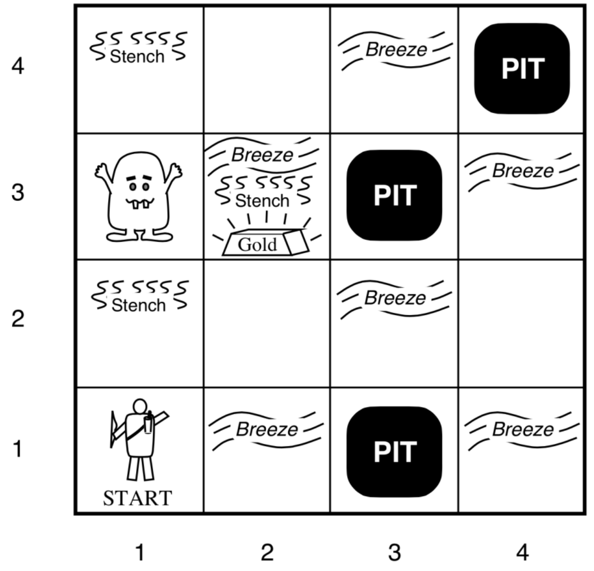
$$\begin{tikzcd} S_{12} \rightarrow W_{13} \lor W_{12} \lor W_{22} \lor W_{11} \arrow[dr] & & S_{12} \arrow[dl] \\ & W_{13} \lor W_{12} \lor W_{22} \lor W_{11} & \end{tikzcd}$$

(R1) $\neg S_{11} \rightarrow \neg W_{11} \land \neg W_{12} \land \neg W_{21}$

(R2) $\neg S_{21} \rightarrow \neg W_{11} \land \neg W_{21} \land \neg W_{22} \land \neg W_{31}$

(R3) $\neg S_{12} \rightarrow \neg W_{11} \land \neg W_{12} \land \neg W_{22} \land \neg W_{13}$

(R4) $S_{12} \rightarrow W_{13} \lor W_{12} \lor W_{22} \lor W_{11}$



**Apply Unit resolution on W13 ∨ W12 ∨ W22 ∨W11 and ¬ W11 :

After applying Unit resolution formula on W13 ∨ W12 ∨ W22 ∨ W11 and ¬ W11 we will get W13 ∨ W12 ∨ W22.

$$\begin{tikzcd} W_{13} \lor W_{12} \lor W_{22} \lor W_{11} \arrow[dr] & & \neg W_{11} \arrow[dl] \\ & (W_{13} \lor W_{12} \lor W_{22}) & \end{tikzcd}$$

Knowledge-base for Wumpus World

Prove that Wumpus is in the room (1, 3):

**Apply Unit resolution on W13 ∨ W12 ∨ W22 and ¬ W22 : After applying Unit resolution on W13 ∨ W12 ∨ W22, and ¬W22, we will get W13 ∨ W12 as output.

$$\begin{tikzcd}
W_{13} \lor W_{12} \lor W_{22} \arrow[dr] & & \neg W_{22} \arrow[dl] \\
& W_{13} \lor W_{12} &
\end{tikzcd}$$

**Apply Unit Resolution on W13 ∨ W12 and ¬ W12 :

After Applying Unit resolution on W13 ∨ W12 and ¬ W12, we will get W13 as an output, hence it is proved that the Wumpus is in the room [1, 3].

$$\begin{tikzcd}
W_{13} \lor W_{12} \arrow[dr] & & \neg W_{12} \arrow[dl] \\
& \boxed{W_{13}} &
\end{tikzcd}$$

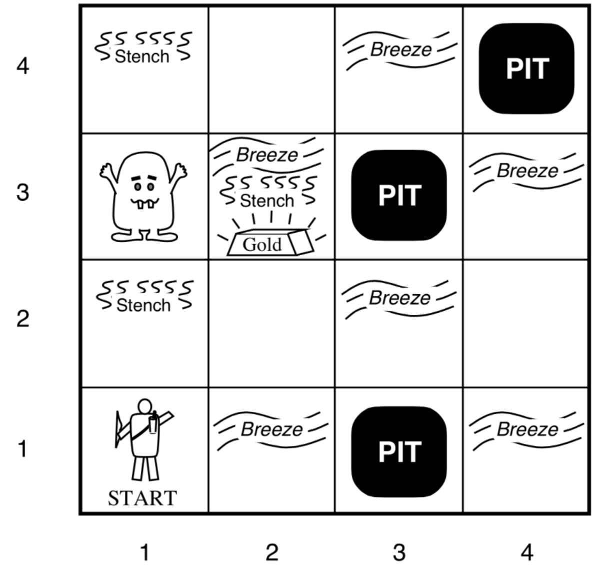
Proved.

(R1) $\neg S_{11} \rightarrow \neg W_{11} \land \neg W_{12} \land \neg W_{21}$

(R2) $\neg S_{21} \rightarrow \neg W_{11} \land \neg W_{21} \land \neg W_{22} \land \neg W_{31}$

(R3) $\neg S_{12} \rightarrow \neg W_{11} \land \neg W_{12} \land \neg W_{22} \land \neg W_{13}$

(R4) $S_{12} \rightarrow W_{13} \lor W_{12} \lor W_{22} \lor W_{11}$



04

PREDICATE LOGIC

Predicate Logic, Quantifier, Inference in FOL, Unifier in FOL

Predicate Logic

- Propositional logic, cannot adequately express the meaning of all statements in mathematics and in natural language. For example, suppose that we know that **"Every computer connected to the university network is functioning properly."** No rules of propositional logic allow us to conclude the truth of the statement.

- Statements involving variables, such as "$x > 3$," "$x = y + 3$," "$x + y = z$," and "**computer x is functioning properly**". These statements are neither true nor false when the values of the variables are not specified.

- A more powerful type of logic is predicate logic Which can be used to express the meaning of a wide range of statements in mathematics and computer science in ways that permit us to reason and explore relationships between objects.

Predicate Logic

- Let consider a statement "**X is greater than 3**"
- The statement "**x is greater than 3**" has two parts. The first part, the variable **x**, is the subject of the statement. The second part—the **predicate**, "**is greater than 3**"—refers to a property that the subject of the statement can have. We can denote the statement "**x is greater than 3**" by **P(x)**, where P denotes the predicate "**is greater than 3**" and x is the variable.
- The statement **P(x)** is also said to be the value of the propositional function **P** at **x**. Once a value has been assigned to the variable **x**, the statement **P(x)** becomes a proposition and has a truth value.

Basic Elements of First-order logic

Following are the basic elements of FOL syntax:

Constant	1, 2, A, John, Mumbai, cat,....
Variables	x, y, z, a, b,....
Predicates	Brother, Father, >,....
Function	sqrt, ...
Connectives	∧, ∨, ¬, ⇒, ⇔
Equality	==
Quantifier	∀, ∃

Atomic sentences in First-order logic

Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms. We can represent atomic sentences as **Predicate (term1, term2, ..., term n)**.

Example: Ravi and Ajay are brothers: => **Brothers(Ravi, Ajay).**

Chinky is a cat: => **cat (Chinky)**.

Complex sentences in First-order logic

Complex sentences are made by combining atomic sentences using connectives.

Example: Some boys play cricket

Here, the predicate is "play(x, y)," where x= boys, and y= game. Since there are some boys so we will use ∃, and it will be represented as:

$$\exists x \; boys(x) \rightarrow play(x, cricket).$$

Quantifier

- When the **variables** in a **propositional function** are assigned values, the resulting statement becomes a **proposition** with a certain **truth value**. However, there is another important way, called **quantification**, **to create a proposition from a propositional function**.
- **Quantification** expresses the extent to which a predicate is true over a range of elements. In English, the words **all, some, many, none**, and **few** are used in quantifications.
- There are two types of quantifier in predicate logic –
  - Universal Quantifier
  - Existential Quantifier

The Universal Quantifier

- The *universal quantification* of ***P(x)*** is the statement "***P(x)*** for all values of ***x*** in the domain."
- The notation **∀*xP(x)*** denotes the universal quantification of ***P(x)***. Here ∀ is called the **universal quantifier.** We read ∀***xP(x)*** as "for all ***x P(x)***" or "for every ***x P(x)***."
- An element for which ***P(x)*** is false is called a **counter example** of ∀***xP(x)*.**
- Besides "for all" and "for every," universal quantification can be expressed in many other ways, including "all of," "for each," "given any," "for arbitrary," "for each," and "for any."

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

Solution:

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

The Existential Quantifier

- The existential quantification of P(x) is the proposition "There exists an element x in the domain such that P(x)." We use the notation ∃xP(x) for the existential quantification of P(x). Here ∃ is called the existential quantifier.
- Besides the phrase "**there exists**,"we can also express existential quantification in many other ways, such as by using the words "**for some**," "**for at least one**," or "**there is**."
- The existential quantification **∃xP(x)** is read as "**There is an x such that P(x)**," "**There is at least one x such that P(x)**," or "**For some xP(x)**."

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

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

The Existential Quantifier

- Points to remember:

  **The syntax of FOL determines which collection of symbols is a logical expression in first-order logic. The basic syntactic elements of first-order logic are symbols. We write statements in short-hand notation in FOL.

  **The main connective for universal quantifier ∀ is implication →.

  **The main connective for existential quantifier ∃ is and ∧.

- Properties of Quantifiers:

  **In universal quantifier, ∀x∀y is similar to ∀y∀x.

  **In Existential quantifier, ∃x∃y is similar to ∃y∃x.

  **∃x∀y is not similar to ∀y∃x.

Some Examples of FOL using quantifier:

1. All birds fly.

   $\forall x\ bird(x) \rightarrow fly(x)$.

2. Mary loves everyone.

   $\forall x\ love\ (Mary, x)$

3. Every man respects his parent.

   $\forall x\ man(x) \rightarrow respects\ (x, parent)$

4. Some boys play cricket.

   $\exists x\ boys(x) \rightarrow play(x, cricket)$

5. Everyone loves himself.

   $\forall x\ love(x,x)$

6. Everyone loves everyone.

   $\forall x\ \forall y\ love\ (x, y)$

7. Every student smiles.

   $\forall x\ (student(x) \rightarrow smile(\ x))$

Some Examples of FOL using quantifier:

8. Every student except Kabila smiles.

$\forall x\ ((\text{student}(x) \land x \neq \text{Kabila}) \rightarrow \text{smile}(x))$

9. All students like both Mathematics and Science.

$\forall (x)\ [\ \text{student}(x) \rightarrow \text{like}(x, \text{Mathematics}) \land \text{like}(x, \text{Science})].$

9. Not all students like both Mathematics and Science.

$\neg\forall (x)\ [\ \text{student}(x) \rightarrow \text{like}(x, \text{Mathematics}) \land \text{like}(x, \text{Science})].$

10. Everyone walks or talks.

$\forall x\ (\text{walk}(x) \lor \text{talk}(x))$

11. Every student walks or talks.

$\forall x\ (\text{student}(x) \rightarrow (\text{walk}(x) \lor \text{talk}(x))$

12. Every student who loves Mary is happy.

$\forall x\ ((\text{student}(x) \land \text{love}(x, \text{Mary})) \rightarrow \text{happy}(x)))$

14. Everyone loves someone.

$\forall x\ \exists y\ \text{love}(x, y)$

Inference in First-Order Logic

- Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences. Before understanding the FOL inference rule, let's understand some basic terminologies used in FOL.
- **Substitution**: Substitution is a fundamental operation performed on terms and formulas. It occurs in all inference systems in first-order logic. The substitution is complex in the presence of quantifiers in FOL. If we write F[a/x], so it refers to substitute a constant "a" in place of variable "x".
- **Equality**: First-Order logic does not only use predicate and terms for making atomic sentences but also uses another way, which is equality in FOL. For this, we can use equality symbols which specify that the two terms refer to the same object.

