MODULE 4

4	**Knowledge and Reasoning**		10	CO3
	4.1	Knowledge based Agents, The Wumpus World, inference procedures, First Order Logic: Syntax and Semantic, Inference in FOL, Unification and lifting, Forward chaining, backward Chaining, Resolution, Answer set programming		

Uncertain Knowledge and Reasoning:

Uncertainty, Acting under uncertainty, Representing knowledge in an uncertain domain, The semantics of belief network, Inference in Bayesian network,

KNOWLEDGE BASED AGENTS

- “Humans, it seems, know things and, what they know helps them do things”
  - These are not empty statements.
  - They make strong claims about how the intelligence of humans is **achieved—not by purely reflex mechanisms** but by **processes of reasoning** that operate on internal representations of knowledge.

In AI, this approach to intelligence is embodied in KNOWLEDGE-BASED AGENTS

An intelligent agent needs **knowledge** about the real world for taking decisions and **reasoning** to act efficiently.

- Knowledge-based agents are those agents who have the capability of
  - **maintaining an internal state of knowledge,**
  - **reason over that knowledge,**
  - **update their knowledge after observations and**
  - **take actions.**

These agents can represent the world with some formal representation and act intelligently.

- **For example,(knowledge & implementation level)**
  - Consider an "**automated air conditioner**." The inbuilt knowledge stored in its system is that " It would **adjust its temperature according to the weather**." This represents the **knowledge level** of the agent. The **actual working** and its adjustment define the **implementation level** of the knowledge-based agent.
- Knowledge-based agents are composed of two main parts:
  - **Knowledge-base and**
  - **Inference system**.

Another example?

A knowledge-based agent[central component] must able to do the following:

- Represent states, actions, etc.
- Incorporate new percepts
- Update & Deduce the internal representation of the world
- Deduce appropriate actions.

Inference system

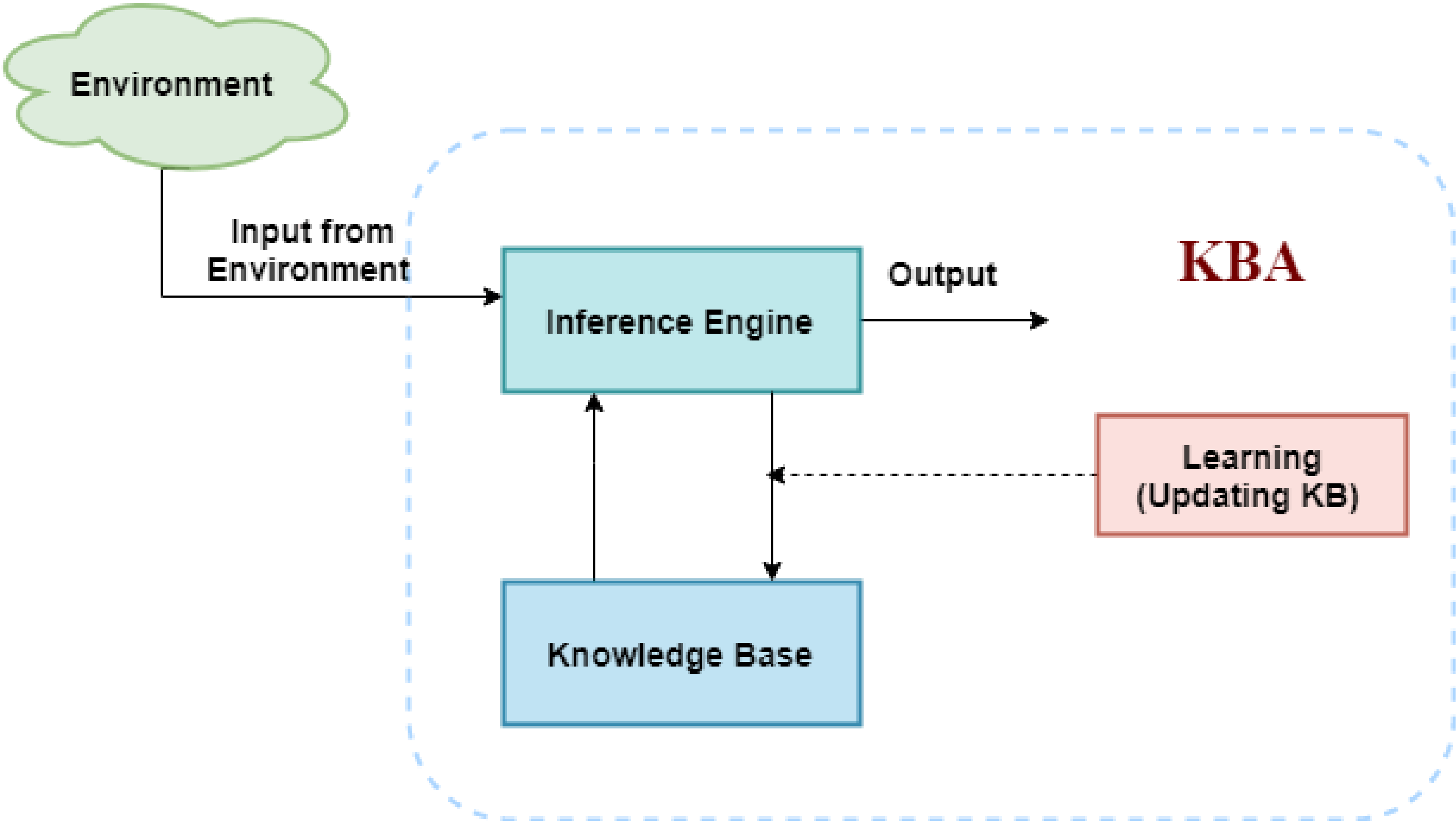
- Inference means deriving new sentences from old.
  - A sentence is a proposition about the world.

  Inference system allows us : -

  - to **add a new sentence** to the knowledge base.
  - **applies logical rules** to the KB to **deduce** new information.
  - **generates new facts** so that an agent can update the KB.
  - works mainly in two rules which are given as:

  Forward chaining-FACT → GOAL

  Backward chaining- GOAL→FACT


Environment
Input from Environment
Inference Engine
Output
KBA
Learning (Updating KB)
Knowledge Base

OPERATIONS PERFORMED BY KBA

Three operations which are performed by KBA in order to show the intelligent behaviour:

- **TELL:** This operation **tells** the *knowledge base*

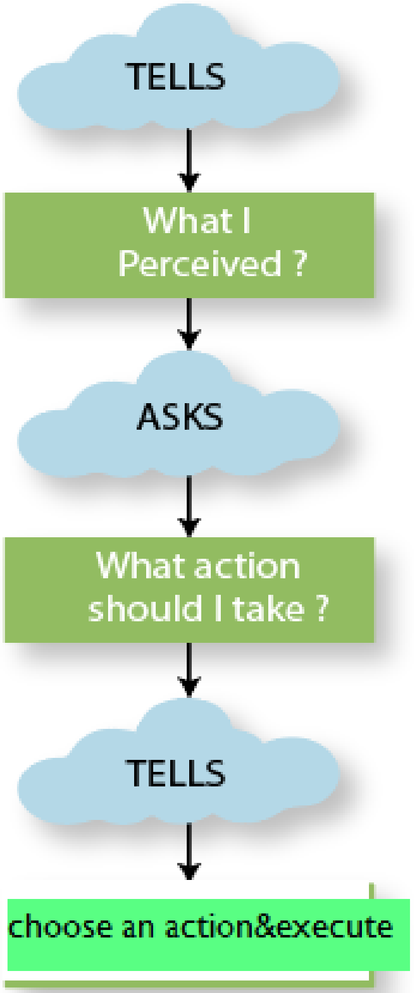
  what it **perceives** from the environment.

- **ASK:** This operation **asks** *the knowledge base*

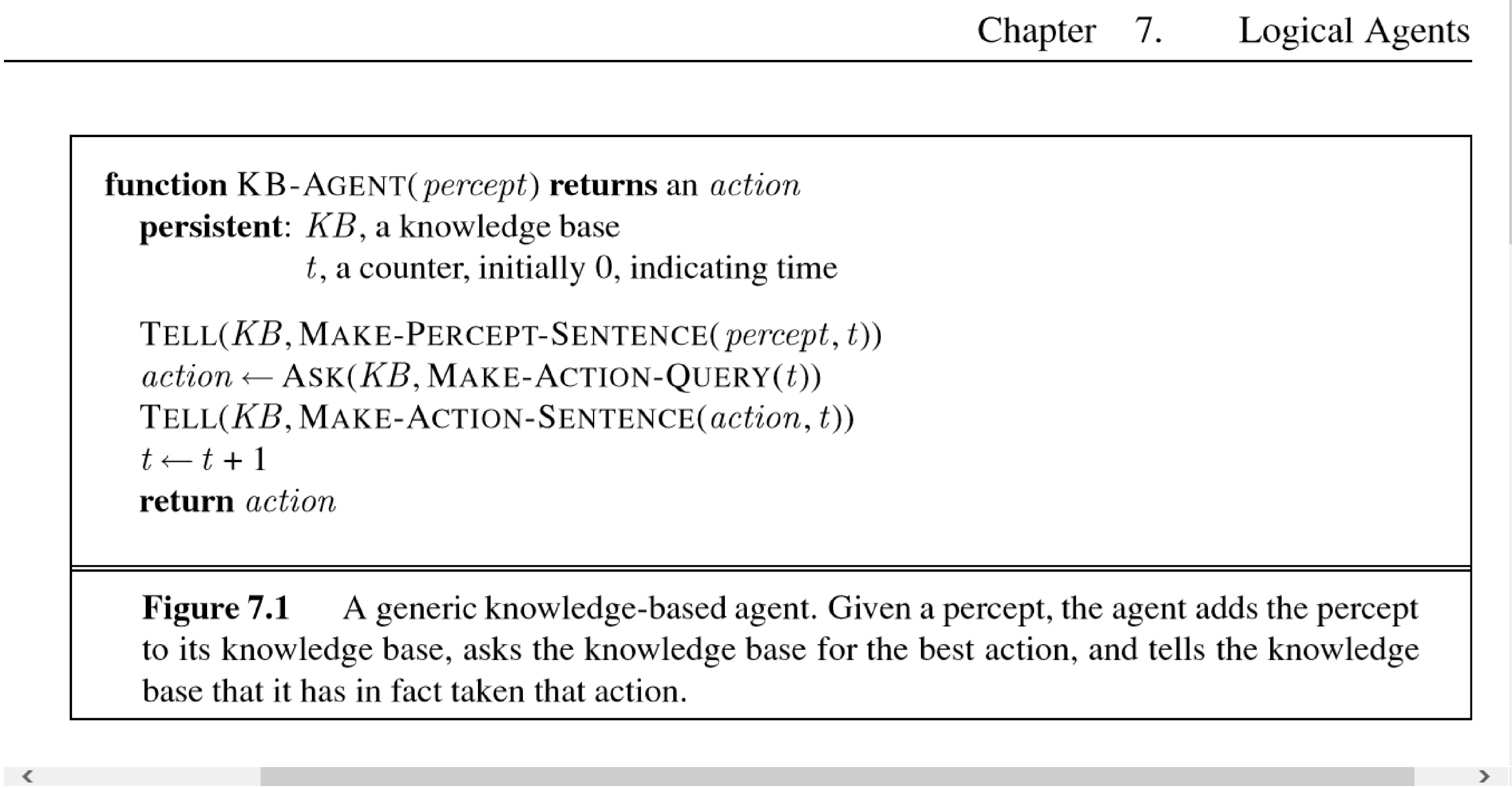
  what **action** it should perform.

  - extensive reasoning may be done about
    - current state of the world,
    - outcomes of possible action sequences, and so on.

- **PERFORM:** It performs the selected action.



Mechanism of an Agent 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-PERCEPT-SENTENCE()**

Returns a sentence which tells the **perceived information by the agent** at a given time.

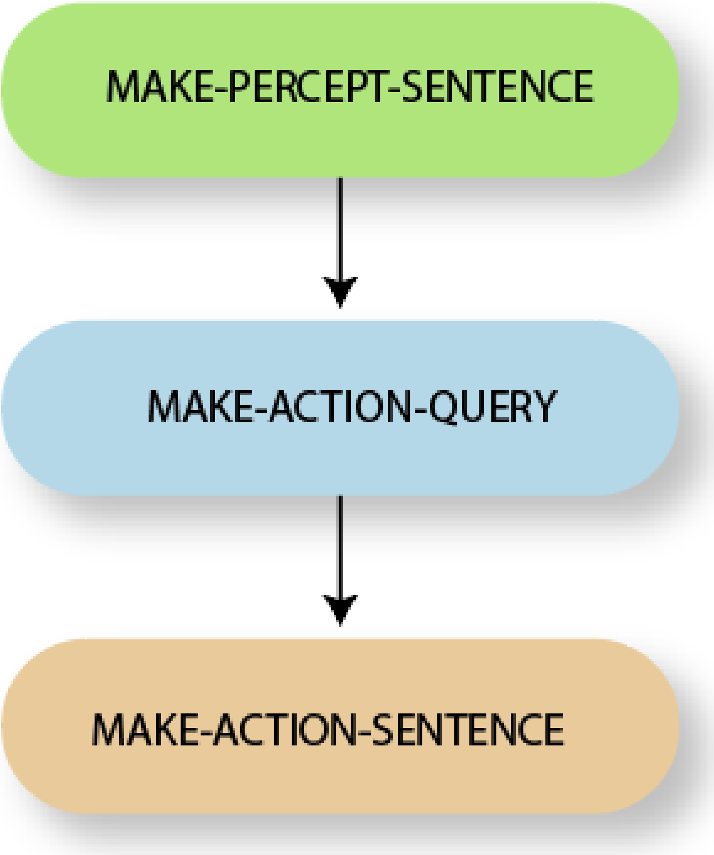
- **MAKE-ACTION-QUERY()**

Returns a sentence which tells **what action the agent** must take at the current time.

- **MAKE-ACTION-SENTENCE()**

Returns a sentence which tells an **action is selected** as well as executed.

The details of the inference mechanisms are hidden inside TELL and ASK



Functions hiding details of Knowledge Representation Language

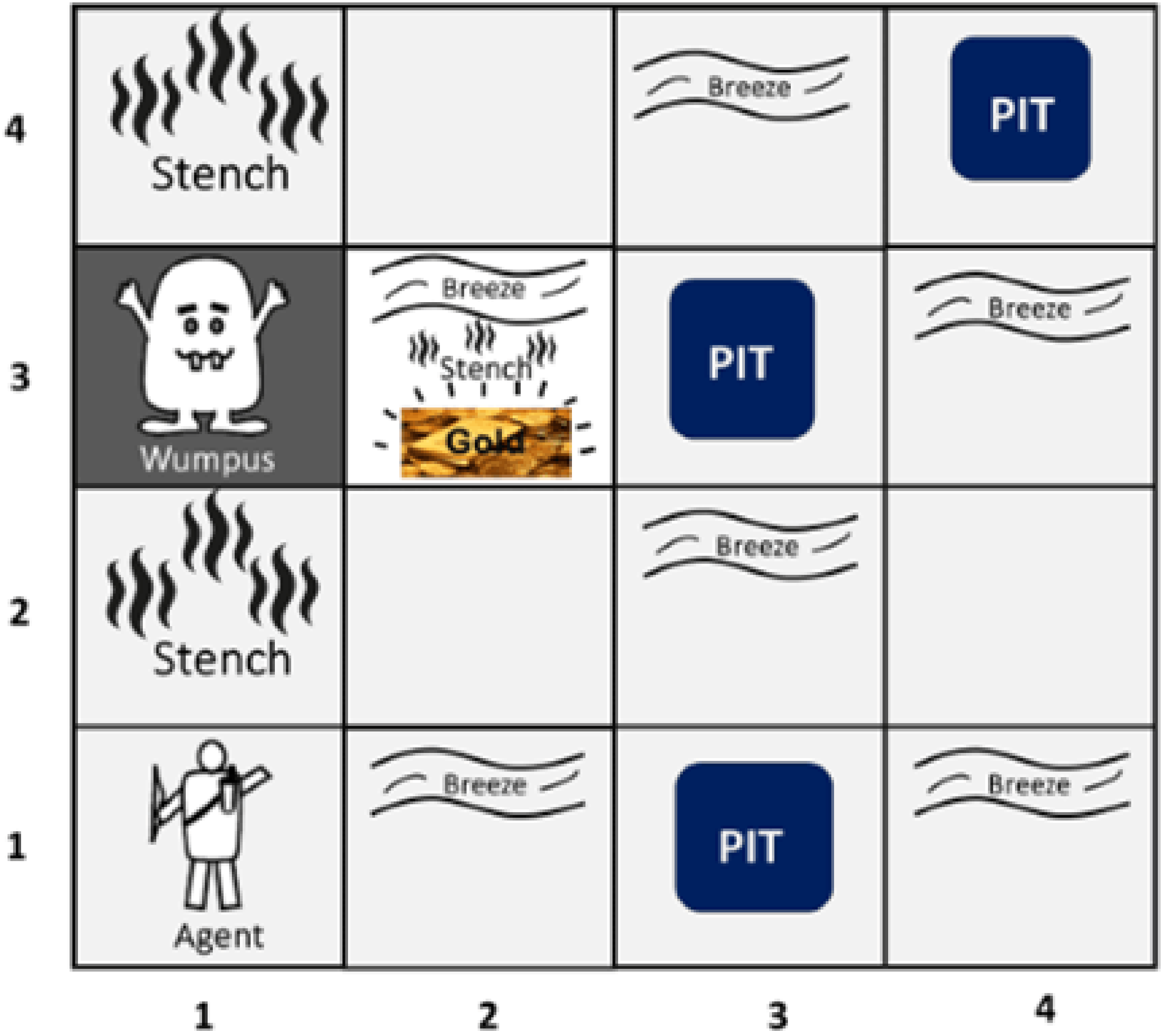
Various levels of knowledge-based agent:

- Knowledge level
- Logical level
- Implementation level

Approaches to designing a knowledge-based agent:

- **Declarative Approach**-Initializing &Writing sentences in KB
- **Procedural Approach**-encoding desired behaviour in KB

Wumpus World Problem



THE WUMPUS WORLD-Describe an environment in which knowledge-based agents can show their worth

The wumpus world is a **cave** consisting of rooms connected by passage ways. Lurking somewhere in the cave is the **terrible 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 contain bottomless pits that will trap anyone who wanders into these rooms (except for the wumpus, which is too big to fall in). The only mitigating feature of this bleak environment is the possibility of finding a heap of gold. Although the wumpus world is rather tame by modern computer game standards, it illustrates some important points about intelligence

It is a simple **world** example to illustrate the **worth** of a “knowledge-based agent” and to “represent” knowledge representation

There are also some components which can help the agent to navigate the cave

- The rooms adjacent to the **Wumpus** room are **smelly**, so that it would have some stench.
- The room adjacent to **PITs** has a **breeze**, so if the agent reaches near to PIT, then he will perceive the breeze.
- There will be **glitter** in the room if and only if the room has **gold**.
- **The Wumpus can be killed by the agent if the agent is facing towards it, and Wumpus will emit a horrible scream which can be heard anywhere in the cave.**

Assumptions

- 4×4 grid, tiles numbered (1,1) to (4,4)
- The agent starts in (1,1),agent is aware of the env (KB)
- The beast Wumpus sits at a random tile, unknown to the agent,
- – a pile of gold sits at another random tile, unknown to the agent,
- – some pits are located at random tiles, unknown to the agent.
- – if the agent enters the tile of the Wumpus, he will be eaten,
- – if the agent enters a pit, he will be trapped.