  Example: Brother (John) = Smith.

  As in the above example, the object referred by the Brother (John) is similar to the object referred by Smith. The equality symbol can also be used with negation to represent that two terms are not the same objects.

  Example: $\neg(x=y)$ which is equivalent to $x \neq y$.

Inference Rules in First-Order Logic

- As propositional logic we also have inference rules in first-order logic, so following are some basic inference rules in FOL:
  - Universal Generalization
  - Universal Instantiation
  - Existential Instantiation
  - Existential introduction

Universal Generalization (Inference in FOL)

- Universal generalization is a valid inference rule which states that if premise P(c) is true for any arbitrary element c in the universe of discourse, then we can have a conclusion as ∀ x P(x).
- It can be represented as: $\dfrac{P(c)}{\forall x\, P(x)}$
- This rule can be used if we want to show that every element has a similar property.

Example:

Let's represent,

P(c): "A byte contains 8 bits",

so for ∀ x P(x) "All bytes contain 8 bits.", it will also be true.

Universal Instantiation (Inference in FOL)

- Universal instantiation is also called as universal elimination or UI is a valid inference rule. It can be applied multiple times to add new sentences.
- The UI rule state that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from ∀ x P(x) for any object in the universe of discourse.
- It can be represented as: $\frac{\forall x\, P(x)}{P(c)}$

Example: 1

IF "Every person like ice-cream"=> ∀x P(x)

so we can infer that

"John likes ice-cream" => P(c)

Example: 2.

Let's take a famous example,

"All kings who are greedy are Evil." So let our knowledge base contains this detail as in the form of FOL:

∀x king(x) ∧ greedy (x) → Evil (x),

So from this information, we can infer any of the following statements using Universal Instantiation:

•King(John) ∧ Greedy (John) → Evil (John),

•King(Richard) ∧ Greedy (Richard) → Evil (Richard),

•King(Father(John)) ∧ Greedy (Father(John)) → Evil (Father(John)),

Existential Instantiation (Inference in FOL)

- Existential instantiation is also called as Existential Elimination, which is a valid inference rule in first-order logic. It can be applied only once to replace the existential sentence.
- This rule states that one can infer P(c) from the formula given in the form of ∃x P(x) for a new constant symbol c. The restriction with this rule is that c used in the rule must be a new term for which P(c) is true.
- It can be represented as: $\dfrac{\exists x\, P(x)}{P(c)}$

Example:

Let's illustrate the previous example,

"All kings who are greedy are Evil." So let our knowledge base contains this detail as in the form of FOL:

∀x king(x) ∧ greedy (x) → Evil (x),

So from this information, we can infer **King(John) ∧ Greedy (John) → Evil (John)** as long as John does not appear in the knowledge base.

Existential Introduction (Inference in FOL)

- An existential introduction is also known as an **existential generalization**, which is a valid inference rule in first-order logic.
- This rule states that if there is some element c in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P.
- It can be represented as: $\dfrac{P(c)}{\exists x P(x)}$

Example:
Let's say that,
P(c): "Priyanka got good marks in English."
so for **∃x P(x)** "someone got good marks in English.“ it will also be true.

Unification in FOL

- Unification is all about making the expression look identical. So, for the given expression to make them look identical we need to do substitution.

Example: Let's say there are two different expressions, P(x, f(y)), and P(a, f(g(z))).

In this example,
we need to make both above statements identical to each other. For this, we will perform the substitution.

P(x, f(y))......... (i)
P(a, f(g(z)))......... (ii)

Substitute x with a, and y with g(z) in the first expression, and it will be represented as a/x and g(z)/y.

With both the substitutions, the first expression will be identical to the second expression and the substitution set will be: [a/x, g(z)/y].

Conditions for unification in FOL

Following are some basic conditions for unification:

- Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified.
- Number of Arguments in both expressions must be identical.
- Unification will fail if there are two similar variables present in the same expression.

Unification Exercise

- Find the Most General Unifier of Q(a, g(x, a), f(y)), Q(a, g(f(b), a), x)}

Here, L_1 = Q(a, g(x, a), f(y)), and L_2 = Q(a, g(f(b), a), x)

S_0 => {Q(a, g(x, a), f(y)); Q(a, g(f(b), a), x)}

SUBST θ= {f(b)/x}

S_1 => {Q(a, g(f(b), a), f(y)); Q(a, g(f(b), a), f(b))}

SUBST θ= {b/y}

S_1 => {Q(a, g(f(b), a), f(b)); Q(a, g(f(b), a), f(b))}, **Successfully Unified.**

Unifier: [a/a, f(b)/x, b/y].

05

RESOLUTION

Resolution Method in AI, Resolution Method in Propositional Logic, Resulation Method in FOL

Resolution Method in AI

- Resolution method is an inference rule which is used in both Propositional as well as First-order Predicate Logic in different ways. This method is basically used for proving the satisfiability of a sentence.
- Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions.
- The key idea for the resolution method is to use the knowledge base and negated goal to obtain null clause(which indicates contradiction)
- Since the knowledge base itself is consistent, the contradiction must be introduced by a negated goal. As a result, we have to conclude that the original goal is true.

Resolution Method in Propositional Logic

- In propositional logic, resolution method is the only inference rule which gives a new clause when two or more clauses are coupled together.
- The process followed to convert the propositional logic into resolution method contains the below steps:
  - Convert the given axiom into clausal form.
  - Apply and proof the given goal using negation rule.
  - Use those literals which are needed to prove.
  - Solve the clauses together and achieve the goal.
- Before solving problems using Resolution method, let’s understand two normal forms:
  - Conjunctive Normal Form(CNF)
  - Disjunctive Normal Form (DNF)

Conjunctive Normal Form(CNF) [Resolution]

In propositional logic, the resolution method is applied only to those clauses which are disjunction of literals. There are following steps used to convert into CNF:

1) Eliminate bi-conditional implication by replacing A ⇔ B with (A → B) ∧ (B →A)

2) Eliminate implication by replacing A → B with ¬A V B.

3) In CNF, negation(¬) appears only in literals, therefore we move it inwards as:

- ¬ (¬A) ≡ A (double-negation elimination
- ¬ (A ∧ B) ≡ (¬A V ¬B) (De Morgan)
- ¬(A V B) ≡ (¬A ∧ ¬B) (De Morgan)

4) Finally, using distributive law on the sentences, and form the CNF as:

(A1 V B1) ∧ (A2 V B2) ∧ …. ∧ (An V Bn).

Note: **CNF can also be described as AND of ORS**

Disjunctive Normal Form (DNF) [Resolution]

This is a reverse approach of CNF. The process is similar to CNF with the following difference:

(A1 Λ B1) V (A2 Λ B2) V…V (An Λ Bn).

In DNF, it is OR of ANDS, a sum of products, or a cluster concept, whereas, in CNF, it is ANDs of Ors.

Example of Propositional Resolution

Consider the following Knowledge Base:

The humidity is high or the sky is cloudy.

If the sky is cloudy, then it will rain.

If the humidity is high, then it is hot.

It is not hot.

Goal: **It will rain.**

Use propositional logic and apply resolution method to prove that the goal is derivable from the given knowledge base.

Example of Propositional Resolution

Solution:

Let's construct propositions of the given sentences one by one:

1) Let, P: Humidity is high.

Q: Sky is cloudy.

It will be represented as **P V Q**.

2) Let, Q: Sky is cloudy.

Let, R: It will rain.

It will be represented as **Q → R**.

3) Let, P: Humidity is high.

Let, S: It is hot.

It will be represented as **P → S**.

4) Let, **¬S**: It is not hot.

Knowledge Base:

The humidity is high or the sky is cloudy.

If the sky is cloudy, then it will rain.

If the humidity is high, then it is hot.

It is not hot.

Goal: **It will rain.**

Now,

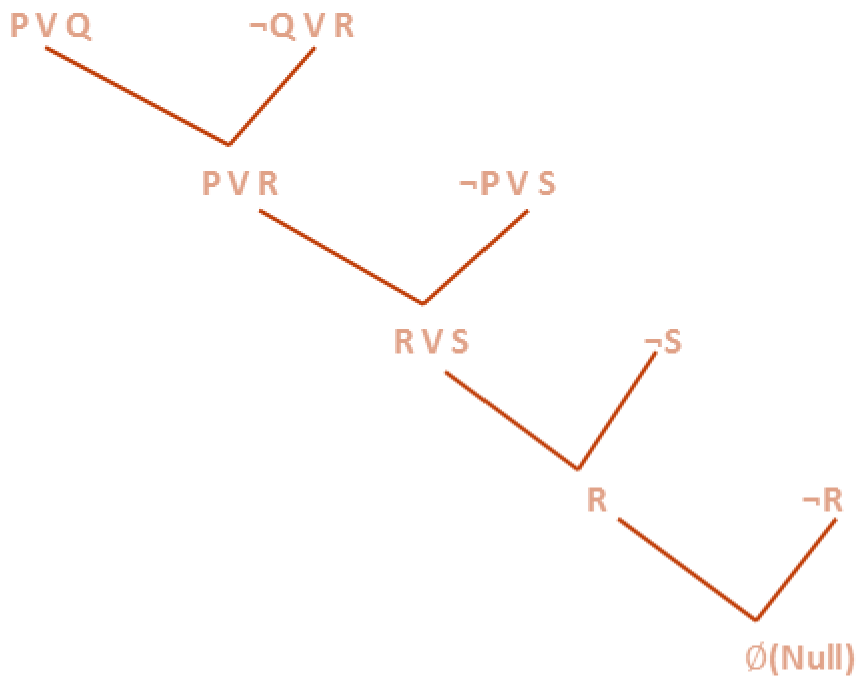
In (2), Q → R will be converted as (¬Q V R)

In (3), P → S will be converted as (¬P V S)

Negation of Goal (¬R): It will not rain.

Example of Propositional Resolution

Finally, apply the rule as shown below:



Knowledge Base:

1. P V Q
2. Q → R : (¬Q V R)
3. P → S: (¬P V S)
4. ¬S

Goal: **It will rain.**

Negation of Goal **(¬R)**

After applying Proof by Refutation (Contradiction) on the goal, the problem is solved, and it has terminated with a Null clause (Ø). Hence, the goal is achieved. Thus, It is not raining.

Example of Propositional Resolution

Consider the following

Knowledge Base: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$

Goal: $\neg P_{1,2}$

Use propositional logic and apply resolution method to prove that the goal is derivable from the given knowledge base.

Self Task!

Resolution Method in Predicate Logic

- Resolution method in FOPL is an uplifted version of propositional resolution method.
- In FOPL, the process to apply the resolution method is as follows:
  - Conversion of facts into first-order logic.
  - Convert FOL statements into CNF
  - Negate the statement which needs to prove (proof by contradiction)
  - Draw resolution graph (unification). Unlike propositional logic, FOPL literals are complementary if one unifies with the negation of other literal.

For example: {Bird(F(x)) V Loves(G(x), x)} and {¬Loves(a, b) V ¬Kills(a, b)}
Eliminate the complementary literals Loves(G(x),x) and Loves(a,b)) with **θ ={a/G(x), b/x}** to give the following output clause: **{Bird(F(x)) V ¬Kills(G(x),x)}**

Example of Resolution in FOL

Consider the following Knowledge Base:

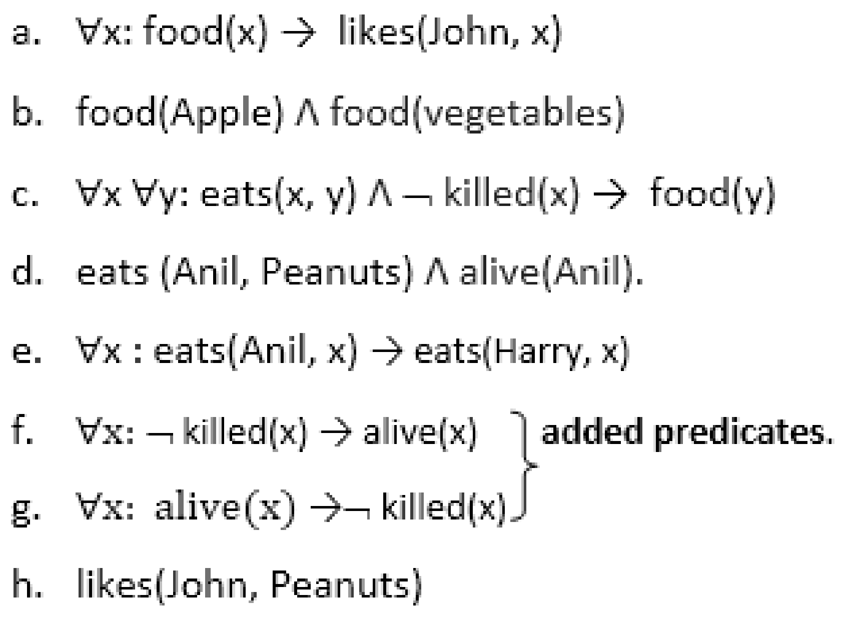
1. John likes all kind of food.
2. Apple and vegetable are food
3. Anything anyone eats and not killed is food.
4. Anil eats peanuts and still alive
5. Harry eats everything that Anil eats.

Goal: **John likes peanuts.**

Use FOL and apply resolution method to prove that the goal is derivable from the given knowledge base.

[Solution] Step-1: Conversion of Facts into FOL

In the first step we will convert all the given statements into its first order logic.



a. $\forall x$: food(x) $\rightarrow$ likes(John, x)

b. food(Apple) $\land$ food(vegetables)

c. $\forall x \forall y$: eats(x, y) $\land \neg$ killed(x) $\rightarrow$ food(y)

d. eats (Anil, Peanuts) $\land$ alive(Anil).

e. $\forall x$: eats(Anil, x) $\rightarrow$ eats(Harry, x)

f. $\forall x$: $\neg$ killed(x) $\rightarrow$ alive(x)

g. $\forall x$: alive(x) $\rightarrow \neg$ killed(x)

added predicates.

h. likes(John, Peanuts)

KB:

1. John likes all kind of food.
2. Apple and vegetable are food
3. Anything anyone eats and not killed is food.
4. Anil eats peanuts and still alive
5. Harry eats everything that Anil eats.

Goal: **John likes peanuts.**

[Solution] Step-2: Conversion of FOL into CNF

2.1) Eliminate all implication (→) and rewrite:

a. ∀x ¬ food(x) V likes(John, x)

b. food(Apple) Λ food(vegetables)

c. ∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)

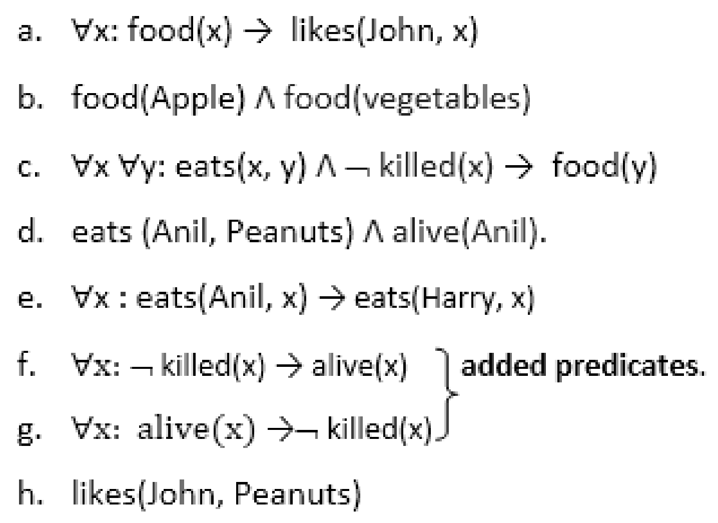
d. eats (Anil, Peanuts) Λ alive(Anil)

e. ∀x ¬ eats(Anil, x) V eats(Harry, x)

f. ∀x¬ [¬ killed(x)] V alive(x)

g. ∀x ¬ alive(x) V ¬ killed(x)

h. likes(John, Peanuts).



a. ∀x: food(x) → likes(John, x)

b. food(Apple) Λ food(vegetables)

c. ∀x ∀y: eats(x, y) Λ ¬ killed(x) → food(y)

d. eats (Anil, Peanuts) Λ alive(Anil).

e. ∀x : eats(Anil, x) → eats(Harry, x)

f. ∀x: ¬ killed(x) → alive(x) } **added predicates.**

g. ∀x: alive(x) →¬ killed(x)

h. likes(John, Peanuts)

[Solution] Step-2: Conversion of FOL into CNF

2.2) Move negation (¬)inwards and rewrite:

a. ∀x ¬ food(x) V likes(John, x)

b. food(Apple) Λ food(vegetables)

c. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)

d. eats (Anil, Peanuts) Λ alive(Anil)

e. ∀x ¬ eats(Anil, x) V eats(Harry, x)

g. ∀x ¬killed(x)] V alive(x)

h. ∀x ¬ alive(x) V ¬ killed(x)

i. likes(John, Peanuts)

After eliminating implication:

a. ∀x ¬ food(x) V likes(John, x)

b. food(Apple) Λ food(vegetables)

c. ∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)

d. eats (Anil, Peanuts) Λ alive(Anil)

e. ∀x ¬ eats(Anil, x) V eats(Harry, x)

f. ∀x¬ [¬ killed(x)] V alive(x)

g. ∀x ¬ alive(x) V ¬ killed(x)

h. likes(John, Peanuts).

[Solution] Step-2: Conversion of FOL into CNF

2.3) Rename variables or standardize variables

a. ∀x ¬ food(x) V likes(John, x)

b. food(Apple) Λ food(vegetables)

c. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)

d. eats (Anil, Peanuts) Λ alive(Anil)

e. ∀w¬ eats(Anil, w) V eats(Harry, w)

f. ∀g ¬killed(g)] V alive(g)

g. ∀k ¬ alive(k) V ¬ killed(k)

i. likes(John, Peanuts).

After move negation (¬) inwards :

a. ∀x ¬ food(x) V likes(John, x)

b. food(Apple) Λ food(vegetables)

c. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)

d. eats (Anil, Peanuts) Λ alive(Anil)

e. ∀x ¬ eats(Anil, x) V eats(Harry, x)

g. ∀x ¬killed(x)] V alive(x)

h. ∀x ¬ alive(x) V ¬ killed(x)

i. likes(John, Peanuts)

[Solution] Step-2: Conversion of FOL into CNF

2.4) Eliminate existential instantiation quantifier by elimination:

In this step, we will eliminate existential quantifier ∃, and this process is known as Skolemization. But in this example problem since there is no existential quantifier so all the statements will remain same in this step.

2.5) Drop Universal quantifiers:

In this step we will drop all universal quantifier since all the Statements are not implicitly quantified so we don't need it.

a. ¬ food(x) V likes(John, x)
b. food(Apple)
c. food(vegetables)
d. ¬ eats(y, z) V killed(y) V food(z)
e. eats (Anil, Peanuts)
f. alive(Anil)
g. ¬ eats(Anil, w) V eats(Harry, w)
h. killed(g) V alive(g)
i. ¬ alive(k) V ¬ killed(k)
j. likes(John, Peanuts).

Rename variables or standardize variables

a. ∀x ¬ food(x) V likes(John, x)
b. food(Apple) Λ food(vegetables)
c. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
d. eats (Anil, Peanuts) Λ alive(Anil)
e. ∀w¬ eats(Anil, w) V eats(Harry, w)
f. ∀g ¬killed(g)] V alive(g)
g. ∀k ¬ alive(k) V ¬ killed(k)
i. likes(John, Peanuts).

2.6) Drop Universal quantifiers: This step will not make any change in this problem.

[Solution] Step-3: Negate the statement to be proved

In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts)

After applying CNF and Step-3:

a. ¬ food(x) V likes(John, x)

b. food(Apple)

c. food(vegetables)

d. ¬ eats(y, z) V killed(y) V food(z)

e. eats (Anil, Peanuts)

f. alive(Anil)

g. ¬ eats(Anil, w) V eats(Harry, w)

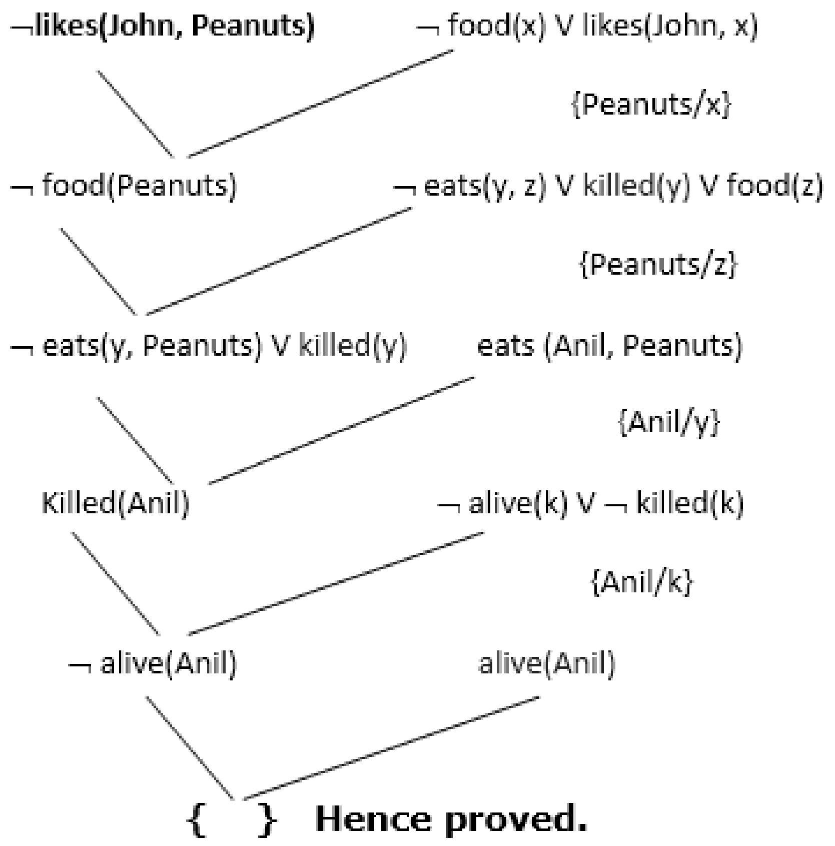
h. killed(g) V alive(g)

i. ¬ alive(k) V ¬ killed(k)

j. ¬ likes(John, Peanuts).

[Solution] Step-4: Draw Resolution graph

Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows:



After applying CNF and Step-3:

a. ¬ food(x) V likes(John, x)

b. food(Apple)

c. food(vegetables)

d. ¬ eats(y, z) V killed(y) V food(z)

e. eats (Anil, Peanuts)

f. alive(Anil)

g. ¬ eats(Anil, w) V eats(Harry, w)

h. killed(g) V alive(g)

i. ¬ alive(k) V ¬ killed(k)

j. ¬ likes(John, Peanuts).