PEAS description

Performance measure

+1000 for climbing out of the cave with the gold,

–1000 for falling into a pit or being eaten by the wumpus,

–1 for each action taken and –10 for using up the arrow.

The game ends either when the agent dies or when the agent climbs out of the cave.

Environment

A 4×4 grid of rooms.

- The agent always starts in the square labeled [1,1], facing to the right.
- The locations of the gold and the wumpus are chosen randomly, with a uniform distribution, from the squares other than the start square.
  - » In addition, each square other than the start can be a pit, with probability 0.2.

Actuators

- Left turn by 90 degree
- Right turn by 90 degree
- Move forward
- Grab
- Release
- Shoot.

Sensors

- The agent has **five sensors**, each of which gives a single bit of information
  - In the square containing the wumpus and in the directly (not diagonally) adjacent squares, the agent will perceive a **Stench**.
  - In the squares directly adjacent to a pit, the agent will perceive a **Breeze**.

    In the square where the **gold** is, the agent will perceive a **Glitter**.
  - When an agent walks into a **wall**, it will perceive a **Bump**.
  - When the **wumpus is killed**, it emits a woeful **Scream** that can be perceived anywhere in the cave.

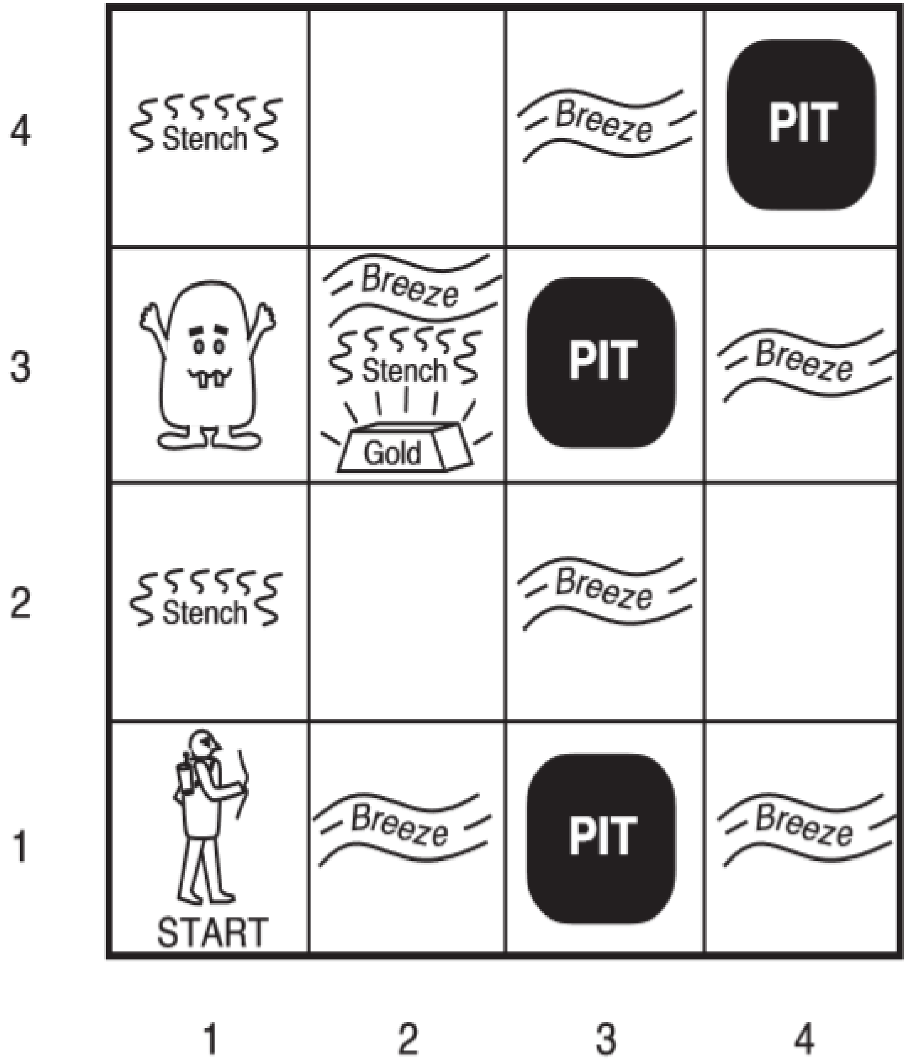


Figure 7.2 A typical wumpus world. The agent is in the bottom left corner, facing righ

- Moves allowed” L , R ,U ,D , G(grab)

Start (1,1)→Stench (1,2) & **Breeze (2,1)**

(2,1)[breeze]

(3,1) [pit] ?

(2,2) [pit]?

Safely moves to (1,1)

Start from (1,2)Stench → **(2,2)** & (1,3) ? W

Conclusion/Inference : - (2,2) IS SAFE !!

Start from (2,2) → (2,3) & Breeze (3,2)

Move to (3,2) : Pit at (3,3) & (3,1)

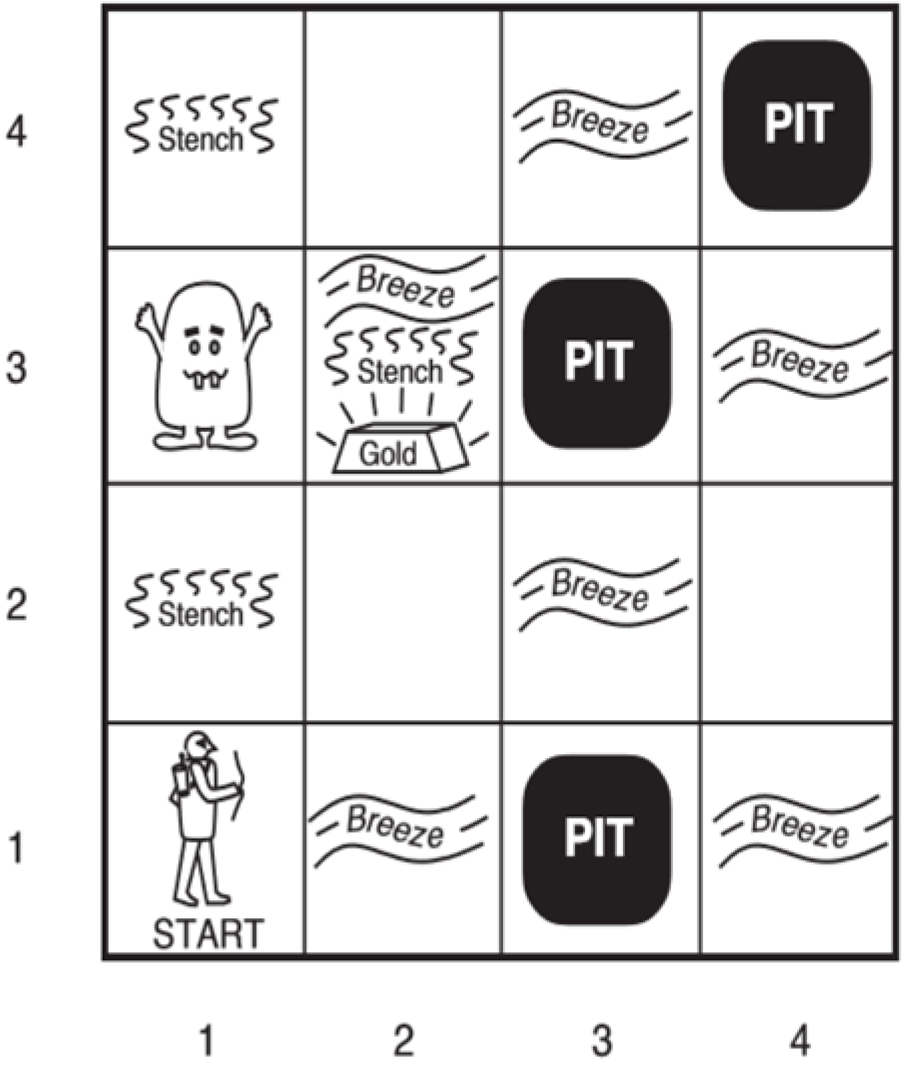
Move back to (2,2) & explore (2,3)

(2,3) → PERCEIVES GOLD !!!!!

Return via the same path agent went by

Uses ARROW→(3,3)Pit , NO SCREAM

(1,3) **Wumpus** , SCREAM



The Wumpus world Properties

- **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.
- **One agent:** The environment is a single agent as we have one agent only and Wumpus is not considered as an agent

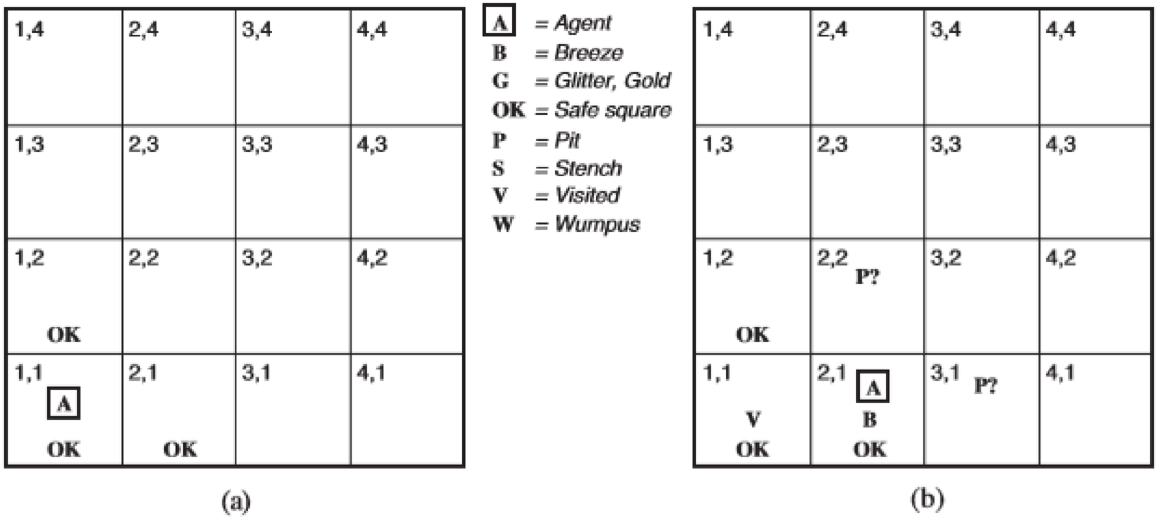


Figure 7.3 The first step taken by the agent in the wumpus world. (a) The initial situation, after percept [*None*, *None*, *None*, *None*, *None*]. (b) After one move, with percept [*None*, *Breeze*, *None*, *None*, *None*].

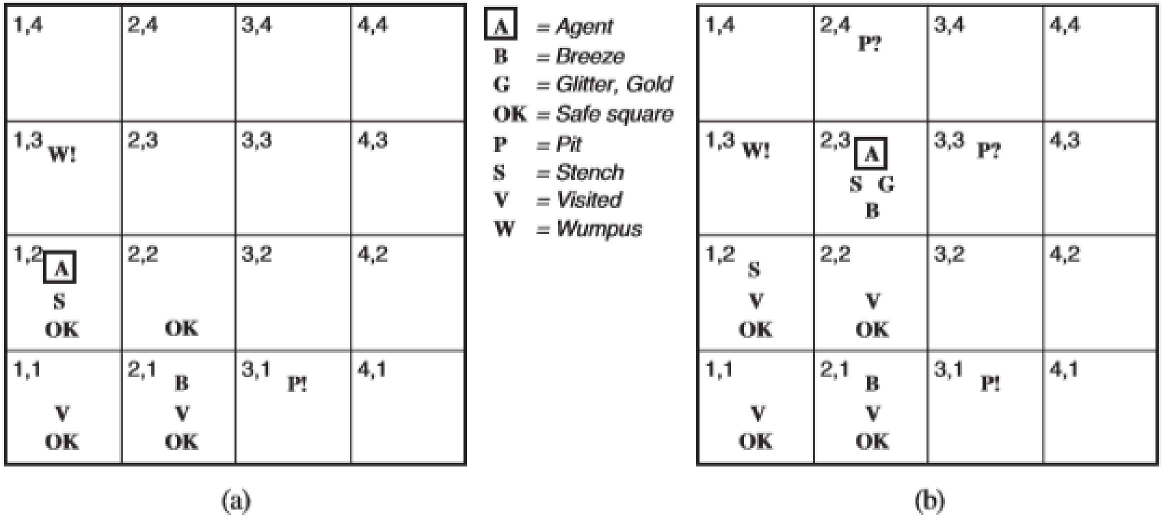


Figure 7.4 Two later stages in the progress of the agent. (a) After the third move, with percept [*Stench*, *None*, *None*, *None*, *None*]. (b) After the fifth move, with percept [*Stench*, *Breeze*, *Glitter*, *None*, *None*].

Logic in general

Logics are formal languages for representing information
such that conclusions can be drawn

Syntax defines the sentences in the language

Semantics define the "meaning" of sentences;
i.e., define truth of a sentence in a world

E.g., the language of arithmetic

$x + 2 \geq y$ is a sentence; $x2 + y >$ is not a sentence

$x + 2 \geq y$ is true iff the number $x + 2$ is no less than the number y

$x + 2 \geq y$ is true in a world where $x = 7,\ y = 1$
$x + 2 \geq y$ is false in a world where $x = 0,\ y = 6$

Propositional Logic (PL)

- 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.

Example:

It is Sunday.(True)

The Sun rises from West (False proposition)

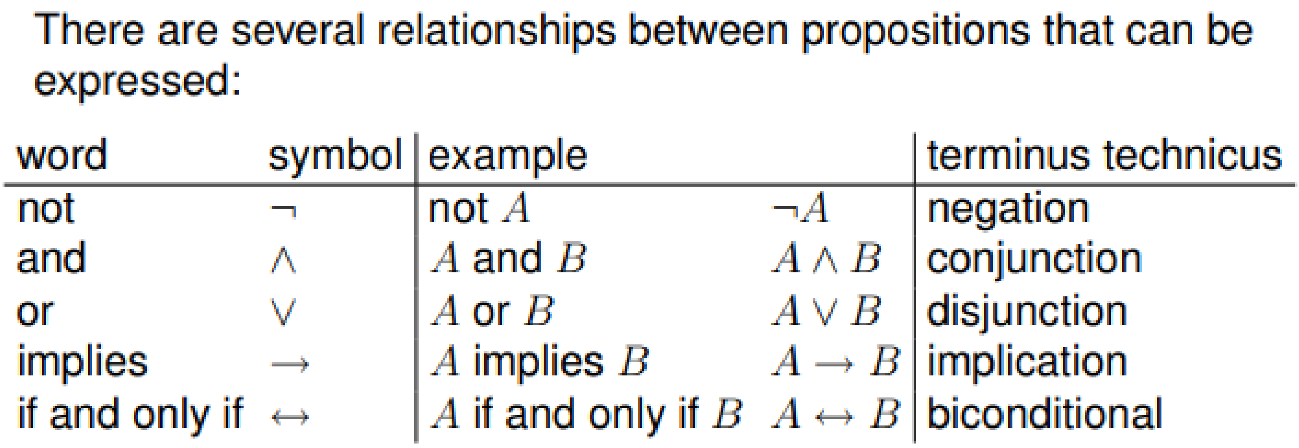
3+3= 7(False proposition)

5 is a prime number.(True)

Some basic facts about PL.

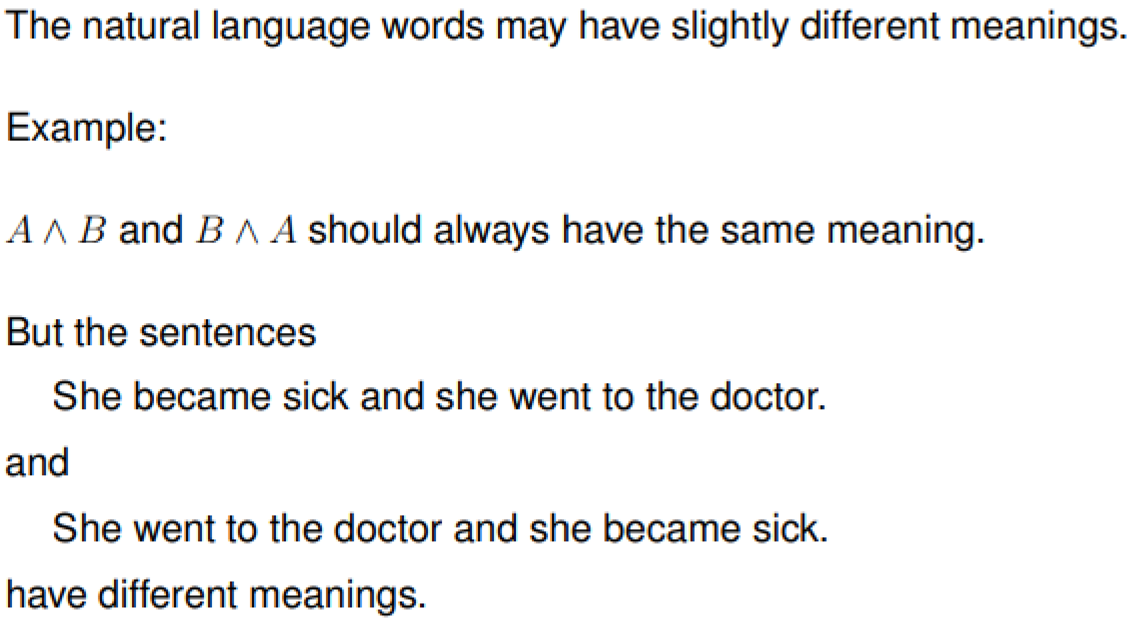
- Boolean logic - works on 0 and 1.
- Use symbolic variables to represent the logic (representing a proposition such A, B, C, P, Q, R, etc.)
- Propositions can be either true or false, but it cannot be both.
- Propositional logic consists of an object, relations or function, and **logical connectives** (logical operators)
- The propositions and connectives are the basic elements of the propositional logic.
  - Connectives is a logical operator which connects two sentences.
- A proposition formula which is always true is called **tautology**, and it is also called a valid sentence.
- A proposition formula which is always false is called **Contradiction**.
- Statements which are questions, commands, or opinions are not propositions such as "**Where is John**", "**How are you**", "**What is your name**", are **NOT PROPOSITIONS.**

PROPOSTIONAL LOGIC-”**REVISION**”



There are several relationships between propositions that can be expressed:

word	symbol	example		terminus technicus
not	$\neg$	not A	$\neg A$	negation
and	$\wedge$	A and B	$A \wedge B$	conjunction
or	$\vee$	A or B	$A \vee B$	disjunction
implies	$\rightarrow$	A implies B	$A \rightarrow B$	implication
if and only if	$\leftrightarrow$	A if and only if B	$A \leftrightarrow B$	biconditional



The natural language words may have slightly different meanings.

Example:

$A \wedge B$ and $B \wedge A$ should always have the same meaning.

But the sentences

She became sick and she went to the doctor.

and

She went to the doctor and she became sick.

have different meanings.

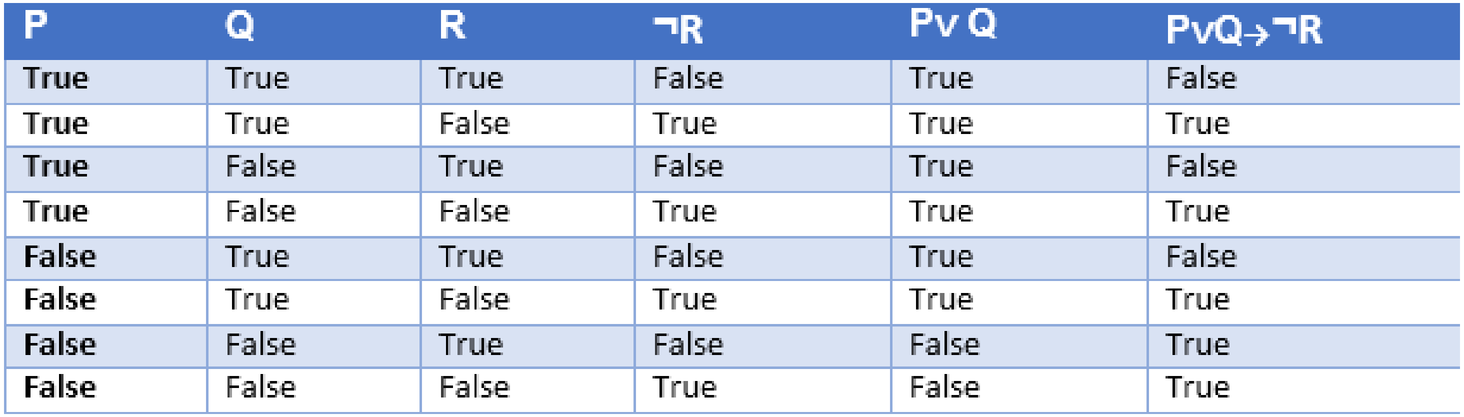
- FORMAL PROOFS
- NORMAL FORMS

Left as an exercise to students

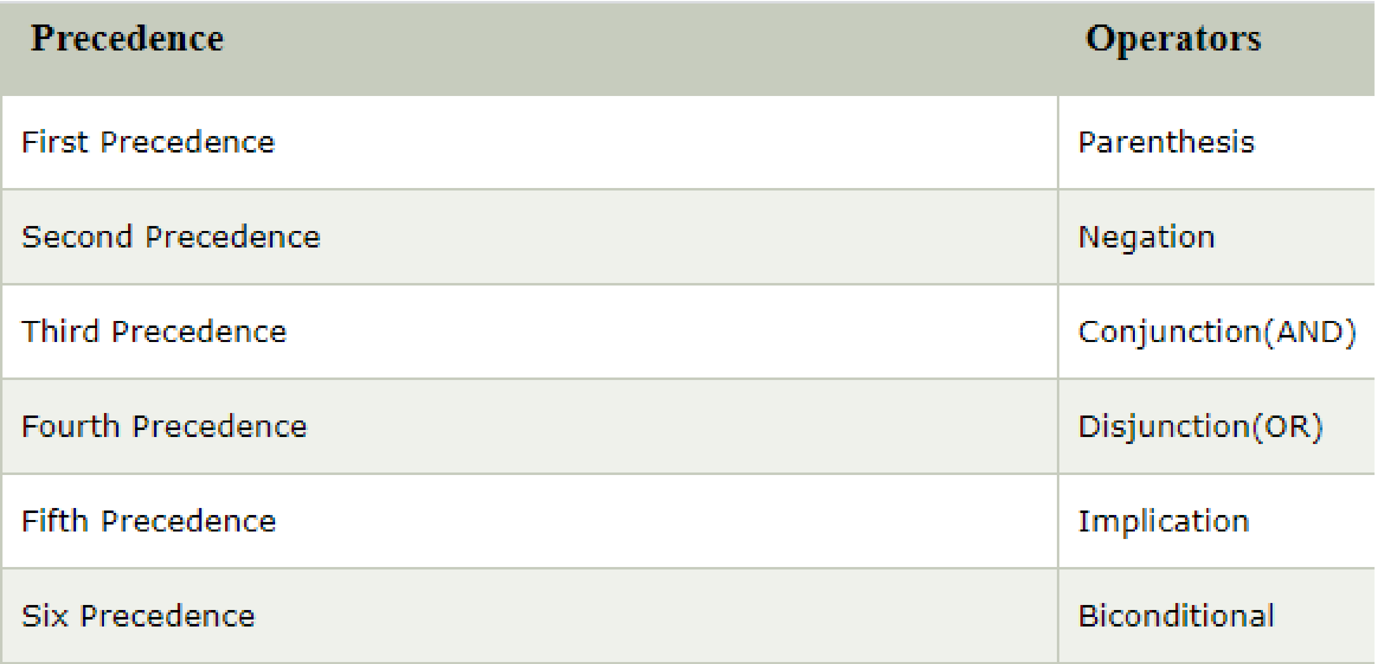
REVISION

- **Negation:** A sentence such as ¬ P is called negation of P.
  - A literal can be either Positive literal or negative literal.
- **Conjunction:** A sentence which has **∧** connective such as, **P ∧ Q** is called a conjunction.
  - **Example:** Rohan is intelligent and hardworking. It can be written as,
    P= Rohan is intelligent,
    Q= Rohan is hardworking. → P∧ Q.
- **Disjunction:** A sentence which has ∨ connective, such as **P ∨ Q**. is called disjunction, where P and Q are the propositions.
  - **Example: "Ritika is a doctor or Engineer"**,
    Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as **P ∨ Q**.
- **Implication:** A sentence such as P → Q, is called an implication.
  - Implications are also known as if-then rules. It can be represented as
    If it is raining, then the street is wet.
    Let P= It is raining, and Q= Street is wet, so it is represented as P → Q
- **Biconditional:** A sentence such as **P⇔ Q is a Biconditional sentence, example If I am breathing, then I am alive**

  P= I am breathing, Q= I am alive, it can be represented as P ⇔ Q.



P	Q	R	¬R	Pv Q	PvQ→¬R
True	True	True	False	True	False
True	True	False	True	True	True
True	False	True	False	True	False
True	False	False	True	True	True
False	True	True	False	True	False
False	True	False	True	True	True
False	False	True	False	False	True
False	False	False	True	False	True



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

Properties of Operators

- **Commutativity:**
  - P∧ Q= Q ∧ P, or
  - P ∨ Q = Q ∨ P.
- **Associativity:**
  - (P ∧ Q) ∧ R= P ∧ (Q ∧ R),
  - (P ∨ Q) ∨ R= P ∨ (Q ∨ R)
- **Identity element:**
  - P ∧ True = P,
  - P ∨ True= True.
- **Distributive:**
  - P∧ (Q ∨ R) = (P ∧ Q) ∨ (P ∧ R).
  - P ∨ (Q ∧ R) = (P ∨ Q) ∧ (P ∨ R).
- **DE Morgan's Law:**
  - ¬ (P ∧ Q) = (¬P) ∨ (¬Q)
  - ¬ (P ∨ Q) = (¬ P) ∧ (¬Q).
- **Double-negation elimination:**
  - ¬ (¬P) = P.

Inference Rules

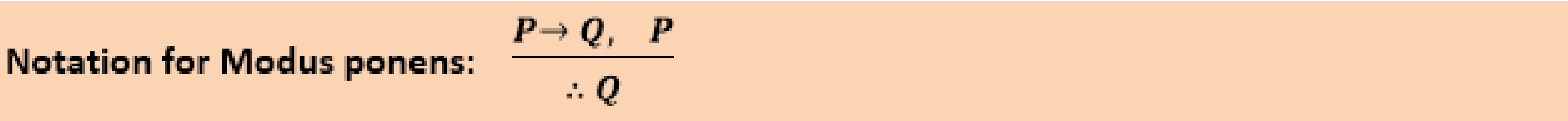
- **Statement-1**: "If I am sleepy then I go to bed"

  P→ Q

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

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

  Thus, if P→ Q is true and P is true then Q will be true.



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

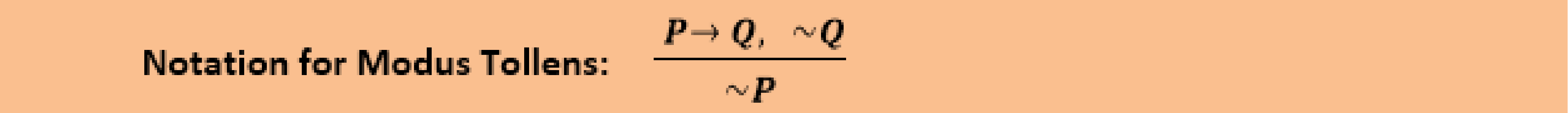
- **Statement-1:** "If I am sleepy then I go to bed" ==>

  P→ Q

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

  Statement-3: Which infers that "**I am not sleepy**"

  => ~P



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

- **Statement-1:** If you have my home key then you can unlock my home. **$P \rightarrow Q$**

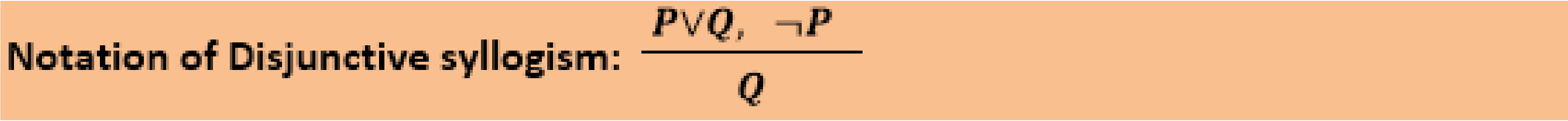
  Statement-2: If you can unlock my home then you can take my money. **$Q \rightarrow R$**

  Conclusion: If you have my home key then you can take my money. **$P \rightarrow R$**

- **Statement-1:** Today is Sunday or Monday. ==>P∨Q

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

  Conclusion: Today is Monday. ==> Q



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

Limitations of Propositional logic:

Cannot represent relations like ALL, some, or none with propositional logic.

Example:

- **All the girls are intelligent.**
- **Some apples are sweet.**

- Propositional logic has limited expressive power.
  - In propositional logic, we cannot describe statements in terms of their properties or logical relationships.

"Some humans are intelligent", or

"Sachin likes cricket."

- To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as **FIRST-ORDER LOGIC.**

Knowledge-base for Wumpus world

- Atomic proposition variable for Wumpus world:
  - Let $\mathbf{P}_{\mathbf{i,j}}$ be true if there is a **P**it in the room [i, j].
  - Let $\mathbf{B}_{\mathbf{i,j}}$ be true if agent perceives **B**reeze in [i, j]
  - Let $\mathbf{W}_{\mathbf{i,j}}$ be true if there is **W**umpus in the square[i, j].
  - Let $\mathbf{S}_{\mathbf{i,j}}$ be true if agent perceives **S**tench in the square [i, j].

Some Propositional Rules for the wumpus world:

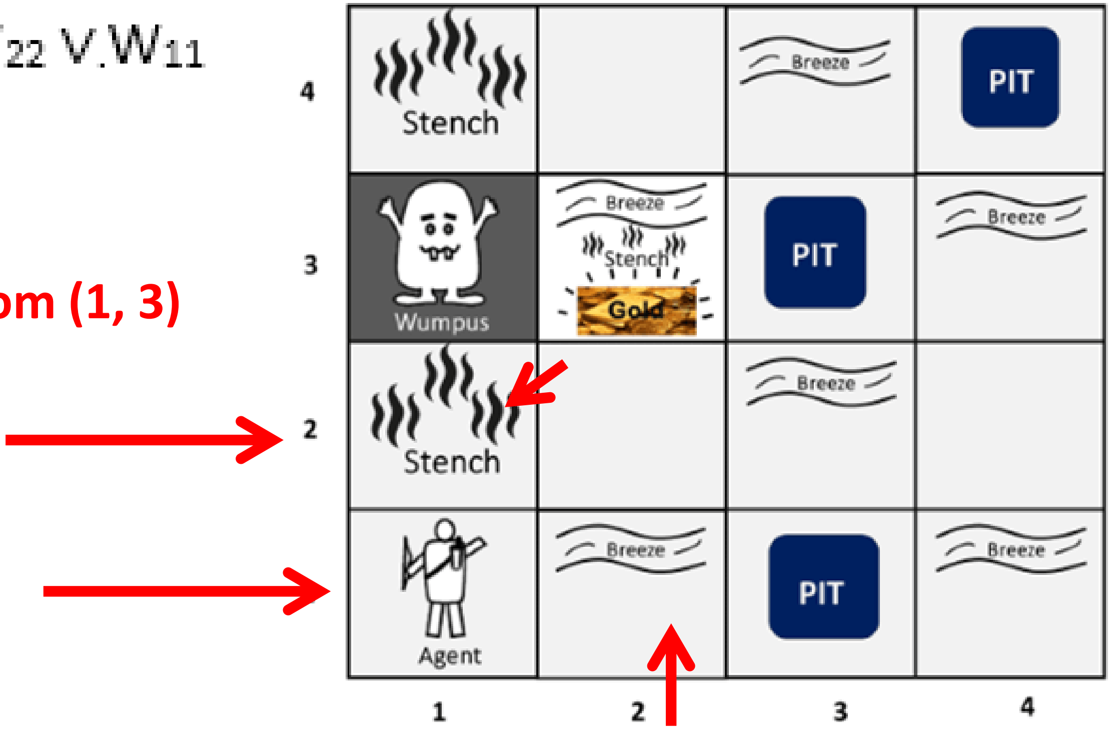
(R1) $\neg S_{11} \Rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

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

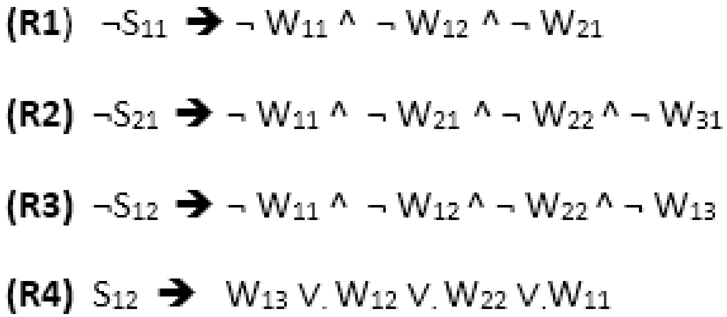
(R3) $\neg S_{12} \Rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{22} \wedge \neg W_{13}$

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

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



1. Apply Modus Ponens with $\neg S_{11}$ and R1
2. Apply Modus Ponens to $\neg S_{21}$, and R2
3. Apply Modus Ponens to S_{12} and R4
4. Apply Unit resolution on $W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$ and $\neg W_{11}$
5. Apply Unit resolution on $W_{13} \vee W_{12} \vee W_{22}$ and $\neg W_{22}$
6. Apply Unit Resolution on $W_{13} \vee W_{12}$ and $\neg W_{12}$

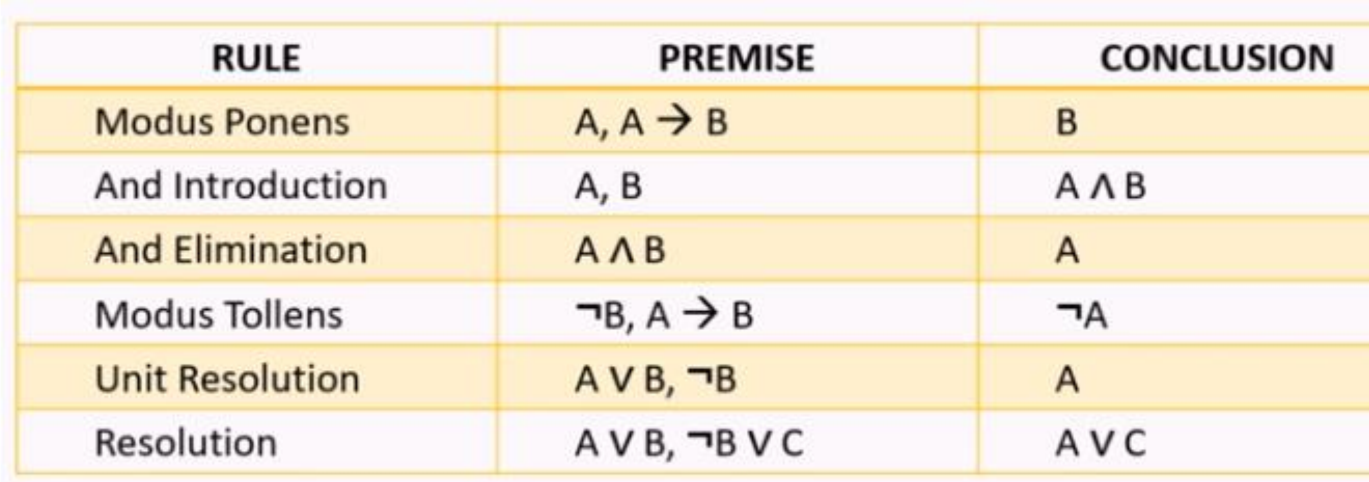


(R1) $\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

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



RULE	PREMISE	CONCLUSION
Modus Ponens	A, A → B	B
And Introduction	A, B	A ∧ B
And Elimination	A ∧ B	A
Modus Tollens	¬B, A → B	¬A
Unit Resolution	A ∨ B, ¬B	A
Resolution	A ∨ B, ¬B ∨ C	A ∨ C

Proving W13

- Apply MP with ~S11 and R1:

  ~W11 ^ ~W12 ^ ~W21

- Apply And-Elimination to this we get 3 sentences:

  ~W11, ~W12, ~W21

- Apply MP to ~S21 and R2, then applying And-elimination:

  ~W22, ~W21, ~W31

- Apply MP to S12 and R4 we obtain:

  W13 v W12 v W22 v W11

- Apply Unit resolution on (W13 v W12 v W22 v W11) and ~W11

  W13 v W12 v W22

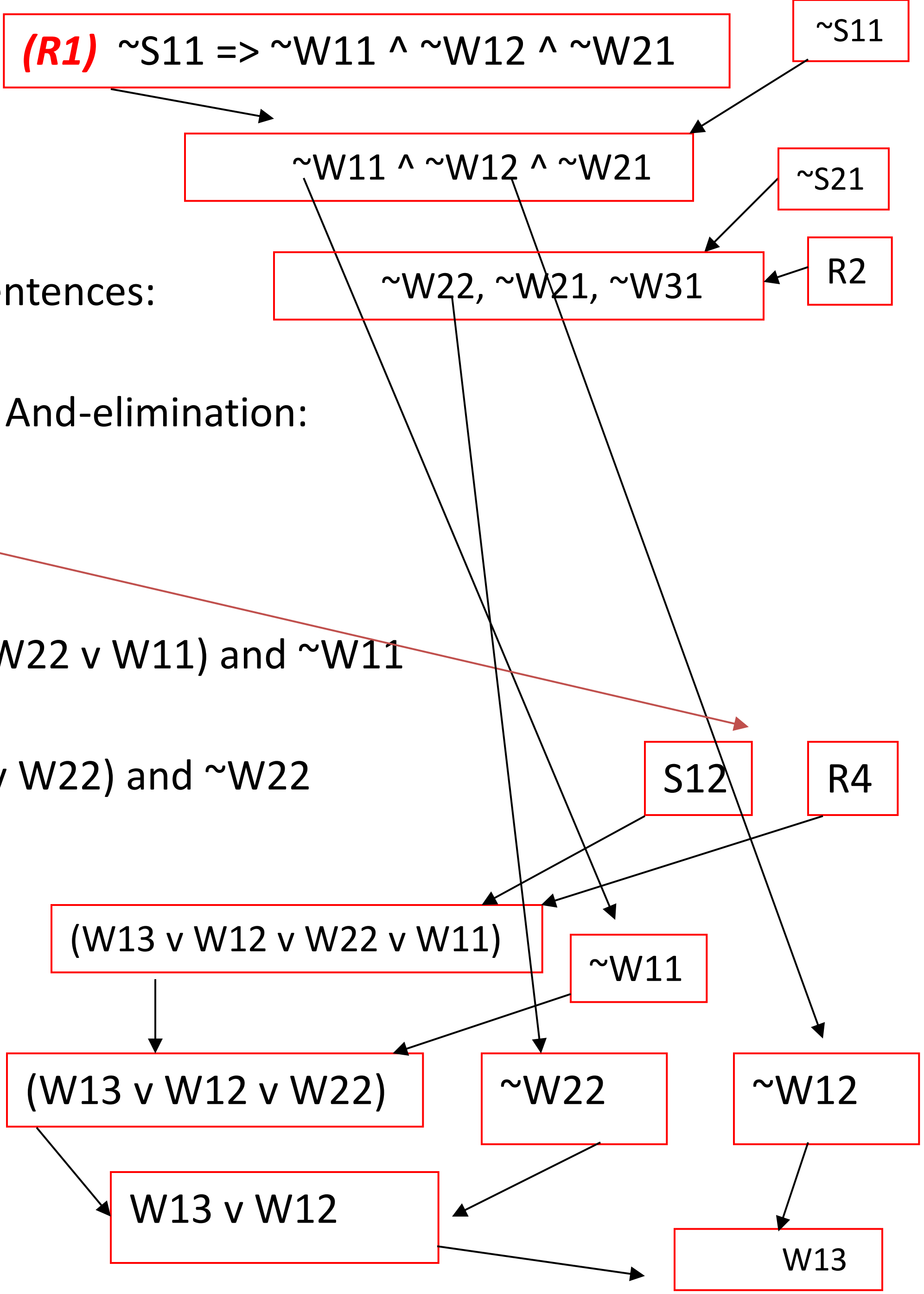
- Apply Unit Resolution with (W13 v W12 v W22) and ~W22

  W13 v W12

- Apply UR with (W13 v W12) and ~W12

  W13

- QED



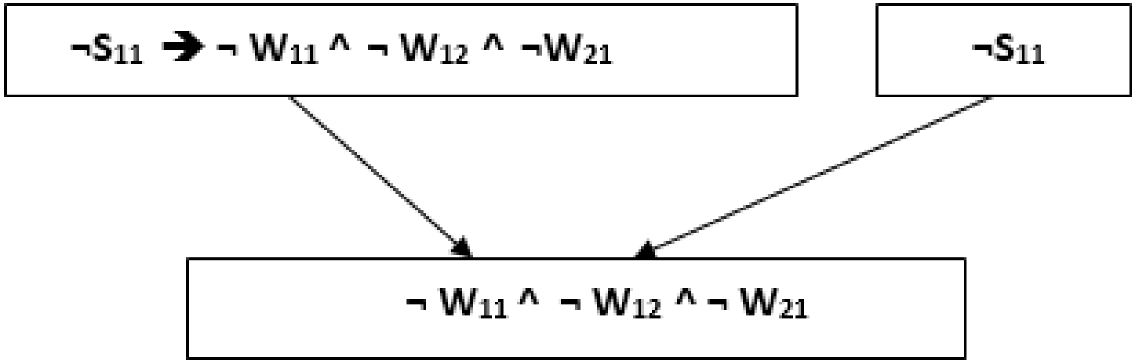
Prove that Wumpus is in the room (1, 3)

Notation for Modus ponens: $\dfrac{P \rightarrow Q, \quad P}{\therefore Q}$

Apply Modus Ponens with ¬S11 and R1:

- We will firstly apply MP rule with R1 which is

$\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$, and $\boldsymbol{\neg S_{11}}$ which will give the output $\neg W_{11} \wedge W_{12} \wedge W_{12}$.



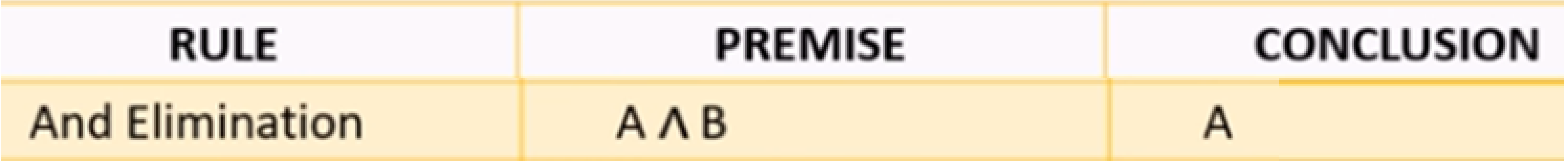
$\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

$\neg S_{11}$

$\neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

Apply And-Elimination Rule:

- After applying And-elimination rule to $\neg W_{11} \land \neg W_{12} \land \neg W_{21}$, we will get three statements:
  $\neg W_{11}$, $\neg W_{12}$, and $\neg W_{21}$.



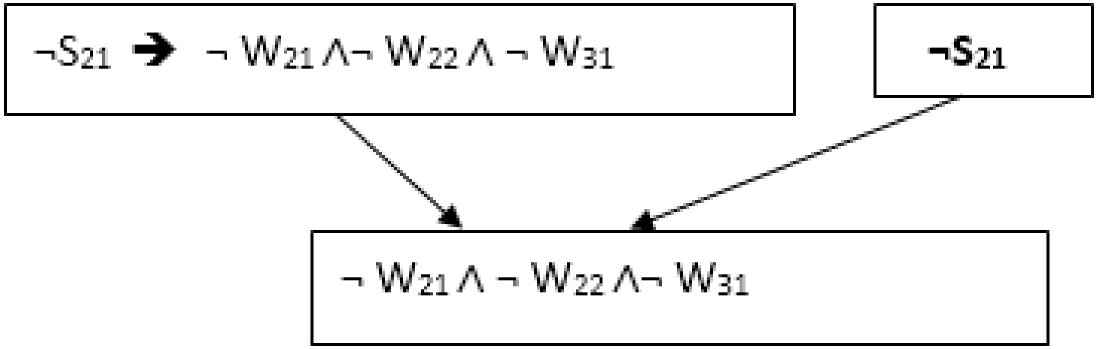
RULE	PREMISE	CONCLUSION
And Elimination	$A \land B$	A

Apply Modus Ponens to $\neg S_{21}$, and R2:

- Now we will apply Modus Ponens to $\neg S_{21}$ and R2 which is

$\neg S_{21} \rightarrow \neg W_{21} \land \neg W_{22} \land \neg W_{31}$, which will give the

Output as **$\neg W_{21} \land \neg W_{22} \land \neg W_{31}$**


$\neg S_{21} \rightarrow \neg W_{21} \land \neg W_{22} \land \neg W_{31}$
$\neg S_{21}$
$\neg W_{21} \land \neg W_{22} \land \neg W_{31}$

- **Apply And -Elimination rule:**

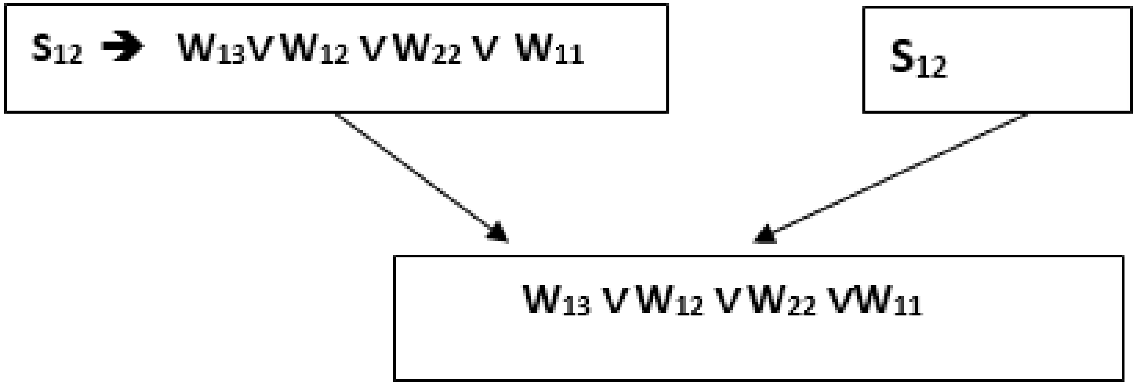
Now again apply And-elimination rule to

$\neg W_{21} \land \neg W_{22} \land \neg W_{31}$, We will get three statements:
$\neg W_{21}$, $\neg W_{22}$, and $\neg W_{31}$.

- **Apply MP to S_{12} and R4:**

Apply Modus Ponens to **S_{12}** and **R_4** which is

$S_{12} \rightarrow W_{13} \lor W_{12} \lor W_{22} \lor .W_{11}$, we will get the output as **$W_{13} \lor W_{12} \lor W_{22} \lor .W_{11}$**.



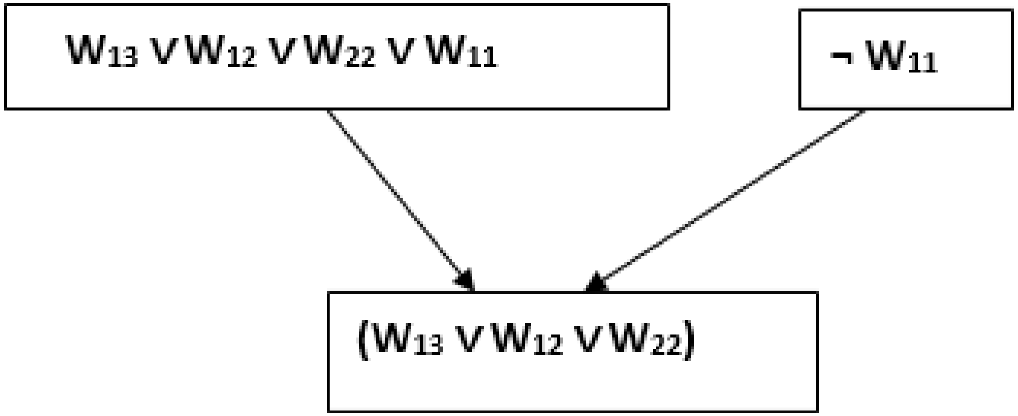
$S_{12} \rightarrow W_{13} \lor W_{12} \lor W_{22} \lor W_{11}$

S_{12}

$W_{13} \lor W_{12} \lor W_{22} \lor W_{11}$

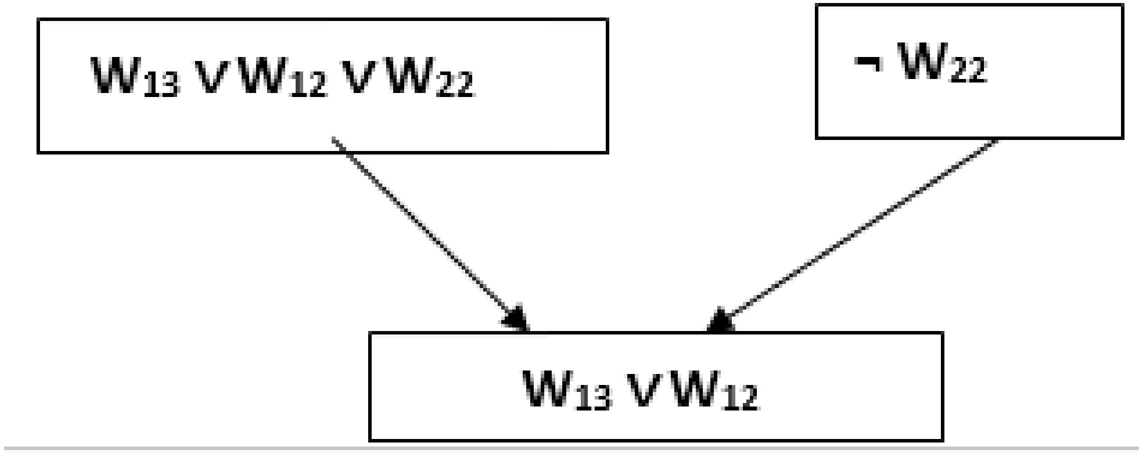
- **Apply Unit resolution on W_{13} ∨ W_{12} ∨ W_{22} ∨W_{11} and ¬ W_{11} :**

After applying Unit resolution formula on W_{13} ∨ W_{12} ∨ W_{22} ∨W_{11} and ¬ W_{11} we will get W_{13} ∨ W_{12} ∨ W_{22}.

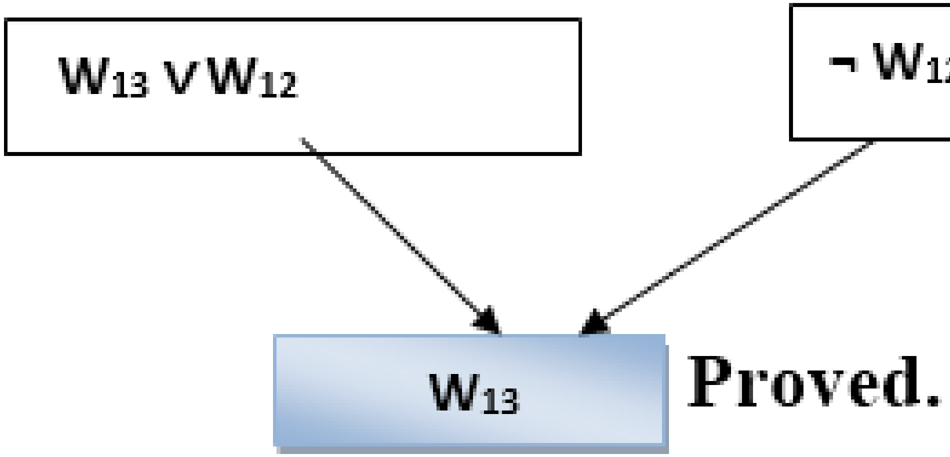

W_{13} ∨ W_{12} ∨ W_{22} ∨ W_{11}
¬ W_{11}
(W_{13} ∨ W_{12} ∨ W_{22})

- **Apply Unit resolution on $W_{13} \lor W_{12} \lor W_{22}$ and $\neg W_{22}$:**

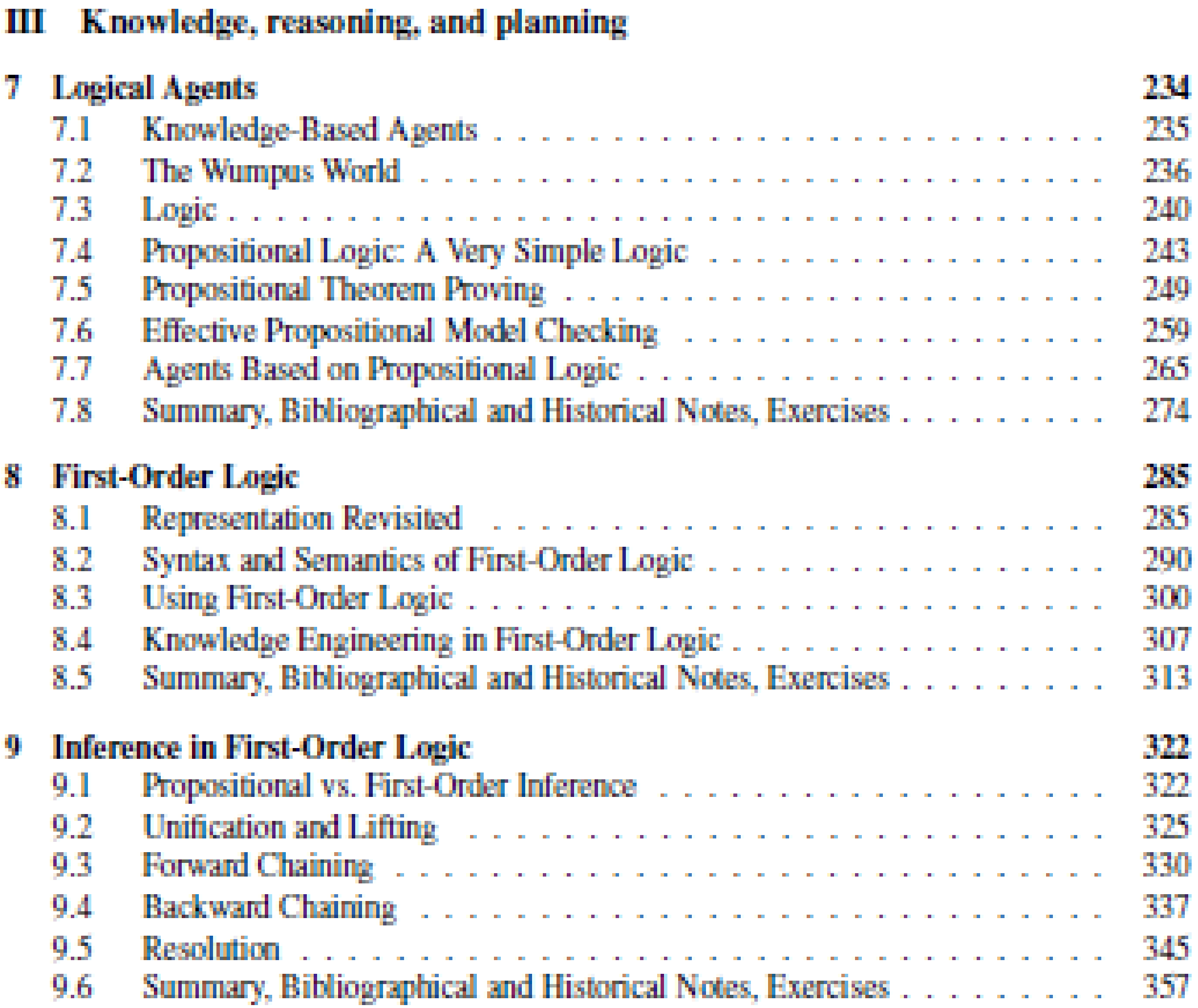
After applying Unit resolution on $\mathbf{W_{13}} \lor \mathbf{W_{12}} \lor \mathbf{W_{22}}$, and $\neg\mathbf{W_{22}}$, we will get $\mathbf{W_{13}} \lor \mathbf{W_{12}}$ as output.



- **Apply Unit Resolution on W_{13} ∨ W_{12} and ¬ W_{12} :**
- After Applying Unit resolution on **W_{13} ∨ W_{12} and ¬ W_{12}**, we will get **W_{13}** as an output, hence it is proved that the Wumpus is in the room [1, 3].



FOL-First Order Logic



III Knowledge, reasoning, and planning

Ex 1 : Parrots are green.