06

INFERENCE ENGINE

Forward Chaining, Backward Chaining

Inference Engine

- The inference engine is the component of the intelligent system in artificial intelligence, which applies logical rules to the knowledge base to infer new information from known facts. The first inference engine was part of the expert system. Inference engine commonly proceeds in two modes, which are:
  - Forward chaining
  - Backward chaining

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

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

Forward Chaining

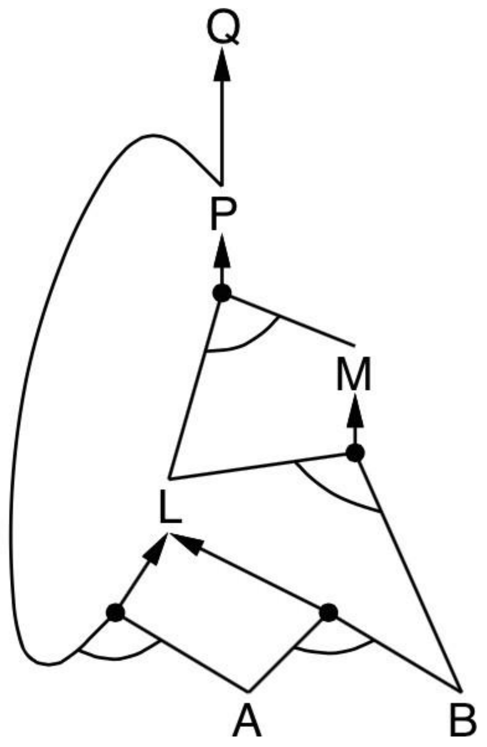
- **Forward chaining** is also known as a forward deduction or forward reasoning method when using an **inference engine**. Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more 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.

Properties of Forward-Chaining:

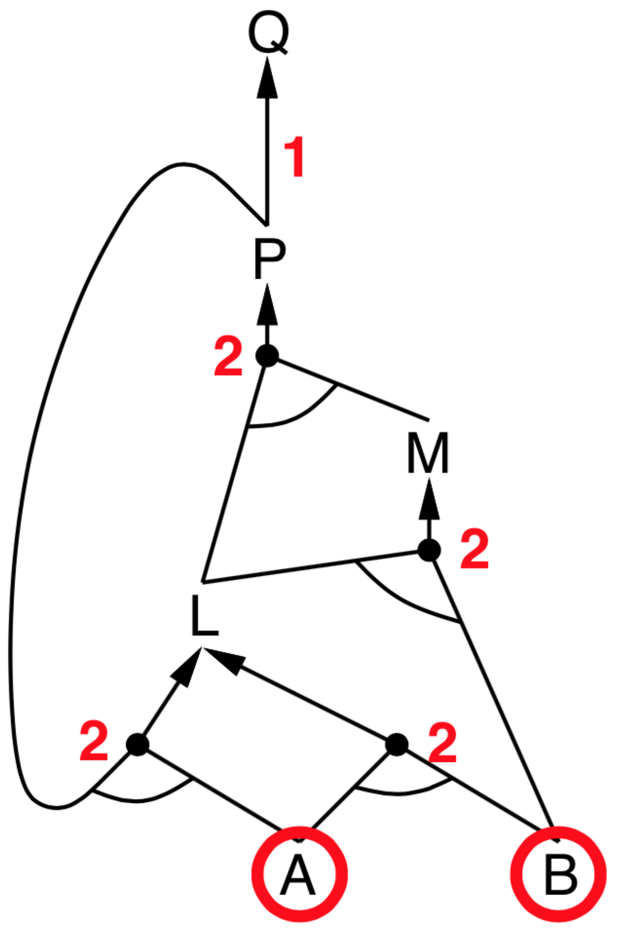
- It is a down-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward-chaining approach is also called as data-driven as we reach to the goal using available data.
- Forward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

Forward Chaining

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

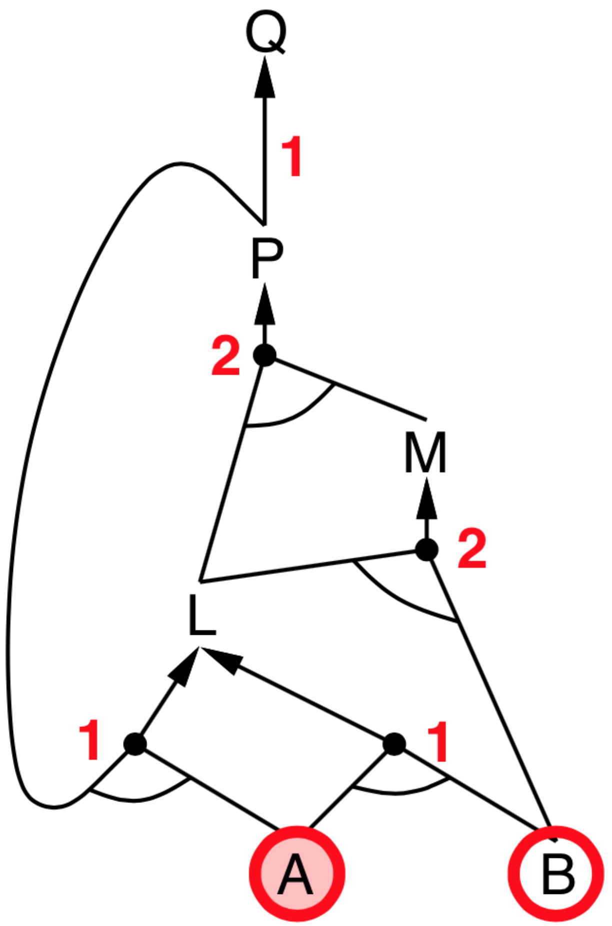


Forward Chaining



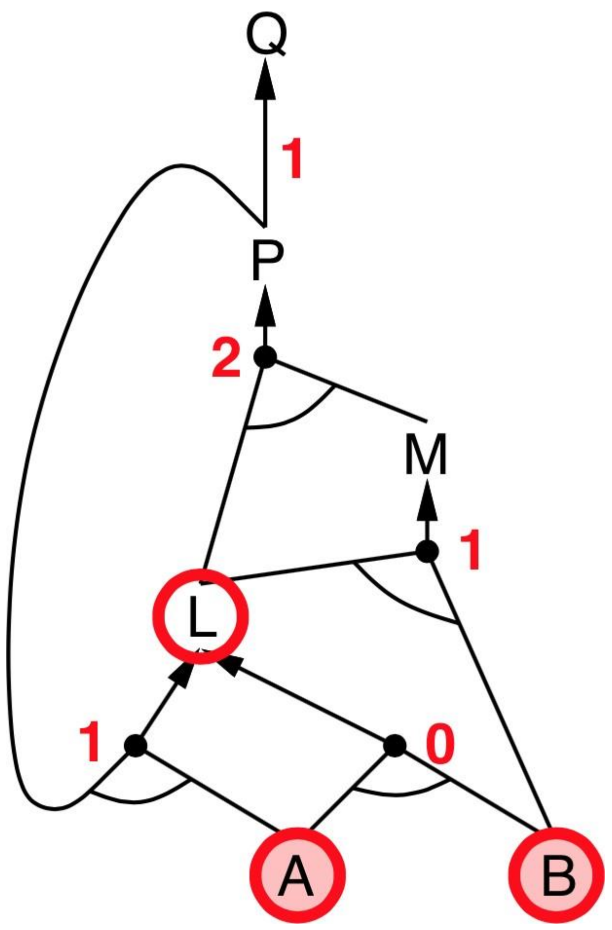
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Forward Chaining



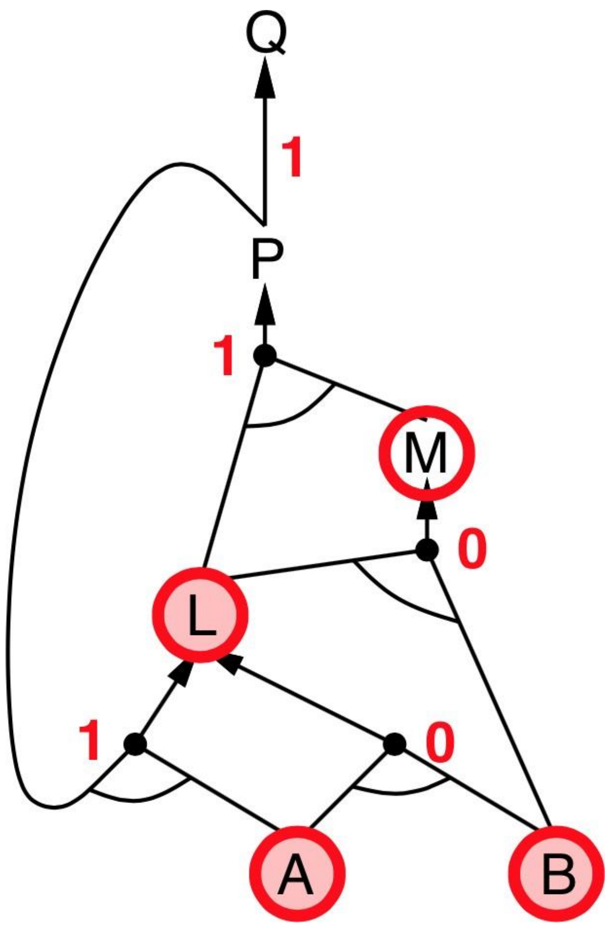
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Forward Chaining



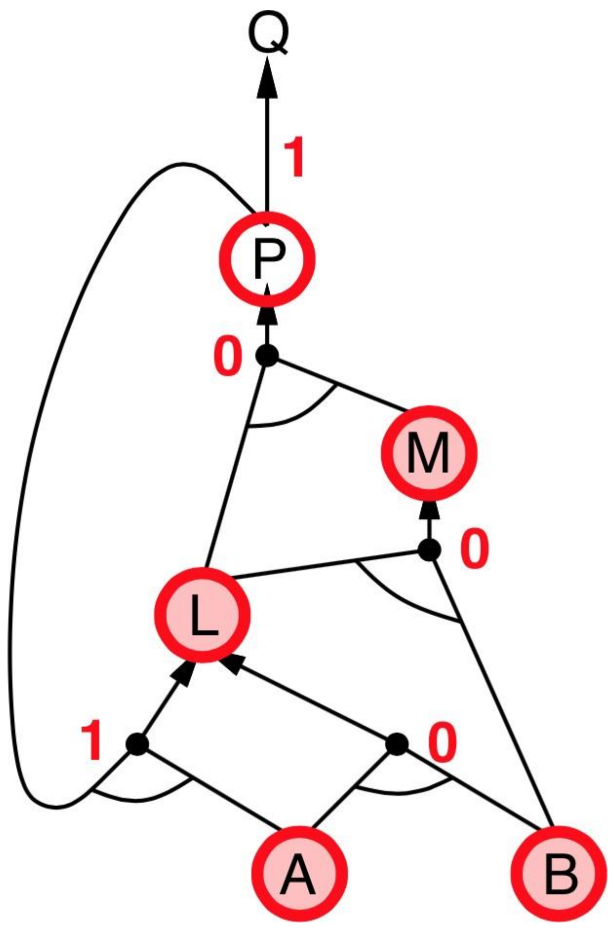
$$
\begin{aligned}
P &\Rightarrow Q \\
L \wedge M &\Rightarrow P \\
B \wedge L &\Rightarrow M \\
A \wedge P &\Rightarrow L \\
A \wedge B &\Rightarrow L \\
A & \\
B &
\end{aligned}
$$