Parrots(X)→Green(X)

Ex 2: All kids like toys.

like (kids , toys).

Ex 3: Some kids like toys.

like (kids , toys).

Quantifiers! FOL

FIRST-ORDER LOGIC (FOL)

- Knowledge representation in artificial intelligence.
  - Extension to propositional logic.
- Represent natural language statements in a concise way.
  - (**Predicate logic or First-order predicate logic**)
- Powerful language
- First-order logic (like natural language) assumes:
  - **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, wumpus, ...
  - **Relations: It can be unary relation such as:** red, round, is adjacent, **or n-any relation such as:** the sister of, brother of, has color, comes between……….
  - **Function:** Father of, best friend, third inning of, end of,……..

Syntax of FOL

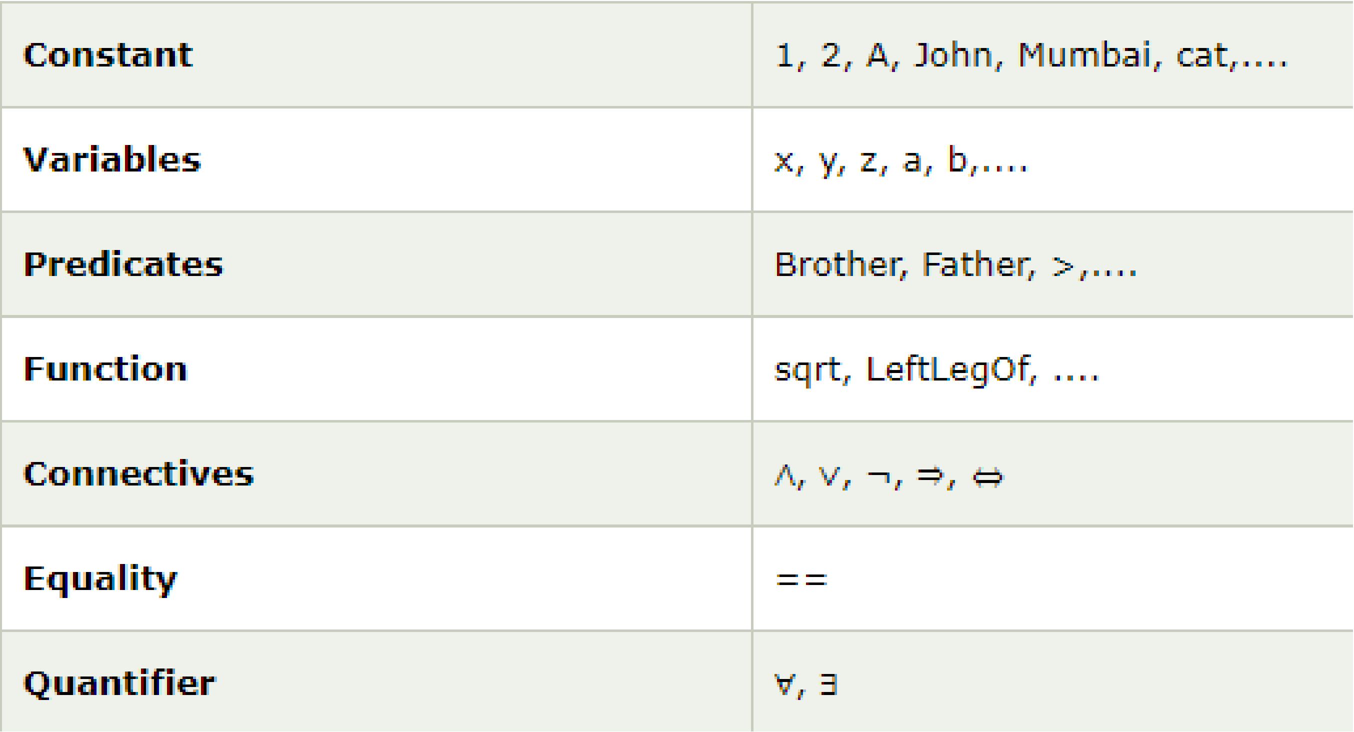
- User defines these pRoytives:
  - **Constant symbols** (i.e., the "individuals" in the world) E.g., Divya, 3
  - **Function symbols** (mapping individuals to individuals) E.g., father-of(Divya) = John, color-of(Sky) = Blue
  - **Predicate symbols** (mapping from individuals to truth values) E.g., greater(5,3), green(Grass), color(Grass, Green)

Syntax of FOL

FOL supplies these pRoytives

- **Variable symbols**. E.g., x, y
- **Connectives**. Same as in PL: not (~), and (^), or (v), implies (=>), if and only if (<=>)
- **Quantifiers**: Universal (∀) and Existential (∃)

Basic elements of FOL syntax



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

Semantics of FOL

Lets understand with an example,

Consider the sentence “Elephants are big”. There are many ways to represent this sentence.

HasSize(Elephant, Big)

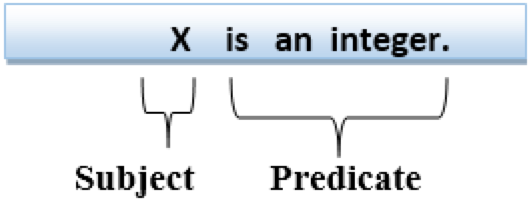
SizeOF(Elephant)= Big

Atomic sentences

- 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**.
- Representation: - **Predicate (term1, term2, ..., term n)**.
- **Example:**
  - **Ravi and Ajay are brothers: => Brothers(Ravi, Ajay)**

Complex Sentences

- Complex sentences are made by combining atomic sentences using connectives.
- **First-order logic statements can be divided into two parts:**
  - **Subject:** Subject is the main part of the statement.
  - **Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.
- **Consider the statement: "x is an integer."**, it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.



Quantifiers in First-order logic:

Universal Quantifier, (for all, everyone, everything)

- **For all x**
- **For each x**
- **For every x**

$\forall x \; man(x) \rightarrow drink\ (x, coffee).$

It will be read as: There are all x where x is a man who drink coffee.

Existential quantifier, (for some, at least one)

- **There exists a 'x.'**
- **For some 'x.'**
- **For at least one 'x.‘**

$\exists x: kid(x) \wedge intelligent(x)$

It will be read as: There are some x where x is a kid who is intelligent.

Universal quantification

$\forall \langle variables \rangle \; \langle sentence \rangle$

Everyone at Berkeley is smart:
$\forall x \;\; At(x, Berkeley) \;\Rightarrow\; Smart(x)$

$\forall x \;\; P$ is true in a model m iff P is true with x being **each** possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of P

$$\begin{aligned} & (At(KingJohn, Berkeley) \;\Rightarrow\; Smart(KingJohn)) \\ \wedge\; & (At(Richard, Berkeley) \;\Rightarrow\; Smart(Richard)) \\ \wedge\; & (At(Berkeley, Berkeley) \;\Rightarrow\; Smart(Berkeley)) \\ \wedge\; & \ldots \end{aligned}$$

Existential quantification

$\exists \langle variables \rangle \ \ \langle sentence \rangle$

Someone at Stanford is smart:
$\exists x \ \ At(x, Stanford) \land Smart(x)$

$\exists x \ \ P$ is true in a model m iff P is true with x being **some** possible object in the model

Roughly speaking, equivalent to the disjunction of instantiations of P

$$
\begin{array}{ll}
 & (At(KingJohn, Stanford) \land Smart(KingJohn)) \\
\lor & (At(Richard, Stanford) \land Smart(Richard)) \\
\lor & (At(Stanford, Stanford) \land Smart(Stanford)) \\
\lor & \ldots
\end{array}
$$

Points to remember:

- **The main connective for universal quantifier $\forall$ is implication $\rightarrow$**
- **The main connective for existential quantifier $\exists$ is and $\land$**

Properties of Quantifiers:

- In universal quantifier, $\forall x \forall y$ is similar to $\forall y \forall x$.
- In Existential quantifier, $\exists x \exists y$ is similar to $\exists y \exists x$.
- $\exists x \forall y$ is not similar to $\forall y \exists x$.

Example FOL using quantifier:

- **1. All birds fly.**
- In this question the predicate is "**fly(bird)**."

  And since there are all birds who fly so it will be represented as follows.

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

- **2. Every man respects his parent.**
- In this question, the predicate is "**respect(x, y)," where x=man, and y= parent**.

  Since there is every man so will use ∀, and it will be represented as follows:

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

- **3. Some boys play cricket.**
- In this question, 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**:

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

Some more examples

All students are smart.

$\forall$ x (Student(x) $\rightarrow$ Smart(x))

There exists a student.

$\exists$ x Student(x).

There exists a smart student.

$\exists$ x (Student(x) $\wedge$ Smart(x))

Every student loves some student.

$\forall$ x (Student(x) $\rightarrow$ $\exists$ y (Student(y) $\wedge$ Loves(x,y)))

Bill is a student.

Student(Bill)

Bill takes either Analysis or Geometry (but not both)

Takes(Bill, Analysis) $\Leftrightarrow$ $\neg$ Takes(Bill, Geometry)

Bill takes Analysis or Geometry (or both).

$\text{Takes}(\text{Bill}, \text{Analysis}) \lor \text{Takes}(\text{Bill}, \text{Geometry})$

Bill takes Analysis and Geometry.

$\text{Takes}(\text{Bill}, \text{Analysis}) \land \text{Takes}(\text{Bill}, \text{Geometry})$

Bill does not take Analysis.

$\neg\, \text{Takes}(\text{Bill}, \text{Analysis}).$

No student loves Bill.

$\neg\, \exists\, x\ (\ \text{Student}(x) \land \text{Loves}(x, \text{Bill})\ )$

Bill has at least one sister.

$\exists\, x\ \text{SisterOf}(x,\text{Bill})$

Bill has no sister.

$\neg\, \exists\, x\ \text{SisterOf}(x,\text{Bill})$

KNOWLEDGE ENGINEERING

- Imagine an education company wanting to automate the teaching of children in subjects from biology to computer science (requiring to capture the knowledge of teachers and subject matter experts-SME)
- Oncologists choosing the best treatment for their patients (requiring expertise and knowledge from information contained in medical journals, textbooks, and drug databases).

Knowledge Engineering in FOL

The following steps are used to develop the knowledge base:

1. Identify the problem or task.

2. Assemble the relevant knowledge to the given problem or task.

3. Decide on a vocabulary of predicates, functions and constants.

4. Encode the general knowledge about the domain and a description of the specific problem instance.

5. Apply queries to the inference procedure and get the answers.

6. Finally, **debug** the knowledge base.

- **Identify the Task**
  - • Outline the range of questions
  - • Determine available facts
  - • What is needed to connect problems to answers?
- **Assemble the Relevant Knowledge**
  - • Knowledge acquisition
  - • Understand the scope of the knowledge base
  - • How does the domain work?

- **Decide on a Vocabulary**
  - • Determine vocabulary of predicates, functions, and constants
  - • How is the knowledge base represented?
  - **ONTOLOGY-**specification of the meanings of the symbols in an information system i.e **specification of a conceptualization**
- **Encode General knowledge**

  • Encode the general knowledge about the domain
- **Encode the Problem**

  • Describe the specific problem instance using the general knowledge

- **Pose Queries**
  - – • Let the inference procedure operate on the axioms to derive facts
- **Debugging**
  - • Check the knowledge base for errors

Representing Facts in First-Order Logic

1. Lucy is a professor

2. All professors are people.

3. John is the dean.

4. Deans are professors.

5. All professors consider the dean a friend or don’t know him.

6. Everyone is a friend of someone.

7. People only criticize people that are not their friends.

8. Lucy criticized John .

Knowledge base:

- $\text{is-prof}(\text{lucy})$
- $\forall x\ (\ \text{is-prof}(x) \rightarrow \text{is-person}(x)\ )$
- $\text{is-dean}(\text{John})$
- $\forall x\ (\text{is-dean}(x) \rightarrow \text{is-prof}(x))$
- $\forall x\ (\forall y\ (\ \text{is-prof}(x) \wedge \text{is-dean}(y) \rightarrow \text{is-friend-of}(y,x) \vee \neg \text{knows}(x, y)\ )\ )$
- $\forall x\ (\exists y\ (\ \text{is-friend-of}\ (y, x)\ )\ )$
- $\forall x\ (\forall y\ (\text{is-person}(x) \wedge \text{is-person}(y) \wedge \text{criticize}\ (x,y) \rightarrow \neg \text{is-friend-of}\ (y,x)))$
- $\text{criticize}(\text{lucy}, \text{John}\ )$

Question: Is John no friend of Lucy?

$\neg \text{is-friend-of}(\text{John}\ ,\text{lucy})$

Knowledge Engineering

1. All professors are people.
2. Deans are professors.
3. All professors consider the dean a friend or don’t know him.
4. Everyone is a friend of someone.
5. People only criticize people that are not their friends.

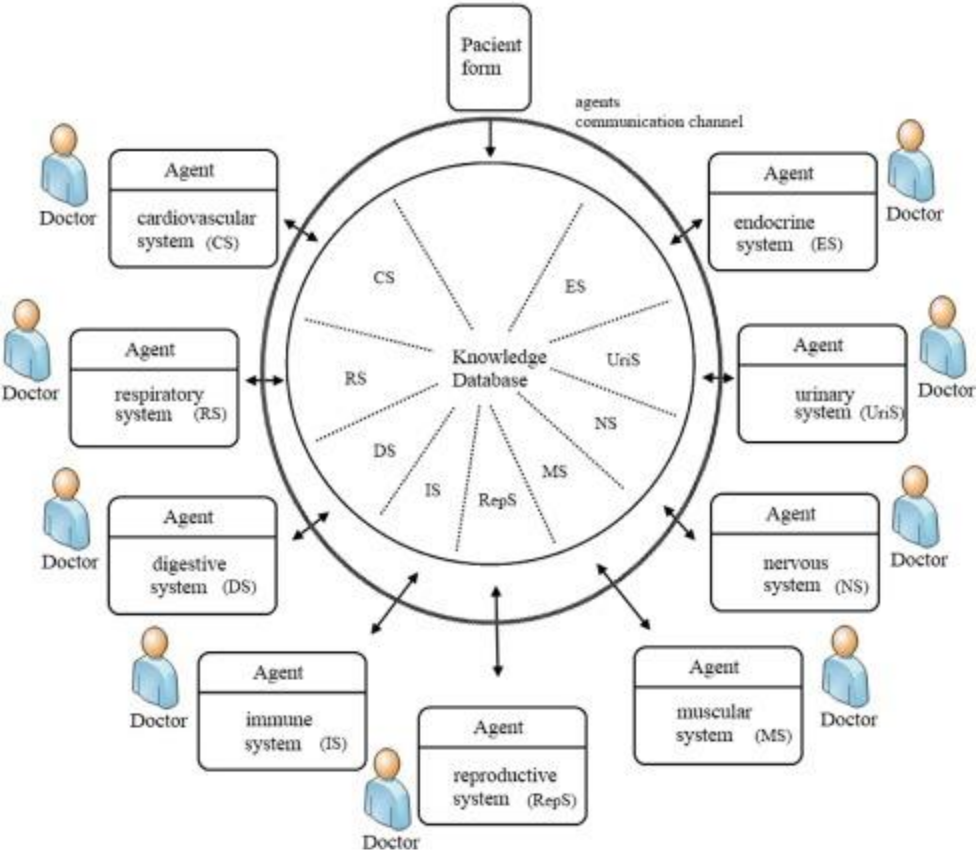
General Knowledge

6. Lucy* is a professor
7. John is the dean.
8. Lucy criticized John.

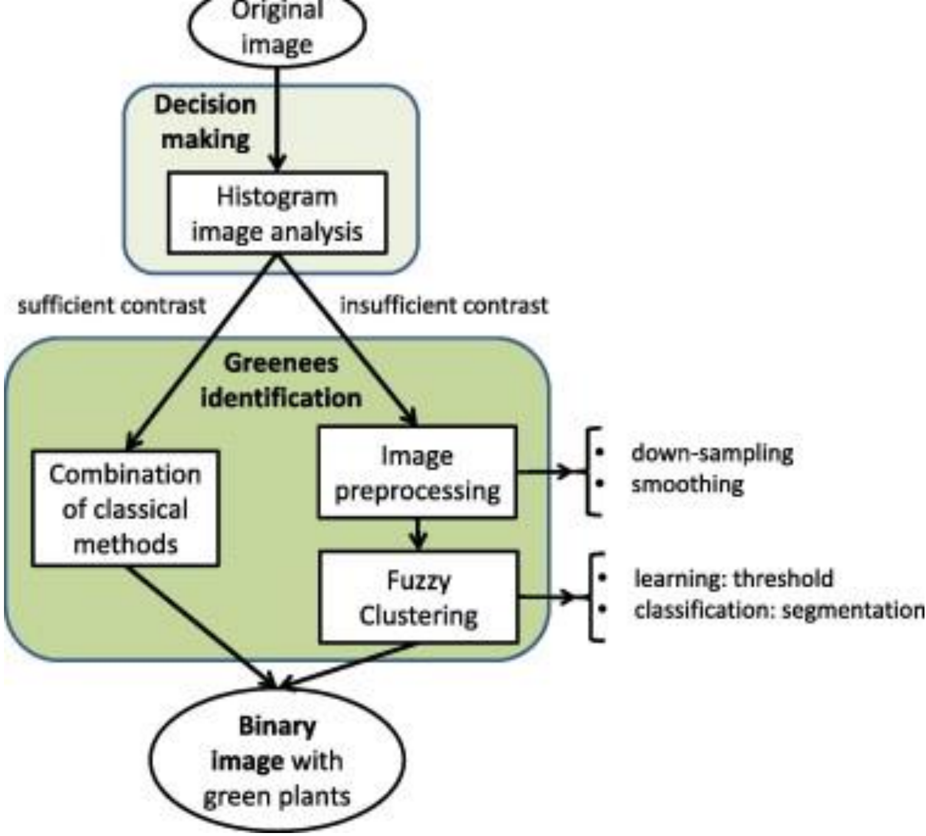
Specific problem

9. Is John a friend of Lucy’s?

Query



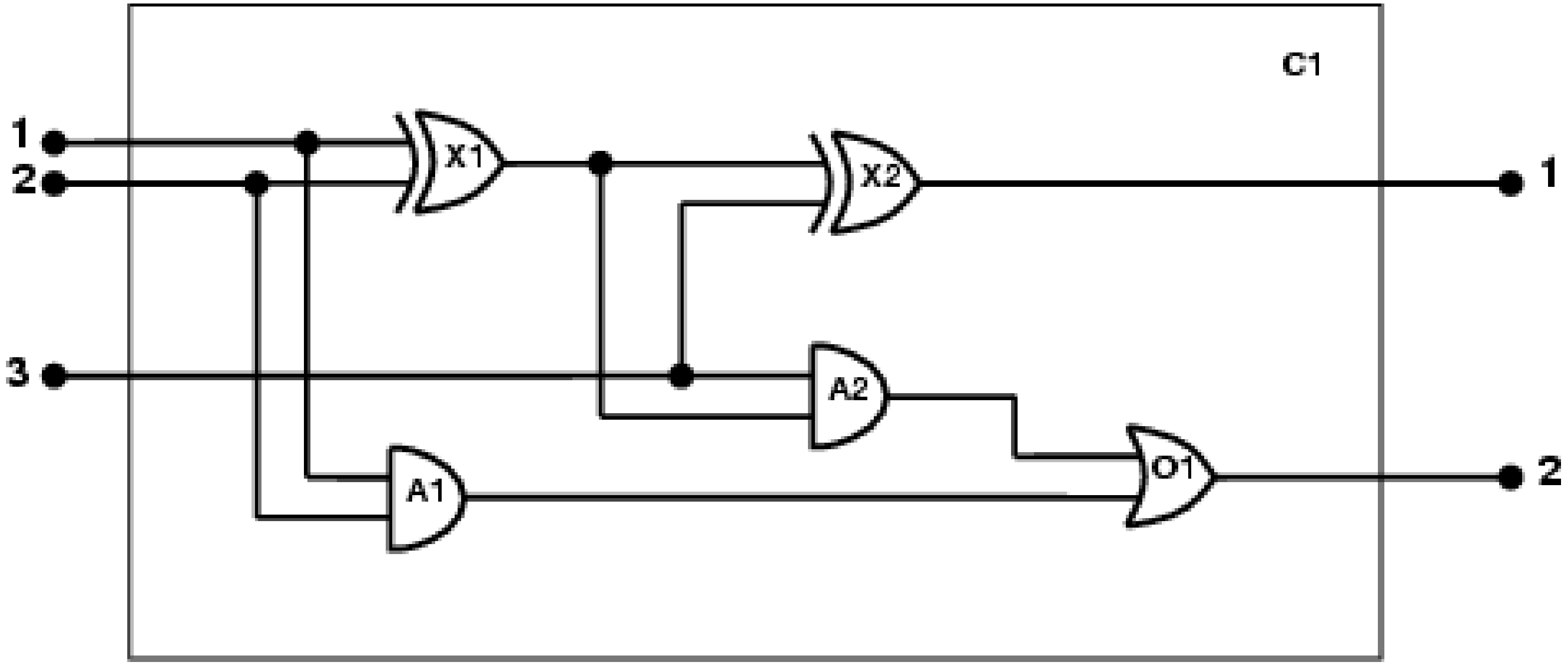
Pacient form
agents communication channel
Knowledge Database
CS
ES
UriS
NS
MS
RepS
IS
DS
RS
Agent
cardiovascular system (CS)
Doctor
Agent
endocrine system (ES)
Doctor
Agent
respiratory system (RS)
Doctor
Agent
urinary system (UriS)
Doctor
Agent
digestive system (DS)
Doctor
Agent
nervous system (NS)
Doctor
Agent
immune system (IS)
Doctor
Agent
reproductive system (RepS)
Doctor
Agent
muscular system (MS)
Doctor