Forward Chaining



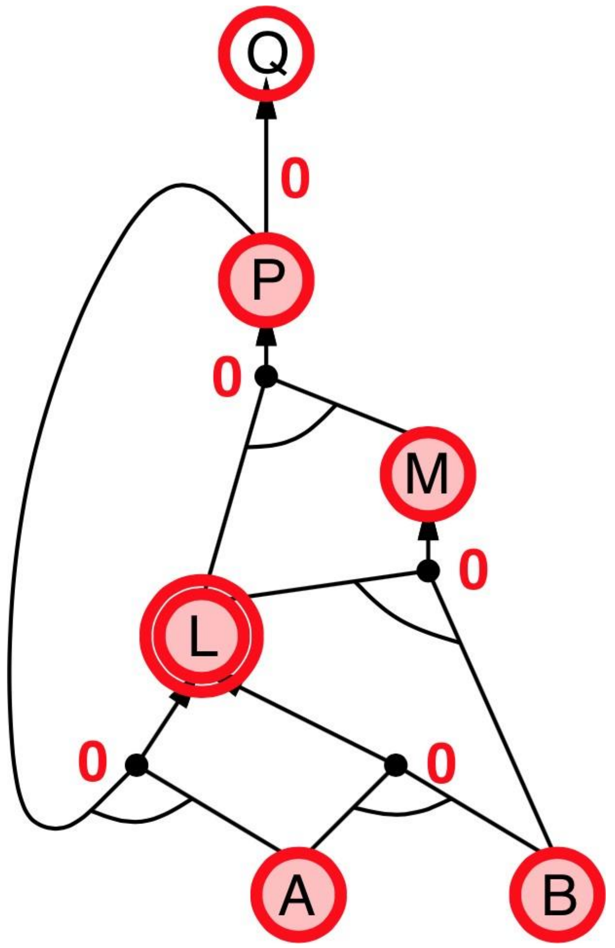
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Forward Chaining



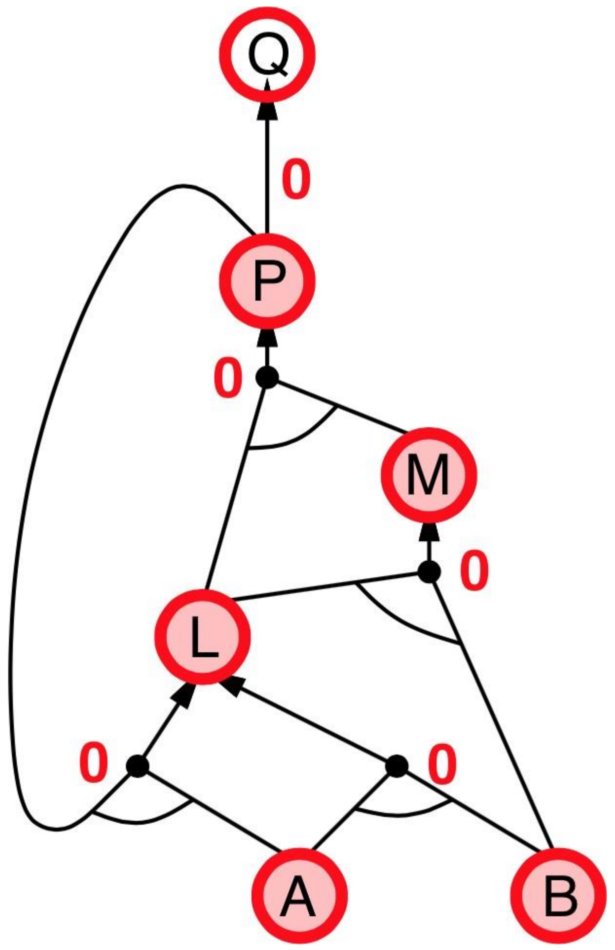
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Forward Chaining



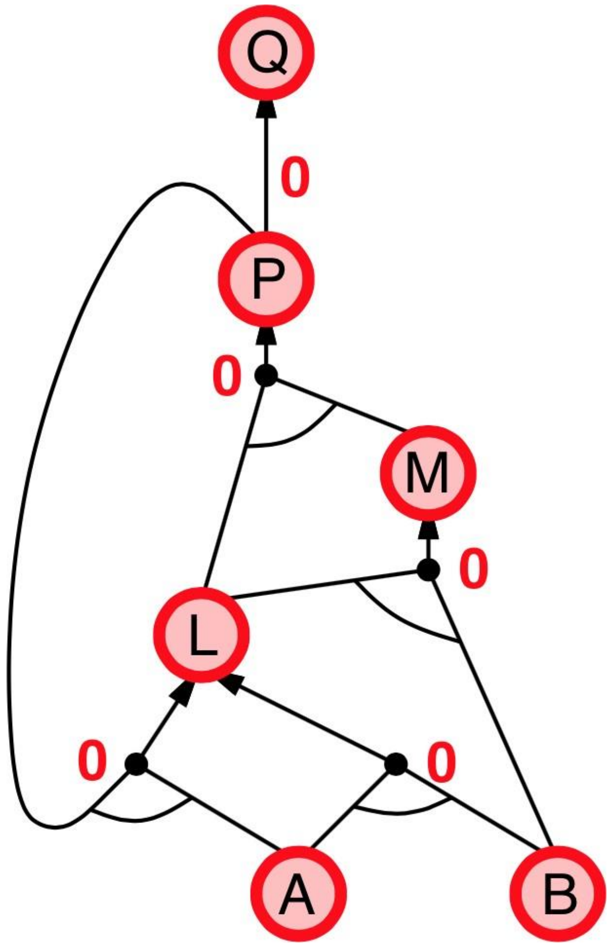
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Forward Chaining



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

Forward Chaining



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

Forward Chaining Example

- "As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen."
- Prove that "**Robert is criminal.**"

To solve the above problem,

first, we will convert all the above facts into first-order definite clauses, and then we will use a **forward-chaining algorithm** to reach the goal.

Forward Chaining Example

Facts Conversion into FOL:

"As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen."

It is a crime for an American to sell weapons to hostile nations. (Let's say p, q, and r are variables)
American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)

Country A has some missiles. ∃p Owns(A, p) ∧ Missile(p). It can be written in two definite clauses by using Existential Instantiation, introducing new Constant T1.
Owns(A, T1) ......(2)
Missile(T1) ....(3)

All of the missiles were sold to country A by Robert.
∀p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A) ......(4)

Missiles are weapons.
Missile(p) → Weapons (p) ....(5)

Enemy of America is known as hostile.
Enemy(p, America) →Hostile(p) ....(6)

Country A is an enemy of America.
Enemy (A, America) ....(7)

Robert is American
American(Robert). ....(8)

Forward Chaining Example

Step-1: In the first step we will start with the known facts and will choose the sentences which do not have implications, such as:

1. American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p)
2. Owns(A, T1)
3. Missile(T1)
4. ∀p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A)
5. Missile(p) → Weapons (p)
6. Enemy(p, America) →Hostile(p)
7. Enemy (A, America)
8. American(Robert)

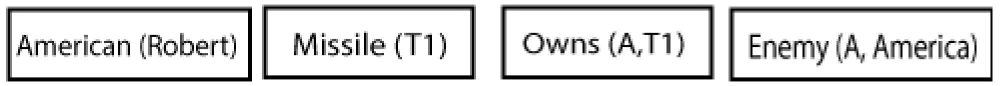
American(Robert),

Enemy(A, America),

Owns(A, T1), and

Missile(T1).

All these facts will be represented as below.

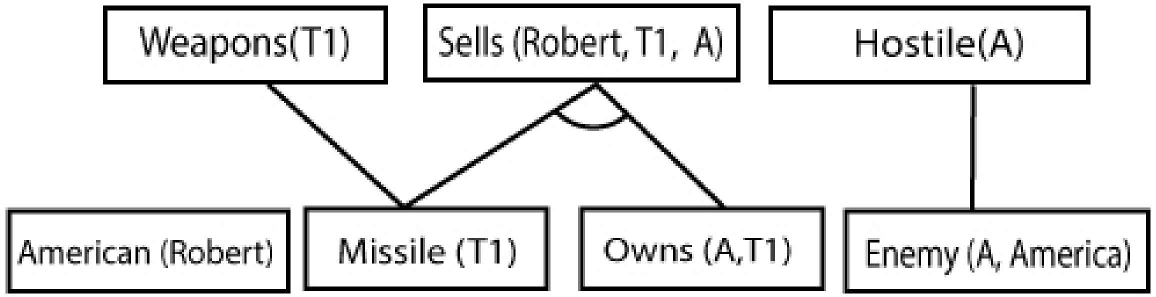


Forward Chaining Example

Step-2: At the second step, we will see those facts which infer from available facts and with satisfied premises.

- Rule-(1) does not satisfy premises, so it will not be added in the first iteration.
- Rule-(2) and (3) are already added.
- Rule-(4) satisfy with the substitution {p/T1}, so Sells (Robert, T1, A) is added, which infers from the conjunction of Rule (2) and (3).
- Rule-(6) is satisfied with the substitution(p/A), so Hostile(A) is added and which infers from Rule-(7).

1. American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p)
2. Owns(A, T1)
3. Missile(T1)
4. ∀p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A)
5. Missile(p) → Weapons (p)
6. Enemy(p, America) →Hostile(p)
7. Enemy (A, America)
8. American(Robert)

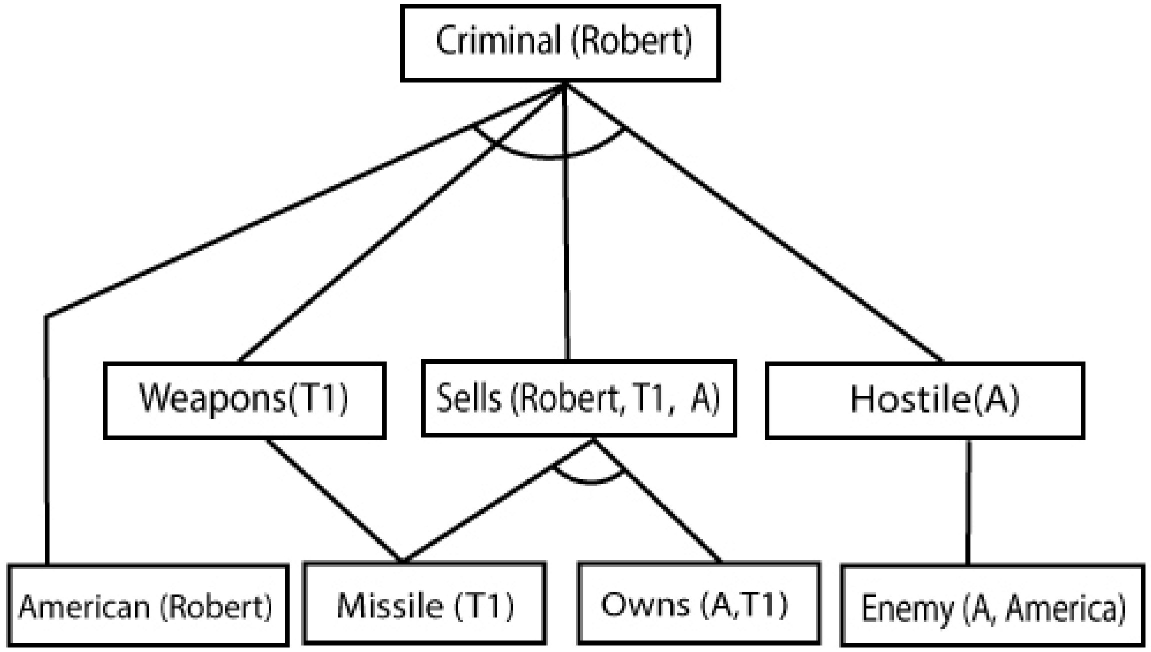


Forward Chaining Example

Step-3:

As we can check Rule-(1) is satisfied with the substitution **{p/Robert, q/T1, r/A}**, so we can add **Criminal(Robert)** which infers all the available facts. And hence we reached our goal statement.

1. American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p)
2. Owns(A, T1)
3. Missile(T1)
4. ∀p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A)
5. Missile(p) → Weapons (p)
6. Enemy(p, America) →Hostile(p)
7. Enemy (A, America)
8. American(Robert)



Backward Chaining

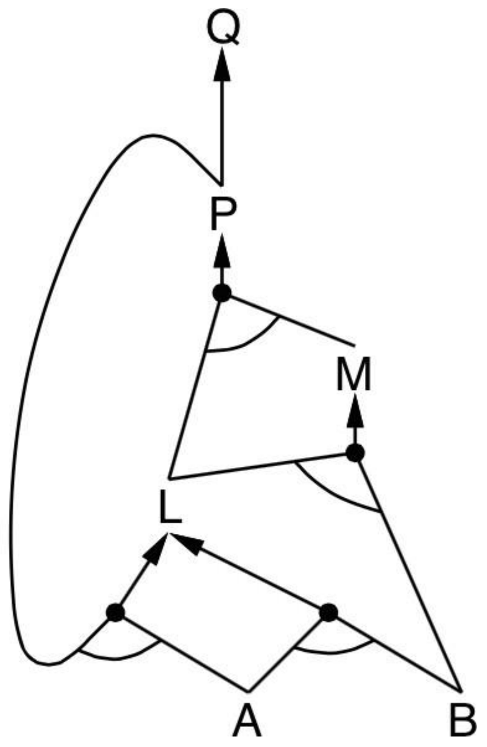
- **Backward-chaining** is also known as a backward deduction or backward reasoning method when using an **inference engine**. A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

  Properties of backward chaining:

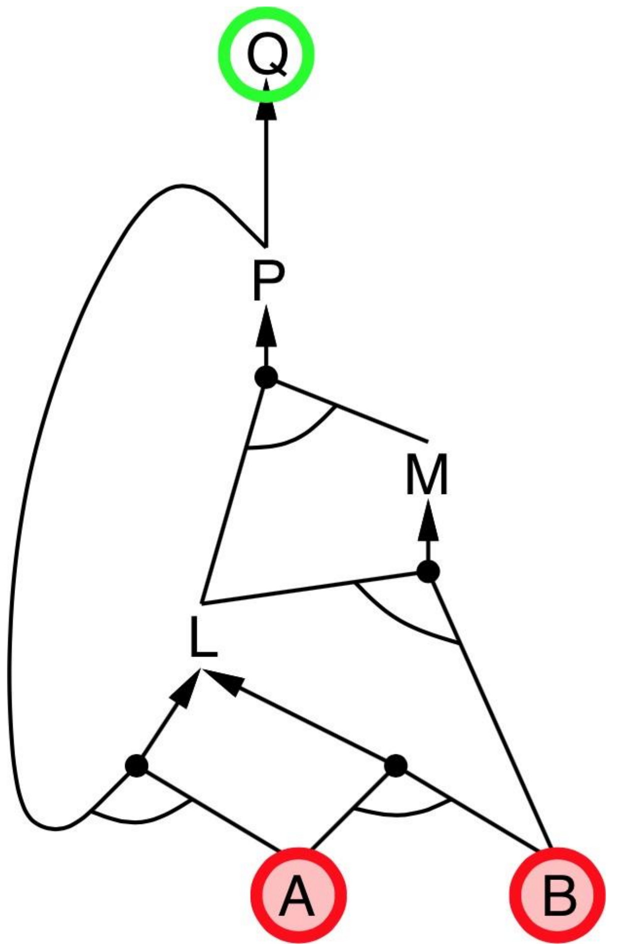
  - It is known as a top-down approach.
  - Backward-chaining is based on modus ponens inference rule.
  - In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.
  - It is called a goal-driven approach, as a list of goals decides which rules are selected and used.
  - Backward -chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.
  - The backward-chaining method mostly used a depth-first search strategy for proof.

Backward Chaining

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

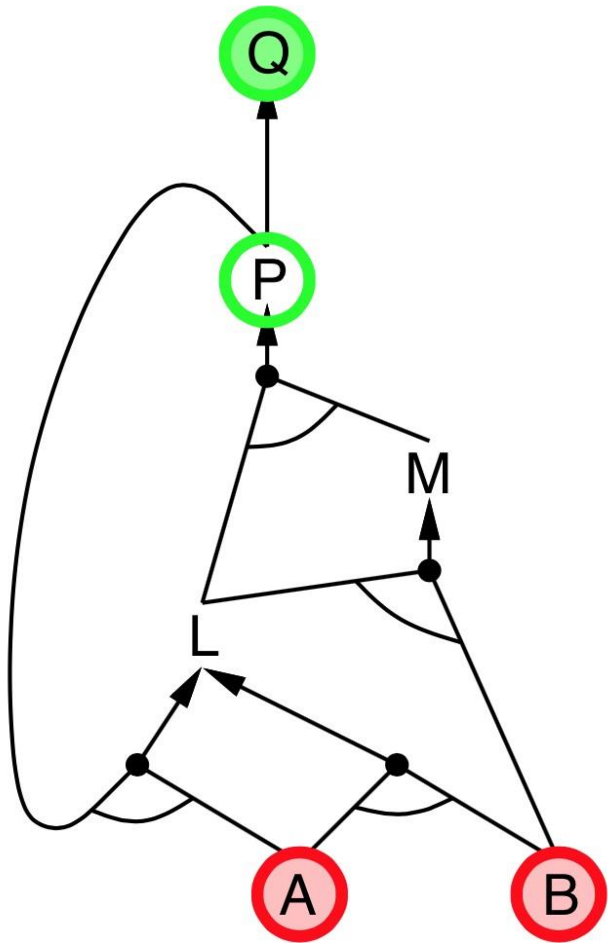


Backward Chaining



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

Backward Chaining



$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

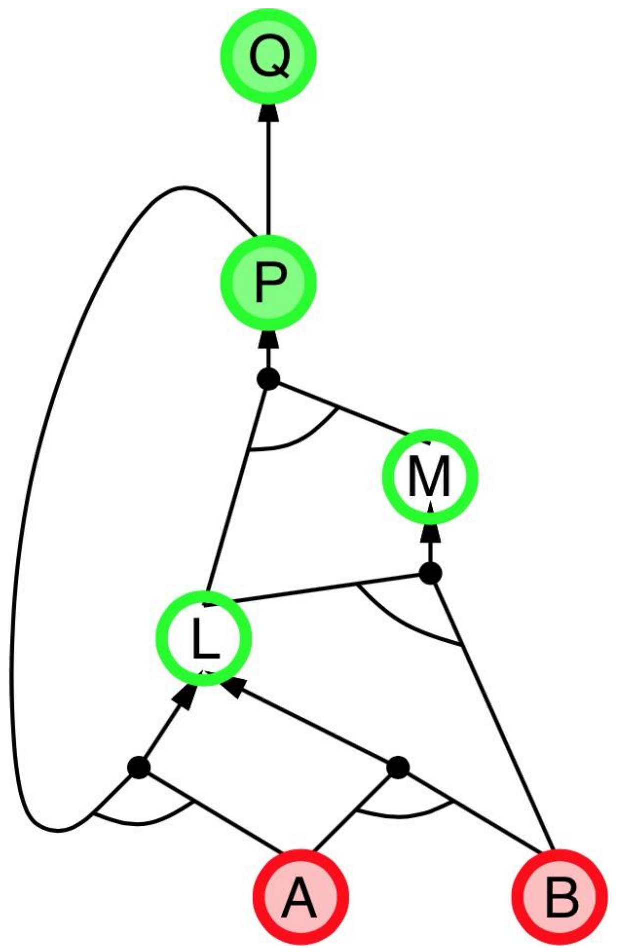
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

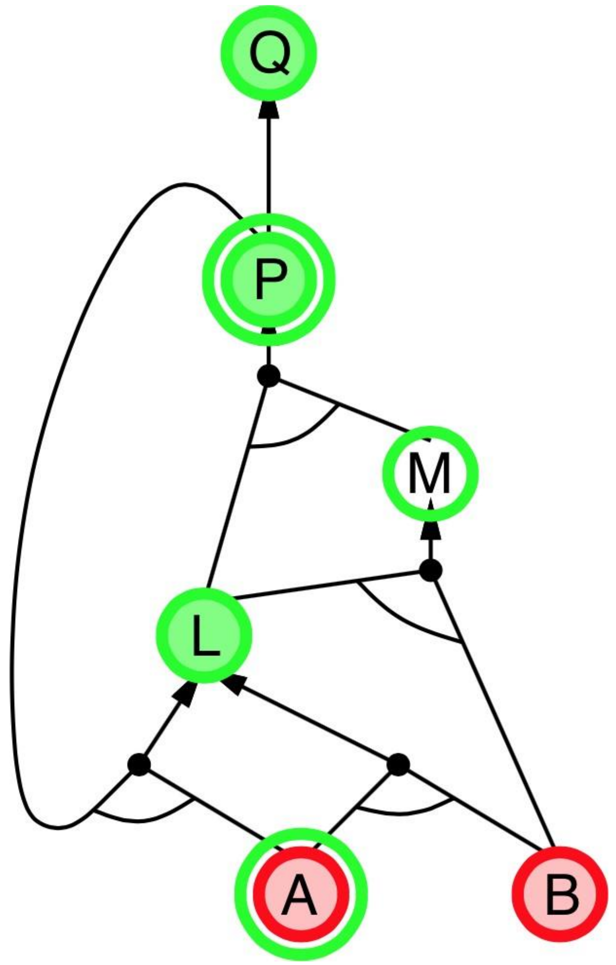
B

Backward Chaining



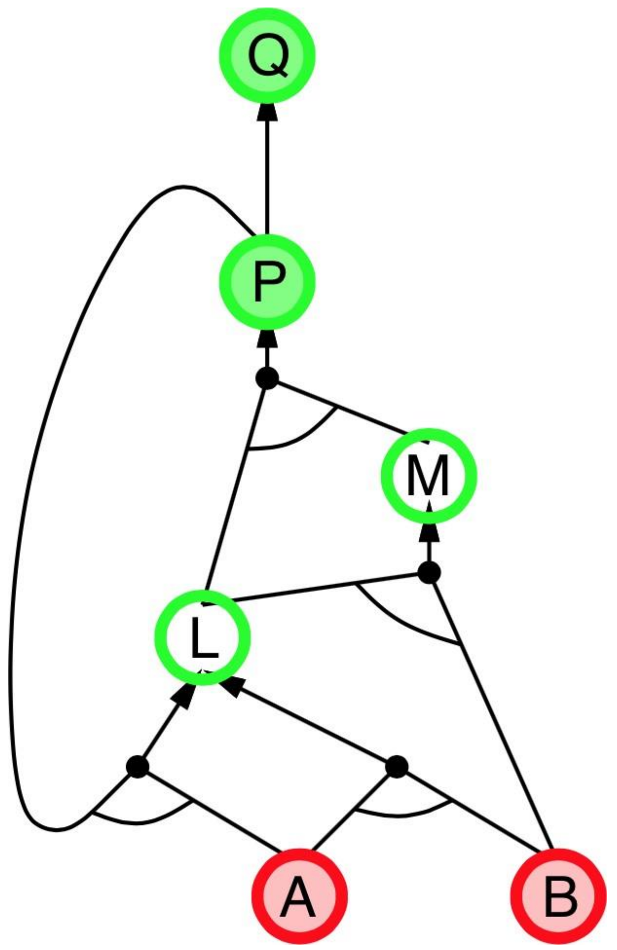
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Backward Chaining