Original image
Decision making
Histogram image analysis
sufficient contrast
insufficient contrast
Greenees identification
Combination of classical methods
Image preprocessing
down-sampling
smoothing
Fuzzy Clustering
learning: threshold
classification: segmentation
Binary image with green plants

The electronic circuits domain

One-bit full adder



The electronic circuits domain

1. **Identify the task**
   - o Does the circuit actually add properly? (circuit verification)
2. **Assemble the relevant knowledge**
   - o Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   - o Irrelevant: size, shape, color, cost of gates
3. **Decide on a vocabulary**
   - o Gate(X_1)
   - o Terminal(x)
   - o Type(X_1) = XOR
   - o Circuit(C_1)
   - o Signal(t)-signal value at terminal t
   - o In(1,X_1)-first i/p terminal at gate X_1

The electronic circuits domain

4. Encode general knowledge of the domain

If two terminals are connected, then they have the same signal:

$\forall t_1, t_2\ \text{Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$

- $\forall t\ \text{Signal}(t) = 1 \lor \text{Signal}(t) = 0$
- $\forall t_1, t_2\ \text{Connected}(t_1, t_2) \Rightarrow \text{Connected}(t_2, t_1)$
- $\forall g\ \text{Type}(g) = \text{OR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \exists n\ \text{Signal}(\text{In}(n,g)) = 1$
- $\forall g\ \text{Type}(g) = \text{AND} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 0 \Leftrightarrow \exists n\ \text{Signal}(\text{In}(n,g)) = 0$
- $\forall g\ \text{Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g))$
- $\forall g\ \text{Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1,g)) \neq \text{Signal}(\text{In}(1,g))$

2. The signal at every terminal is either 1 or 0:
   $\forall t \;\; Terminal(t) \;\Rightarrow\; Signal(t) = 1 \vee Signal(t) = 0 \;.$
3. Connected is commutative:
   $\forall t_1, t_2 \;\; Connected(t_1, t_2) \;\Leftrightarrow\; Connected(t_2, t_1) \;.$
4. There are four types of gates:
   $\forall g \;\; Gate(g) \wedge k = Type(g) \;\Rightarrow\; k = AND \vee k = OR \vee k = XOR \vee k = NOT \;.$
5. An AND gate's output is 0 if and only if any of its inputs is 0:
   $\forall g \;\; Gate(g) \wedge Type(g) = AND \;\Rightarrow$
   $Signal(Out(1, g)) = 0 \;\Leftrightarrow\; \exists n \;\; Signal(In(n, g)) = 0 \;.$
6. An OR gate's output is 1 if and only if any of its inputs is 1:
   $\forall g \;\; Gate(g) \wedge Type(g) = OR \;\Rightarrow$
   $Signal(Out(1, g)) = 1 \;\Leftrightarrow\; \exists n \;\; Signal(In(n, g)) = 1 \;.$
7. An XOR gate's output is 1 if and only if its inputs are different:
   $\forall g \;\; Gate(g) \wedge Type(g) = XOR \;\Rightarrow$
   $Signal(Out(1, g)) = 1 \;\Leftrightarrow\; Signal(In(1, g)) \neq Signal(In(2, g)) \;.$
8. A NOT gate's output is different from its input:
   $\forall g \;\; Gate(g) \wedge (Type(g) = NOT) \;\Rightarrow$
   $Signal(Out(1, g)) \neq Signal(In(1, g)) \;.$
9. The gates (except for NOT) have two inputs and one output.
   $\forall g \;\; Gate(g) \wedge Type(g) = NOT \;\Rightarrow\; Arity(g, 1, 1) \;\;.$
   $\forall g \;\; Gate(g) \wedge k = Type(g) \wedge (k = AND \vee k = OR \vee k = XOR) \;\Rightarrow$
   $Arity(g, 2, 1)$
10. A circuit has terminals, up to its input and output arity, and nothing beyond its arity:
    $\forall c, i, j \;\; Circuit(c) \wedge Arity(c, i, j) \;\Rightarrow$
    $\forall n \;\; (n \leq i \;\Rightarrow\; Terminal(In(c, n))) \wedge (n > i \;\Rightarrow\; In(c, n) = Nothing) \wedge$
    $\forall n \;\; (n \leq j \;\Rightarrow\; Terminal(Out(c, n))) \wedge (n > j \;\Rightarrow\; Out(c, n) = Nothing)$
11. Gates, terminals, signals, gate types, and *Nothing* are all distinct.
    $\forall g, t \;\; Gate(g) \wedge Terminal(t) \;\Rightarrow$
    $g \neq t \neq 1 \neq 0 \neq OR \neq AND \neq XOR \neq NOT \neq Nothing \;.$
12. Gates are circuits.
    $\forall g \;\; Gate(g) \;\Rightarrow\; Circuit(g)$

The electronic circuits domain

5. **Encode the specific problem instance**

Type(X_1) = XOR

Type(A_1) = AND

Type(O_1) = OR

Type(X_2) = XOR

Type(A_2) = AND

Connected(Out(1,X_1),In(1,X_2))

Connected(Out(1,X_1),In(2,A_2))

Connected(Out(1,A_2),In(1,O_1))

Connected(Out(1,A_1),In(2,O_1))

Connected(Out(1,X_2),Out(1,C_1))

Connected(Out(1,O_1),Out(2,C_1))

Connected(In(1,C_1),In(1,X_1))

Connected(In(1,C_1),In(1,A_1))

Connected(In(2,C_1),In(2,X_1))

Connected(In(2,C_1),In(2,A_1))

Connected(In(3,C_1),In(2,X_2))

Connected(In(3,C_1),In(1,A_2))

The electronic circuits domain

6. Pose queries to the inference procedure

What are the possible sets of values of all the terminals for the adder circuit?

$\exists i_1, i_2, i_3, o_1, o_2$ Signal(In(1,C_1)) = $i_1 \wedge$ Signal(In(2,C_1)) = $i_2 \wedge$ Signal(In(3,C_1)) = $i_3 \wedge$ Signal(Out(1,C_1)) = $o_1 \wedge$ Signal(Out(2,C_1)) = o_2

7. Debug the knowledge base

Inference in First-Order Logic

Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences.

FEW TERMINOLOGIES

SUBSTITUTION

EQUALITY

Substitution

- Replaces **VARIABLES** with **CONSTANTS**
- If we write **F[a/x]**, so it refers to substitute a constant "**a**" in place of variable "**x**".
- Ex: SUBT ({ X / 49 } , Perfect_Square (X))

→Perfect_Square(49).

- SUBT ({ X / 49 , Y / 7 } ,Divide (X , Y))

→Divide (4 9 , 7)

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.**
  - **Example: ¬(x=y) which is equivalent to x ≠y.**

FOL inference rules for Quantifier

- 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(ELIMINATION)**
  - **EXISTENTIAL INSTANTIATION(ELIMINATION)-ONLY ONCE**
  - **EXISTENTIAL INTRODUCTION**

Example:1[UG]

$$\frac{P(c)}{\forall x\, P(x)}$$

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

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

Example:2 [UI]

IF "**Every** person like ice-cream"=> ∀x P(x) so we can infer that

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

$$\frac{\forall x\, P(x)}{P(c)}$$

Universal Generalization

- 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).
- **This rule can be used if we want to show that every element has a similar property**.

"All kings who are greedy are Evil."

FOL: -

∀x king(x) ∧ greedy (x) → Evil (x),

→Universal Instantiation

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

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

King(**Father(John)**) ∧ Greedy (**Father(John)**) → Evil (**Father(John)**)

Universal Instantiation

- Also called as universal elimination or UI is a valid inference rule.
  - It can be applied multiple times to add new sentences.
  - The new KB is logically equivalent to the previous KB.
- As per UI, **we can infer any sentence obtained by substituting a ground term for the variable**.
- 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**.

Example 3 [E I]

$$\frac{\exists x\ P(x)}{P(c)}$$

∃x Crown(x) ∧ OnHead(x, John),

- So we can infer: **Crown(K) ∧ OnHead(K, John),** as long as K does not appear in the knowledge base.
  - The above used K is a constant symbol, which is called **Skolem constant**.
  - The Existential instantiation is a special case of **Skolemization process**

Existential Instantiation

- 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.
  - “The new KB is not logically equivalent to old KB, but it will be **satisfiable if old KB was satisfiable.”**
  - 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.

Example 4 :[E G]

$$\frac{P(c)}{\exists x P(x)}$$

"Priyanka got good marks in English.“

"Therefore, someone got good marks in English."

Existential Generalization

Also known as existential introduction - 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.

Name	Rule
Universal instantiation	$\begin{array}{rl} & \forall x P(x) \\ \hline \therefore & P(c) \end{array}$
Universal generalization	$\begin{array}{rl} & P(c) \text{ [for an arbitrary } c\text{]} \\ \hline \therefore & \forall x P(x) \end{array}$
Existential instantiation	$\begin{array}{rl} & \exists x P(x) \\ \hline \therefore & P(c) \text{ [for some particular } c\text{]} \end{array}$
Existential generalization	$\begin{array}{rl} & P(c) \text{ [for some particular } c\text{]} \\ \hline \therefore & \exists x P(x) \end{array}$

Suppose the KB contains just the following:

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

Unification

Process of making two different logical atomic expressions identical by finding a substitution.

- Making expressions look identical via substitution
- Unification depends on the **substitution process.**

- It takes two literals as input and makes them identical using substitution.

P (x , f (y))

P (a , f (g (z)))

→ x = a ; y = g(z)

Unification [a / x , g (z) / y]

Unification

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

$\theta = \{x/John, y/John\}$ works

$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	[redacted]
$Knows(John, x)$	$Knows(y, OJ)$	[redacted]
$Knows(John, x)$	$Knows(y, Mother(y))$	[redacted]
$Knows(John, x)$	$Knows(x, OJ)$	[redacted]

Standardizing apart eliminates overlap of variables, e.g., $Knows(z_{17}, OJ)$

Example: Find the MGU for Unify{King(x), King(John)}

- Let L1 = King(x), L2 = King(John),
- **Substitution θ = {John/x}** is a unifier for these atoms and applying this substitution, and both expressions will be identical.

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.

Algorithm: Unify(L1, L2)

- Step 1: If L1 or L2 is a **variable or constant**, then:
  - a) If **L1 or L2 are identical**, then return **NIL.**
  - b) Else if **L1is a variable,**
    - a. then if L1 occurs in L2, then return FAILURE
    - b. **Else return { (L2/ L1)}-** substitution
  - c) Else **if L2 is a variable**,
    - a. If L2 occurs in L1 then return FAILURE,
    - b. Else **return {(L1/ L2)}.**
  - d) Else return FAILURE.
- Step 2: If the **initial Predicate symbol in L1 and L2 are not same**, then return FAILURE.

- Step. 3: IF L1 and L2 have a **different number of arguments**, then return **FAILURE**.
- Step. 4: Set Substitution set(SUBST) to NIL.
- Step. 5: For i=1 to the number of elements in L1.
  - a) Call Unify function with the ith element of L1 and ith element of L2, and put the result into S.
  - b) If S = failure then returns Failure
  - c) If S ≠ NIL then do,
    - a. Apply S to the remainder of both L1 and L2.
    - b.SUBST= APPEND(S, SUBST).
- Step.6: Return SUBST.

- **4. Find the MGU of UNIFY(prime (11), prime(y))**
- Here, L1 = {prime(11) , and L2 = prime(y)}

  S_0 => {prime(11) , prime(y)}

  SUBST θ= {11/y}

- S_1 => {prime(11) , prime(11)} , **Successfully unified.**

  Unifier: {11/y}.

- **6. UNIFY(knows(Richard, x), knows(Richard, John))**

- Here, L1 = knows(Richard, x), and L2 = knows(Richard, John)

  S_0 => { knows(Richard, x); knows(Richard, John)}

  SUBST θ= {John/x}

  S_1 => { knows(Richard, John); knows(Richard, John)}, **Successfully Unified.**

  Unifier: {John/x}.

Resolution

- Theorem proving technique that proceeds by building refutation proofs, i.e., **proofs by contradictions**.
  - It was invented by a Mathematician John Alan Robinson in the year 1965.
- If there are various statements given, and we **need to prove a conclusion of those statements.**
- **UNIFICATION IS A KEY CONCEPT IN PROOFS BY RESOLUTIONS.**
- Resolution is a single inference rule which can efficiently operate on the **conjunctive normal form or clausal form**. CNF (**P → Q is logically equivalent** to **¬ P ∨ Q**)
  - **Clause**: Disjunction of literals (an atomic sentence) is called a **clause**.
  - It is also known as a unit clause.

The resolution inference rule

- The resolution rule for first-order logic is simply a lifted version of the propositional rule.
- **Resolution can resolve two clauses if they contain complementary literals, which are assumed to be standardized apart so that they share no variables.**
  - Where $\mathbf{l}_i$ and $\mathbf{m}_j$ are complementary literals.
- This rule is also called the **binary resolution rule** because it only resolves exactly two literals.

Resolution Rule of Inference

Example:

$$\begin{array}{lll} \text{Assume:} & E_1 \lor E_2 & \text{playing tennis or raining} \\ \text{and} & \neg E_2 \lor E_3 & \text{not raining or working} \\ \hline \text{Then:} & E_1 \lor E_3 & \text{playing tennis or working} \end{array}$$

“Resolvent”

General Rule:

$$\begin{array}{ll} \text{Assume:} & E \lor E_{12} \lor \ldots \lor E_{1k} \\ \text{and} & \neg E \lor E_{22} \lor \ldots \lor E_{2l} \\ \hline \text{Then:} & E_{12} \lor \ldots \lor E_{1k} \lor E_{22} \lor \ldots \lor E_{2l} \end{array}$$

Note: E_{ij} can be negated.

Another Example

- We can resolve two clauses which are given below:

[Animal (g(x) V Loves (f(x), x)]   and   [¬ Loves(a, b) V ¬Kills(a, b)]

Where two complimentary literals are: **Loves (f(x), x) and ¬ Loves (a, b)**

[Animal (g(x) V ¬ Kills(f(x), x)]

STEPS FOR RESOLUTION:

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph (unification).

Simple Example

- If it is sunny and warm day you will enjoy
- If it is raining you will get wet
- It is a warm day
- It is raining
- It is sunny

Goal : - You will enjoy

Convert to FOL

- **If it is sunny and warm day you will enjoy**
- (sunny) ∧ (warm)→enjoy
- **If it is raining you will get wet**
- raining → wet
- **It is a warm day**
- (warm)
- **It is raining**
- (raining)
- **It is sunny**
- (sunny)

FOL to CNF

- (sunny) ∧ (warm)→enjoy

¬ (sunny ∧ warm) V enjoy

¬ sunny V ¬ warm V enjoy

- raining → wet

¬ raining V wet

Proof by Contradiction(negate the goal)

¬ enjoy

Draw the resolution graph

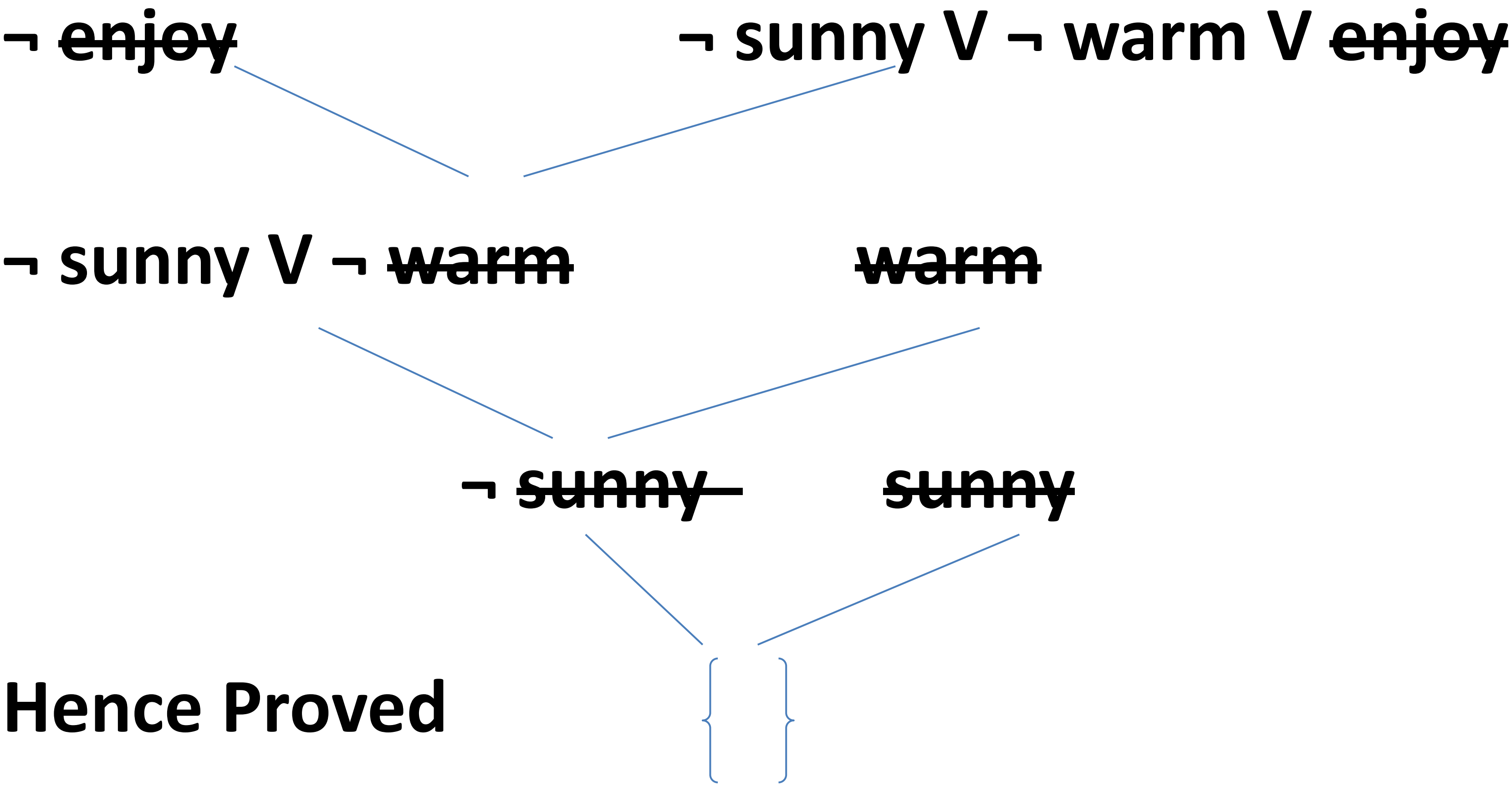
¬ sunny V ¬ warm V enjoy

¬ raining V wet

warm

raining

sunny



¬ ~~enjoy~~

¬ sunny V ¬ warm V ~~enjoy~~

¬ sunny V ¬ ~~warm~~

~~warm~~

¬ ~~sunny~~

~~sunny~~

{ }

Hence Proved

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.