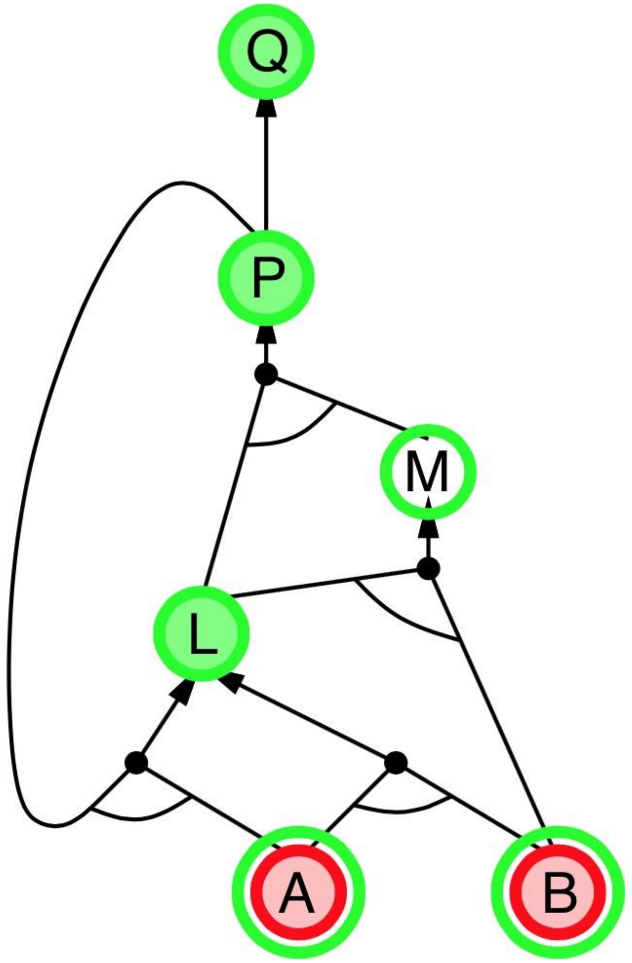
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Backward Chaining



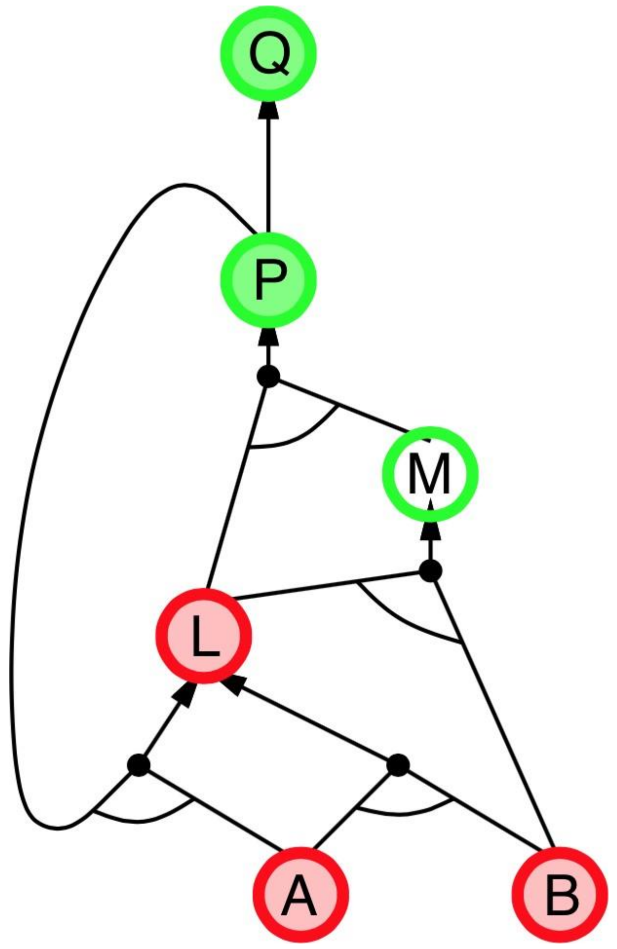
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Backward Chaining



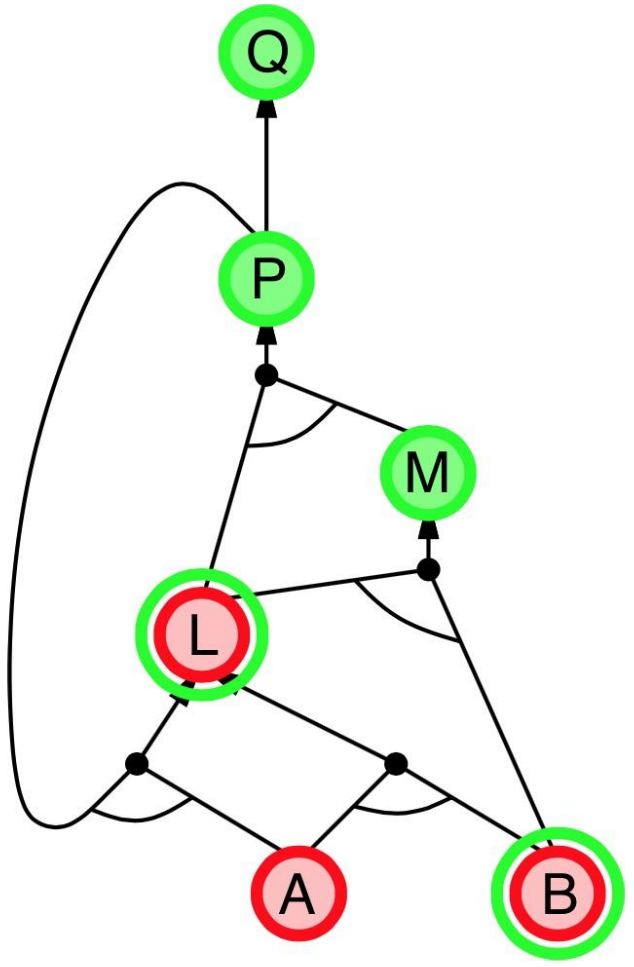
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Backward Chaining



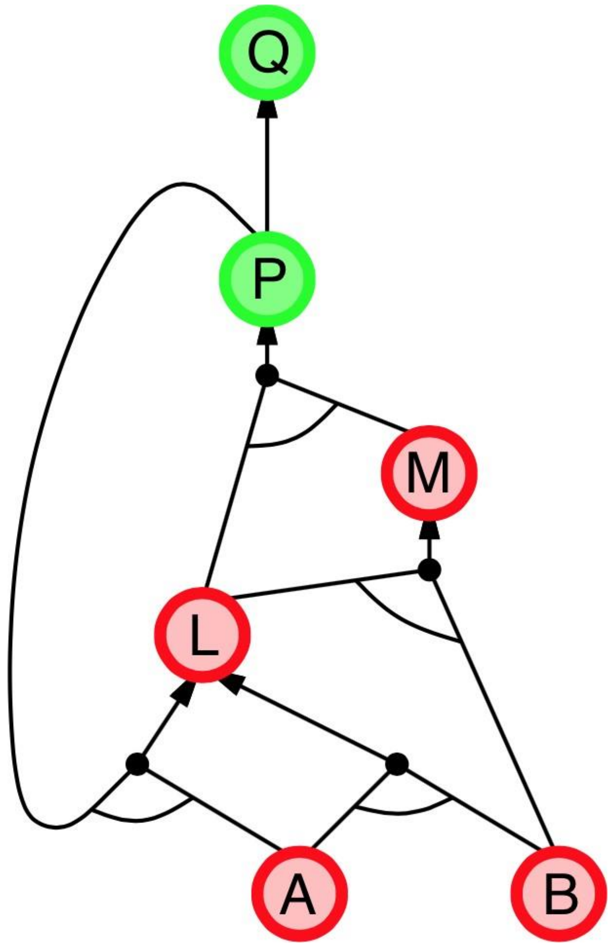
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Backward Chaining



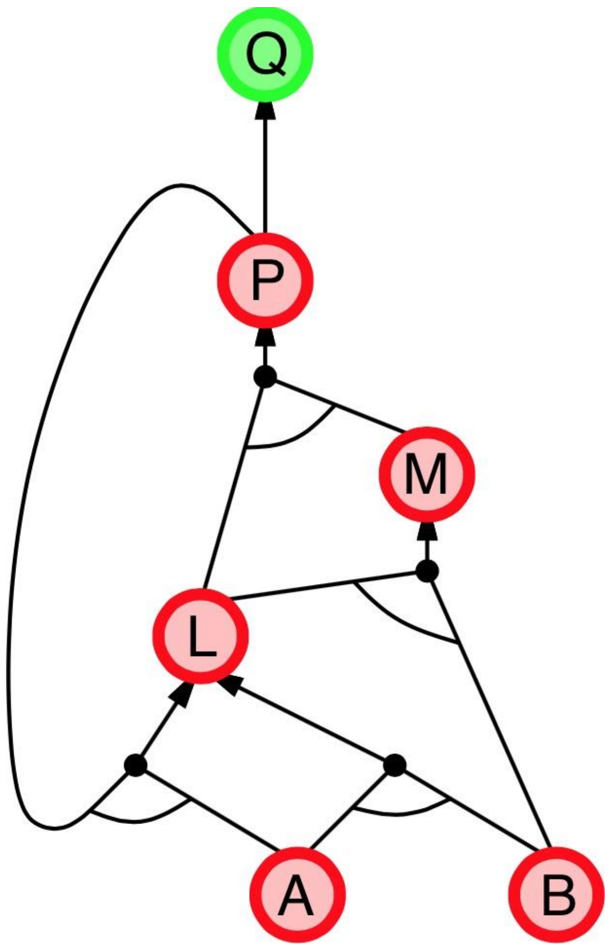
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Backward Chaining



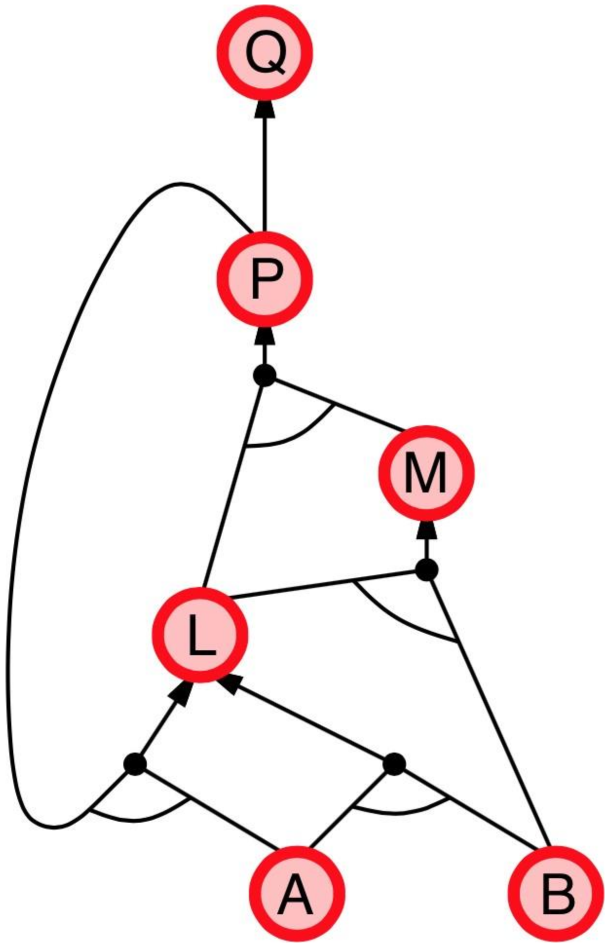
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Backward Chaining



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

Backward Chaining



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

Backward Chaining Example

- **"As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen."**
- Prove that "**Robert is criminal."**

To solve the above problem,

first, we will convert all the above facts into first-order definite clauses, and then we will use a **backward algorithm** to reach the goal.

Backward Chaining Example

Facts Conversion into FOL:

"As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen."

It is a crime for an American to sell weapons to hostile nations. (Let's say p, q, and r are variables)

American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)

Country A has some missiles. **∃p Owns(A, p) ∧ Missile(p)**. It can be written in two definite clauses by using Existential Instantiation, introducing new Constant T1.

Owns(A, T1) ......(2)

Missile(T1) ....(3)

All of the missiles were sold to country A by Robert.

∀p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A) ......(4)

Missiles are weapons.

Missile(p) → Weapons (p) ....(5)

Enemy of America is known as hostile.

Enemy(p, America) →Hostile(p) ....(6)

Country A is an enemy of America.

Enemy (A, America) ....(7)

Robert is American

American(Robert). ....(8)

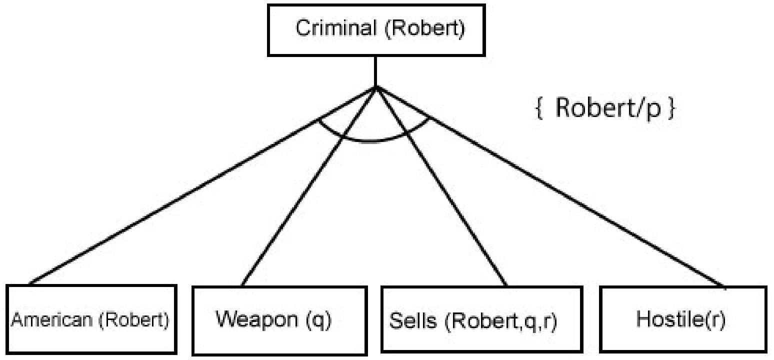
Backward Chaining Example

Step-1: At the first step, we will take the goal fact. And from the goal fact, we will infer other facts, and at last, we will prove those facts true. So our goal fact is "**Robert is Criminal**," so following is the predicate of it.

$$\boxed{\text{Criminal (Robert)}}$$

Step-2: At the second step, we will infer other facts form goal fact which satisfies the rules. So as we can see in **Rule-1**, the goal predicate **Criminal (Robert)** is present with substitution **{Robert/P}**. So we will add all the conjunctive facts below the first level and will replace **p** with **Robert**.

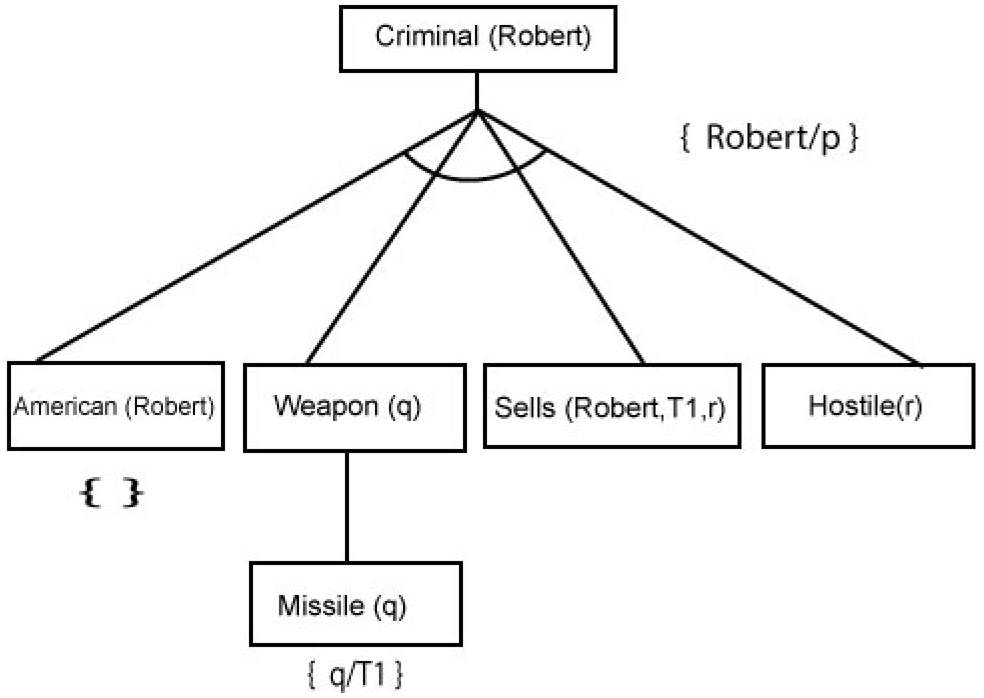
Here we can see **American (Robert)** is a fact, so it is proved here.



1. American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p)
2. Owns(A, T1)
3. Missile(T1)
4. ∀p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A)
5. Missile(p) → Weapons (p)
6. Enemy(p, America) →Hostile(p)
7. Enemy (A, America)
8. American(Robert)

Backward Chaining Example

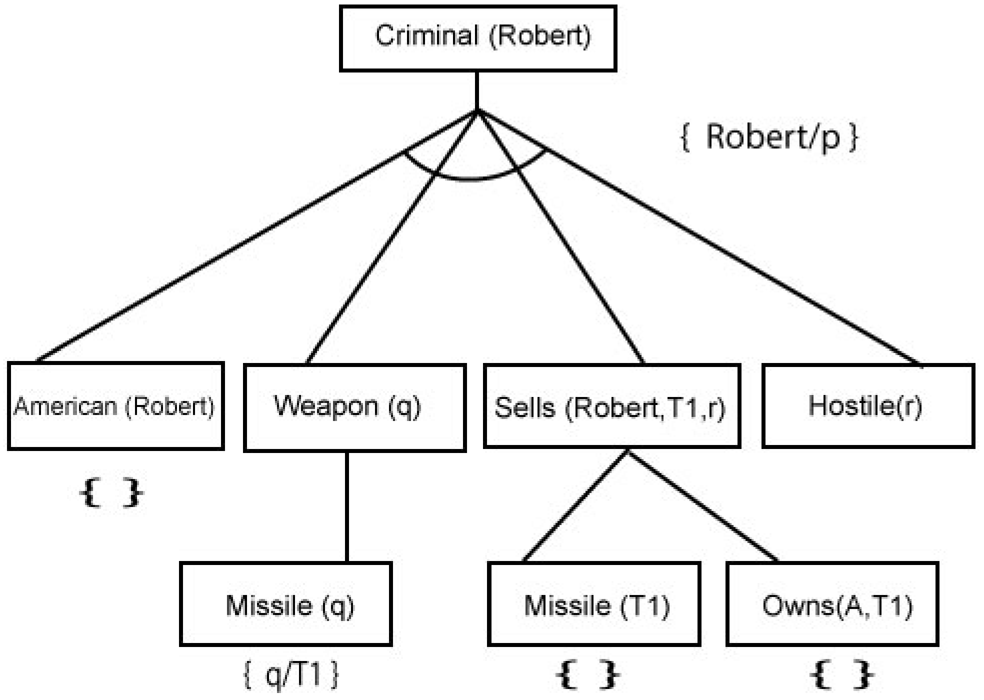
Step-3: At step-3, we will extract further fact **Missile(q)** which infer from **Weapon(q)**, as it satisfies **Rule-(5)**. **Weapon (q)** is also true with the substitution of a constant **T1** at **q.**



1. American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p)
2. Owns(A, T1)
3. Missile(T1)
4. ∀p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A)
5. Missile(p) → Weapons (p)
6. Enemy(p, America) →Hostile(p)
7. Enemy (A, America)
8. American(Robert)

Backward Chaining Example

Step-4: At step-4, we can infer facts **Missile(T1)** and **Owns(A, T1)** form **Sells(Robert, T1, r)** which satisfies the **Rule- 4**, with the substitution of **A** in place of **r**. So these two statements are proved here.

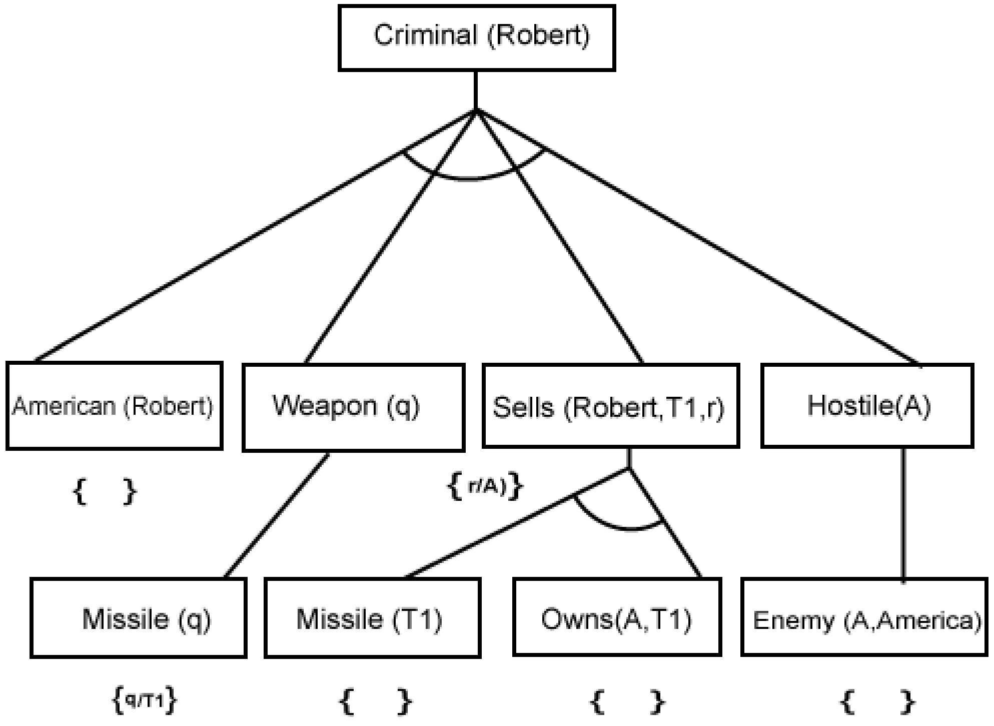


1. American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p)
2. Owns(A, T1)
3. Missile(T1)
4. ∀p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A)
5. Missile(p) → Weapons (p)
6. Enemy(p, America) →Hostile(p)
7. Enemy (A, America)
8. American(Robert)

Backward Chaining Example

Step-5: At step-5, we can infer the fact **Enemy(A, America)** from **Hostile(A)** which satisfies **Rule- 6**. And hence all the statements are proved true using backward chaining.

1. American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p)
2. Owns(A, T1)
3. Missile(T1)
4. ∀p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A)
5. Missile(p) → Weapons (p)
6. Enemy(p, America) →Hostile(p)
7. Enemy (A, America)
8. American(Robert)



THANK YOU!

Any Questions?