1. Let, P: Humidity is high.

Q: Sky is cloudy.

Rep as P V Q

2. Q: Sky is cloudy. …**from(1)**

Let, R: It will rain.

Rep as Q → R.

3. P: Humidity is high. …**from(1)**

Let, S: It is hot.

Rep as P → S.

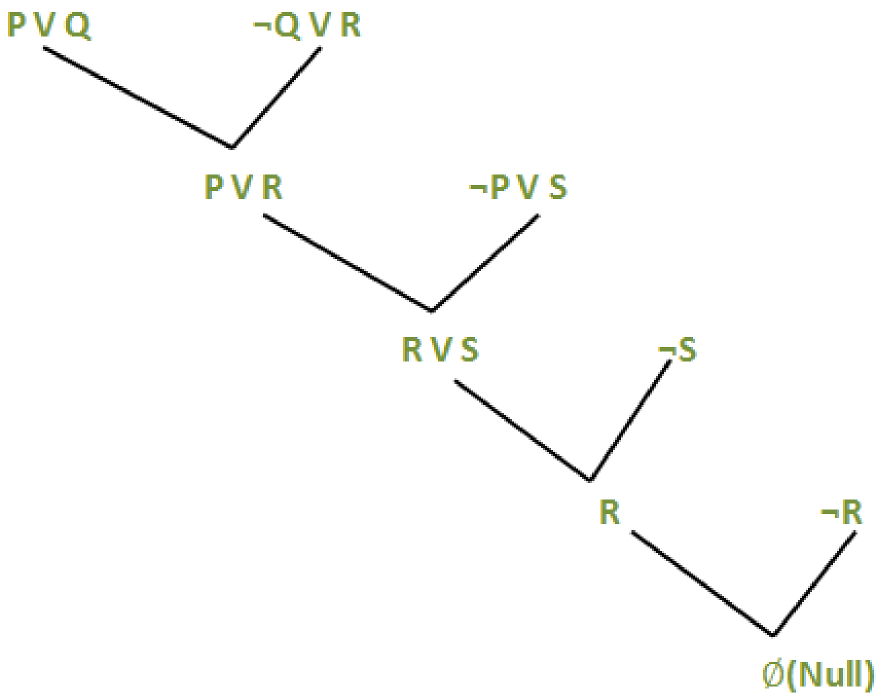
4**. ¬S**: It is not hot.

<u>Applying resolution method:</u>

- 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.

Goal: It will rain.



P V Q

¬Q V R

P V R

¬P V S

R V S

¬S

R

¬R

Ø(Null)

DIVYA WENT TO CITY MALL YESTERDAY EVENING AND GOT A VERY BEAUTIFUL RED PURSE ON DISCOUNT THAT WENT PERFECT WITH HER NEW DRESS.

  - This statement can derive various new statements from the given statement.
  - Moreover, if it has more information on purchases, accessories, styles, human behaviour in the knowledgebase, the number of new statements it can derive increases more and more.
  - Some of the new derivable statements are-

- • Divya purchased a purse.
- • Last evening Divya wasn’t home.
- • There’s a mall named city mall.
- • City mall was open last evening.
- • City mall has at least one shop that sells purses.
- • There is discount sale on purses.
- • Divya has at least one new dress.
- • Divya didn’t have a red purse.
- • There was at least one woman at mall last evening.
- • There was at least one woman named Divya at the mall last evening.
- • Divya doesn’t mind carrying a purse.

1. Gita likes all kinds of food.
2. Mango and chapati are food.
3. Gita eats almond and is still alive.
4. Anything eaten by anyone and is still alive is food.

- **Goal:** Gita likes almond.

1. $\forall x: food(x) \rightarrow likes(Gita,x)$
2. $food(Mango) \wedge food(chapati)$
3. $eats(Gita, almonds) \wedge alive(Gita)$
4. $\forall x \forall y: eats(x,y) \wedge \neg\, killed(x \rightarrow food(y)$
5. **$\forall x: \neg killed(x) \rightarrow alive(x)$**
6. **$\forall x: alive(x) \rightarrow \neg killed(x)$**

1. $\forall x: food(x) \rightarrow likes(Gita,x)$
2. $food(Mango) \land food(chapati)$
3. $eats(Gita, almonds) \land alive(Gita)$
4. **$\forall x \forall y: eats(x,y) \land \neg killed(x$** $\rightarrow food(y)$
5. **$\forall x: \neg killed(x) \rightarrow alive(x)$**
6. **$\forall x: alive(x) \rightarrow \neg killed(x)$**

1. **$\neg food(x) \lor likes(Gita, x)$**
2. $food(Mango), food(chapati)$
3. $eats(Gita, almonds), alive(Gita)$
4. **$\neg eats(x,y) \lor killed(x) \lor food(y)$**
5. **$killed(x) \lor alive(x)$**
6. **$\neg alive(x) \lor \neg killed(x)$**

Negated goal: ¬likes(Gita, almond)

Now, rewrite in CNF form:

1. ¬food(x) V likes(Gita, x)

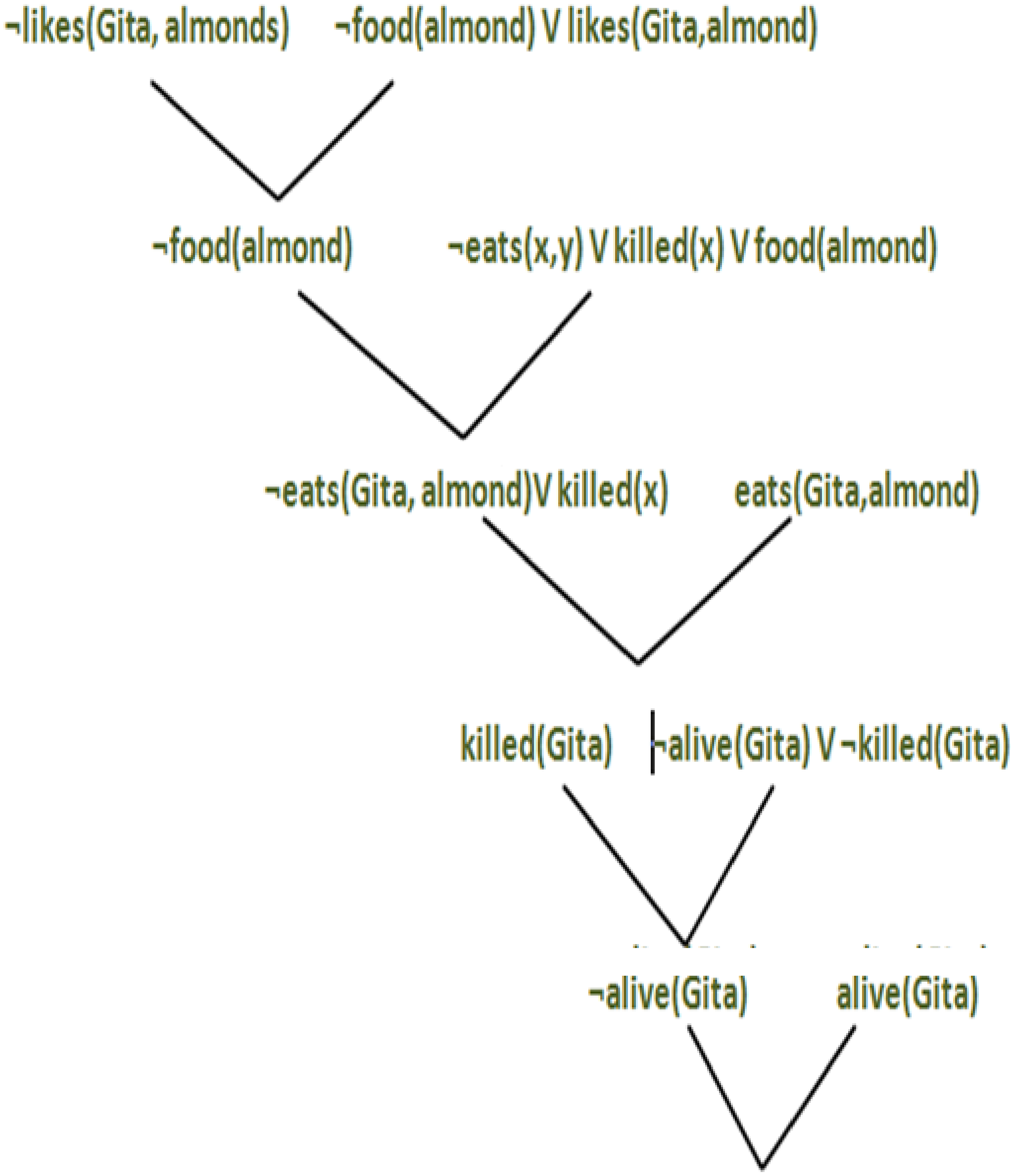
2. food(Mango),food(chapati)

4. ¬eats(x,y) V killed(x) V food(y)

3. eats(Gita, almonds), alive(Gita)

5. killed(x) V alive(x)

6. ¬alive(x) V ¬killed(x)



¬likes(Gita, almonds)
¬food(almond) V likes(Gita,almond)
¬food(almond)
¬eats(x,y) V killed(x) V food(almond)
¬eats(Gita, almond)V killed(x)
eats(Gita,almond)
killed(Gita)
¬alive(Gita) V ¬killed(Gita)
¬alive(Gita)
alive(Gita)

Example:

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

  Prove by resolution that:

John likes peanuts.

Step-1: Conversion of Facts into FOL

a) John likes all kind of food.

b) Apple and vegetable are food

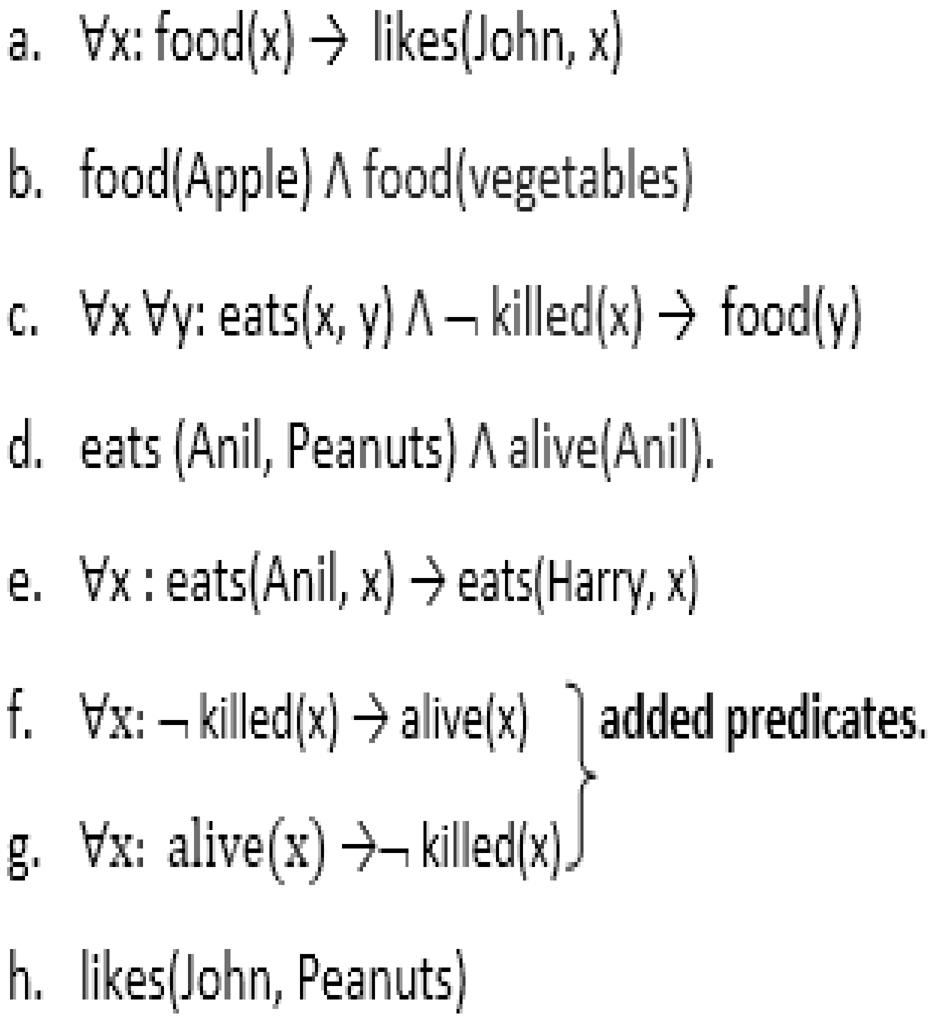
c) Anything anyone eats and not killed is food.

d) Anil eats peanuts and still alive

e) Harry eats everything that Anil eats.

Prove by resolution that:

- John likes peanuts.



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

b. $\text{food}(\text{Apple}) \land \text{food}(\text{vegetables})$

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

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

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

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

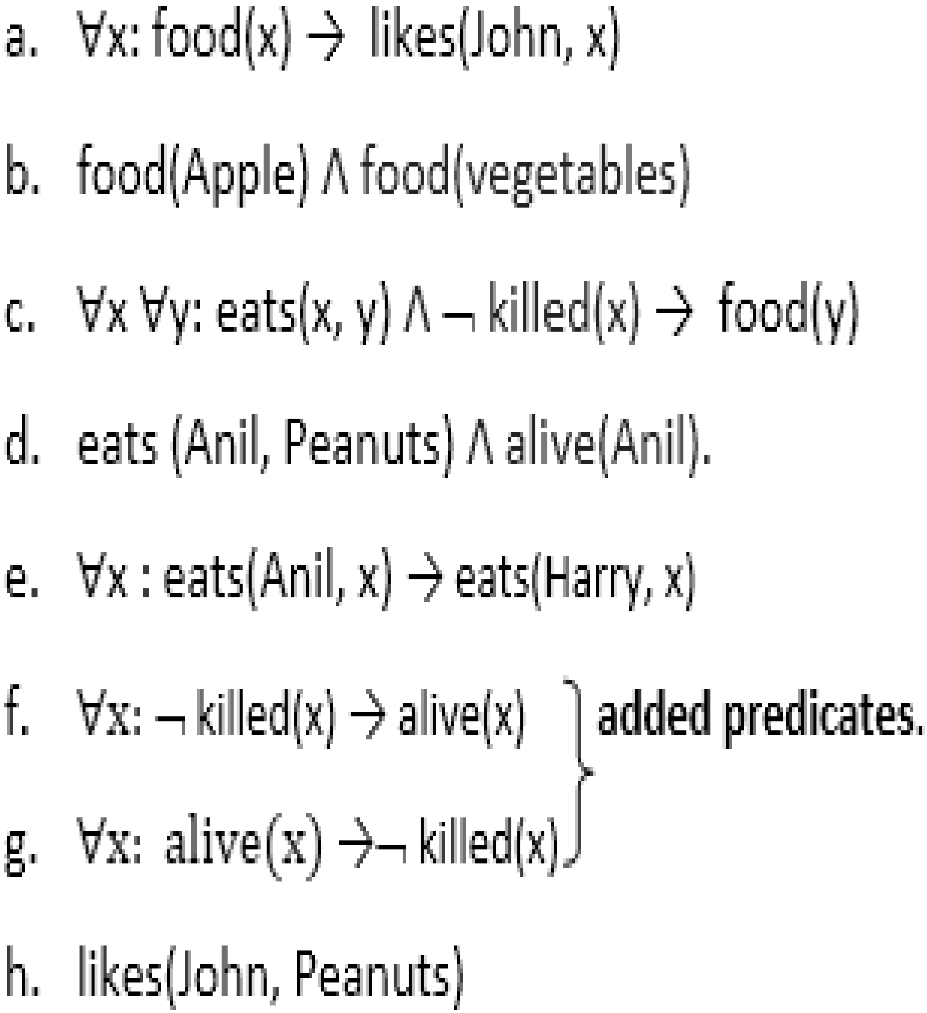
g. $\forall x: \text{alive}(x) \rightarrow \neg \text{killed}(x)$

added predicates.

h. $\text{likes}(\text{John}, \text{Peanuts})$

Step-2: Conversion of FOL into CNF

Eliminate all implication (→) and rewrite



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)

g. ∀x: alive(x) →¬ killed(x)

added predicates.

h. likes(John, Peanuts)

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).

Move negation (¬)inwards and rewrite

∀x ¬ food(x) V likes(John, x)

food(Apple) Λ food(vegetables)

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

eats (Anil, Peanuts) Λ alive(Anil)

∀x ¬ eats(Anil, x) V eats(Harry, x)

∀x killed(x) V alive(x)

∀x ¬ alive(x) V ¬ killed(x)

likes(John, Peanuts).

Rename variables or standardize variables

∀x ¬ food(x) V likes(John, x)

food(Apple) Λ food(vegetables)

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

eats (Anil, Peanuts) Λ alive(Anil)

∀w¬ eats(Anil, w) V eats(Harry, w)

∀g ¬killed(g)] V alive(g)

∀k ¬ alive(k) V ¬ killed(k)

likes(John, Peanuts).

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.

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.

¬ food(x) V likes(John, x)

food(Apple)

food(vegetables)

¬ eats(y, z) V killed(y) V food(z)

eats (Anil, Peanuts)

alive(Anil)

¬ eats(Anil, w) V eats(Harry, w)

killed(g) V alive(g)

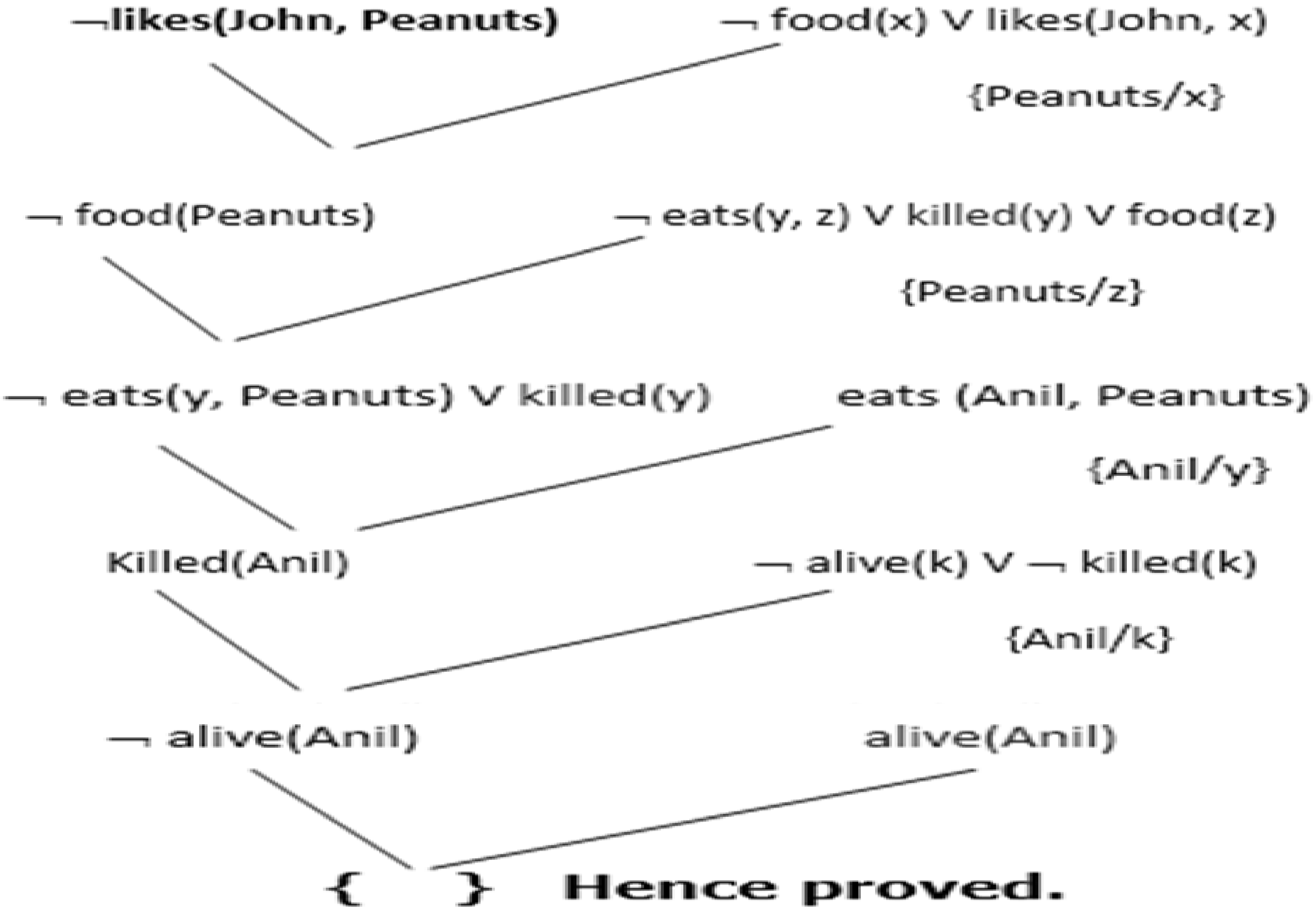
¬ alive(k) V ¬ killed(k)

likes(John, Peanuts).

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)



¬likes(John, Peanuts)

¬ food(x) V likes(John, x)

{Peanuts/x}

¬ food(Peanuts)

¬ eats(y, z) V killed(y) V food(z)

{Peanuts/z}

¬ eats(y, Peanuts) V killed(y)

eats (Anil, Peanuts)

{Anil/y}

Killed(Anil)

¬ alive(k) V ¬ killed(k)

{Anil/k}

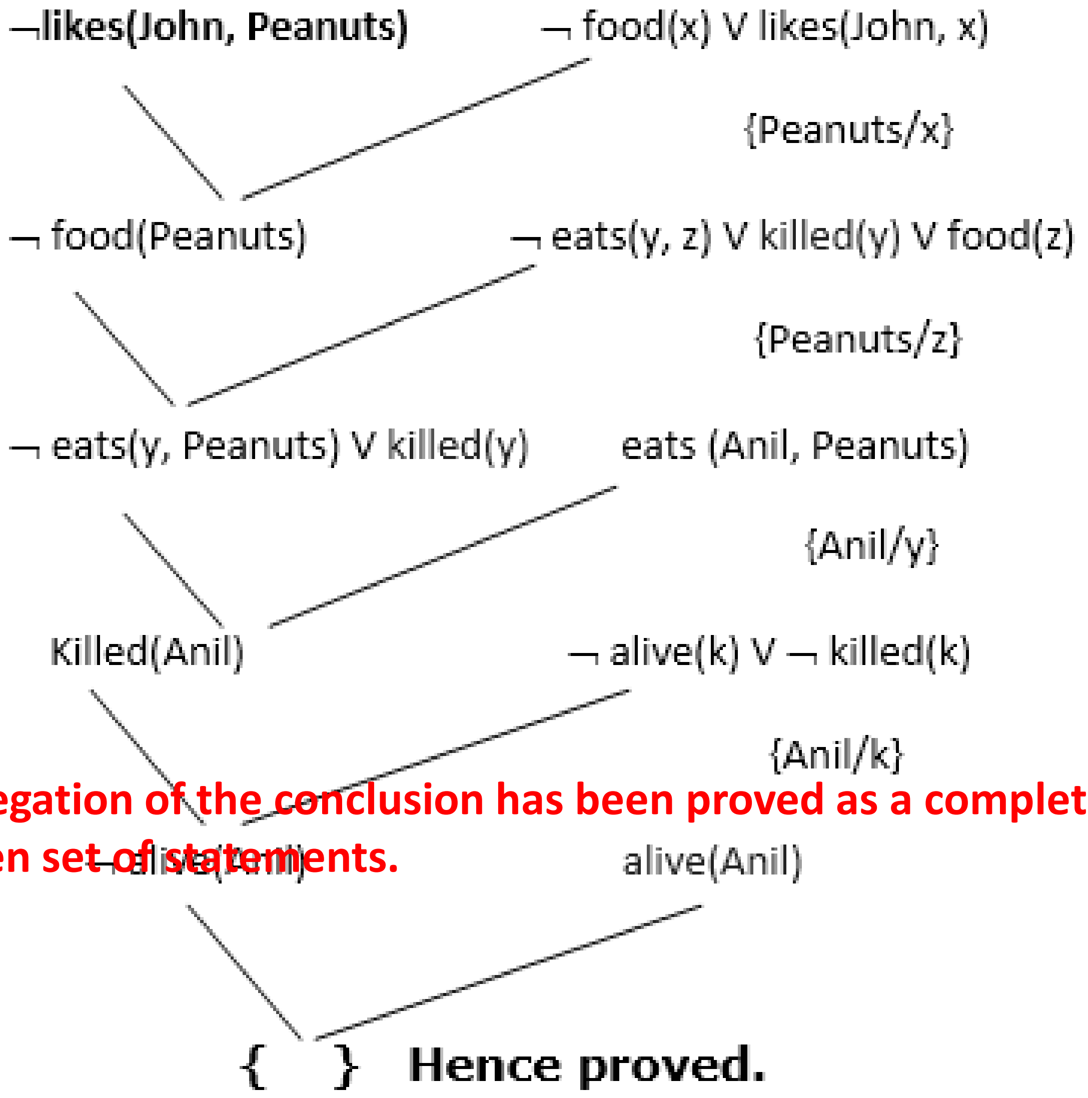
¬ alive(Anil)

alive(Anil)

{ } Hence proved.

Step-4: Draw Resolution graph:

Solve the problem by resolution tree using substitution



¬likes(John, Peanuts)

¬ food(x) V likes(John, x)

{Peanuts/x}

¬ food(Peanuts)

¬ eats(y, z) V killed(y) V food(z)

{Peanuts/z}

¬ eats(y, Peanuts) V killed(y)

eats (Anil, Peanuts)

{Anil/y}

Killed(Anil)

¬ alive(k) V ¬ killed(k)

{Anil/k}

¬ alive(Anil)

alive(Anil)

{ } **Hence proved.**

Hence the negation of the conclusion has been proved as a complete contradiction with the given set of statements.

Explanation of Resolution graph:

- In the first step of resolution graph, **¬likes(John, Peanuts)** , and **likes(John, x)** get resolved(canceled) by substitution of **{Peanuts/x}**, and we are left with **¬ food(Peanuts)**
- In the second step of the resolution graph, **¬ food(Peanuts)** , and **food(z)** get resolved (canceled) by substitution of **{ Peanuts/z}**, and we are left with **¬ eats(y, Peanuts) V killed(y)** .
- In the third step of the resolution graph, **¬ eats(y, Peanuts)** and **eats (Anil, Peanuts)** get resolved by substitution **{Anil/y}**, and we are left with **Killed(Anil)** .
- In the fourth step of the resolution graph, **Killed(Anil)** and **¬ killed(k)** get resolve by substitution **{Anil/k}**, and we are left with **¬ alive(Anil)** .
- In the last step of the resolution graph **¬ alive(Anil)** and **alive(Anil)** get resolved.

Example of Knowledge Base

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

- **Prove that Col. West is a Criminal**

- ... it is a crime for an American to sell weapons to hostile nations:
  - **$American(x) \land Weapon(y) \land Hostile(z) \land Sells(x, y, z) \rightarrow Criminal(x)$**
- Nono ... has some missiles
- **$\exists x\ Owns(Nono, x) \land Missile(x)$**
  - **$Owns(Nono, M1)$ and $Missile(M1)$**
- ... all of its missiles were sold to it by Colonel West
  - **$\forall x\quad Missile(x) \land Owns(Nono, x) \rightarrow Sells(West, x, Nono)$**
- Missiles are weapons*
  - **$Missile(x) \rightarrow Weapon(x)$**

- An enemy of America counts as “hostile”:*
  - **Enemy(x,America) → Hostile(x)**
- West, who is American
  - **American(West)**
- The country Nono, an enemy of America ...
  - **Enemy(Nono,America)**

- Prove that Col. West is a Criminal

i.e $\neg$ Criminal(West)

STEPS FOR RESOLUTION:

1. **Conversion of facts into first-order logic.**

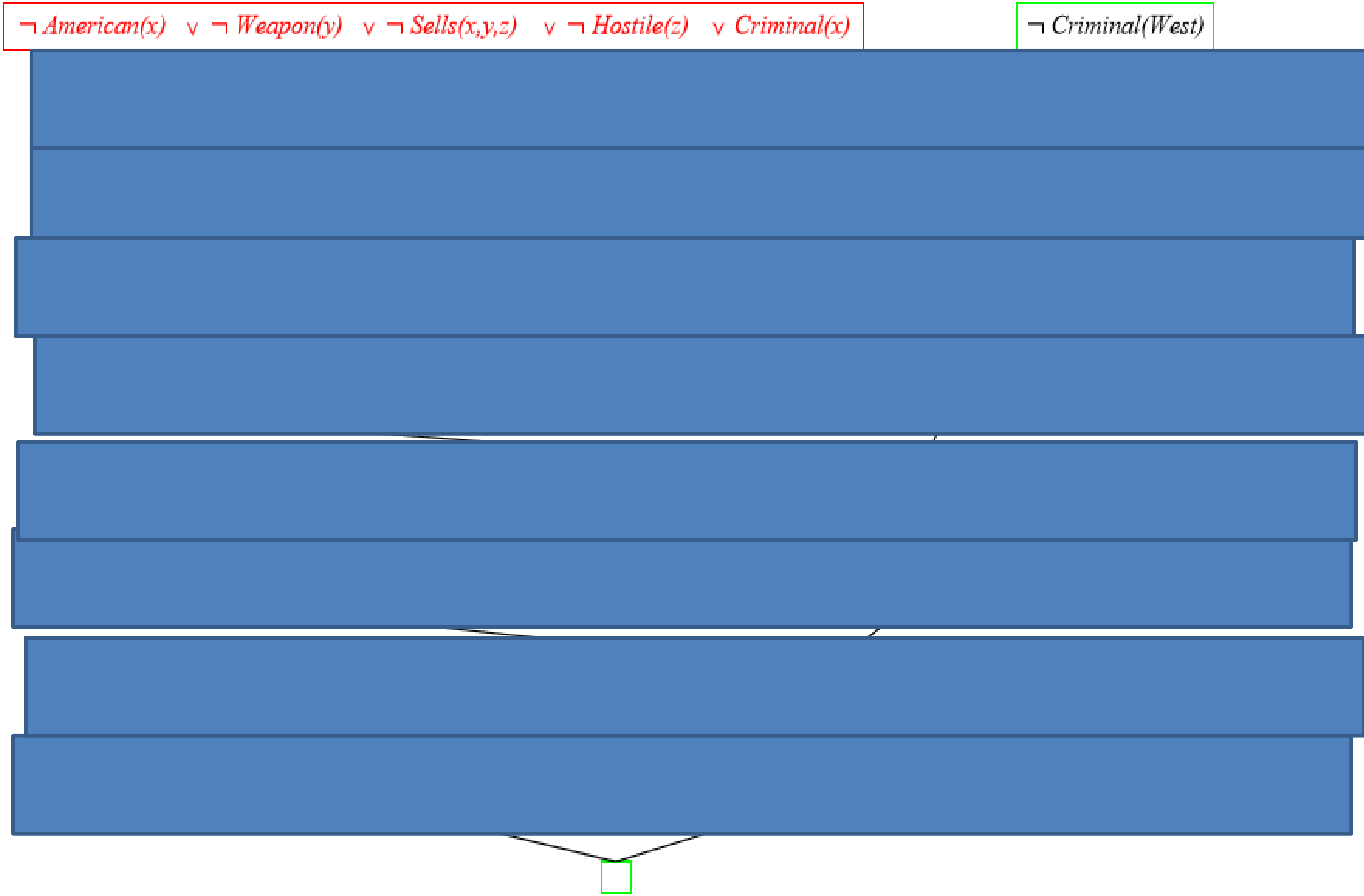
2. **Convert FOL statements into CNF**

3. **Negate the statement which needs to prove (proof by contradiction)**

4. **Draw resolution graph (unification).**

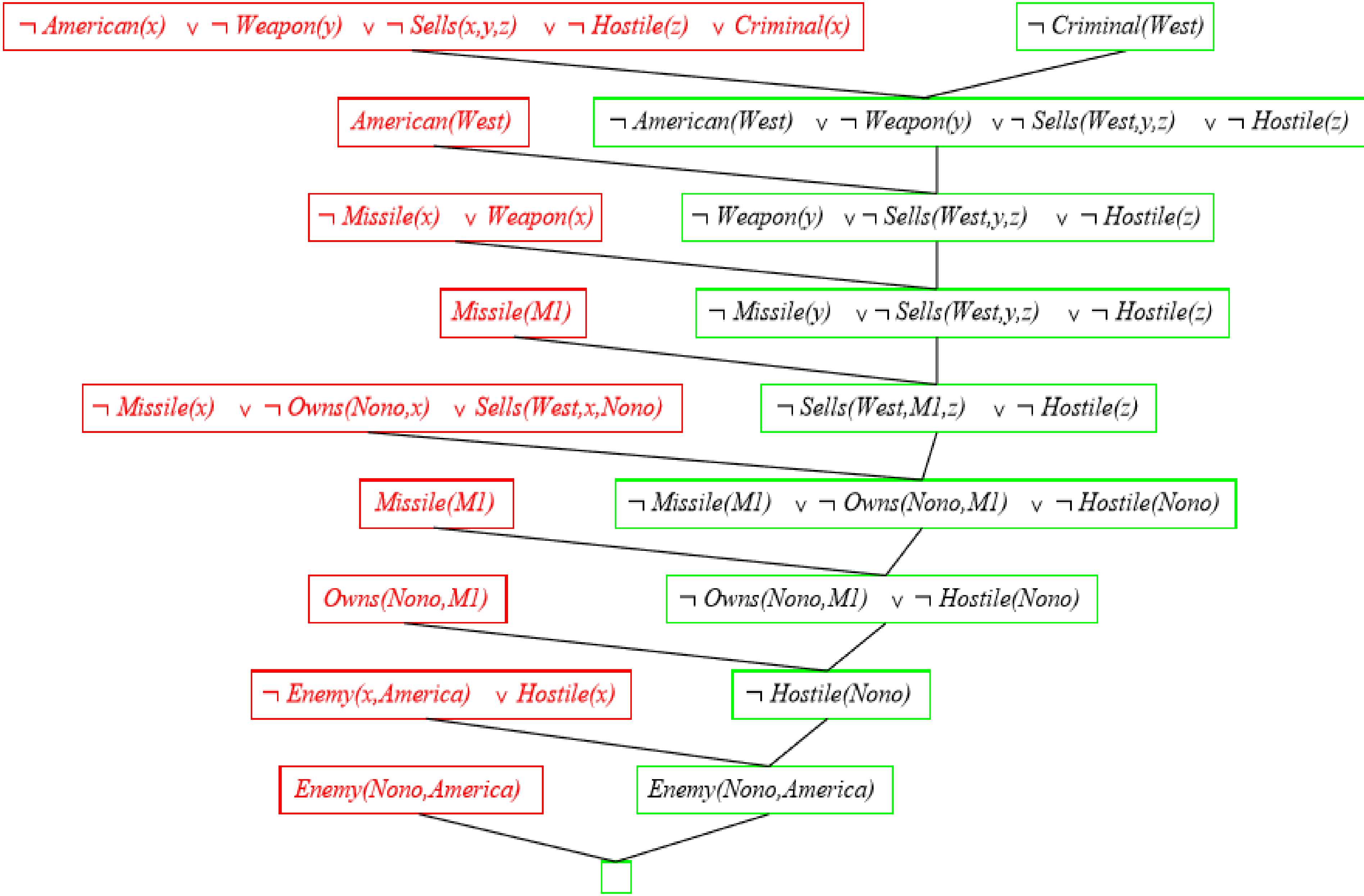
- Rules
  - 1 $American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \implies Criminal(x)$
  - $Missile(M_1)$ and $Owns(Nono, M_1)$
  - $\forall x \;\; Missile(x) \land Owns(Nono, x) \implies Sells(West, x, Nono)$
  - $Missile(x) \Rightarrow Weapon(x)$
  - $Enemy(x, America) \implies Hostile(x)$
  - $American(West)$
  - 7 $Enemy(Nono, America)$

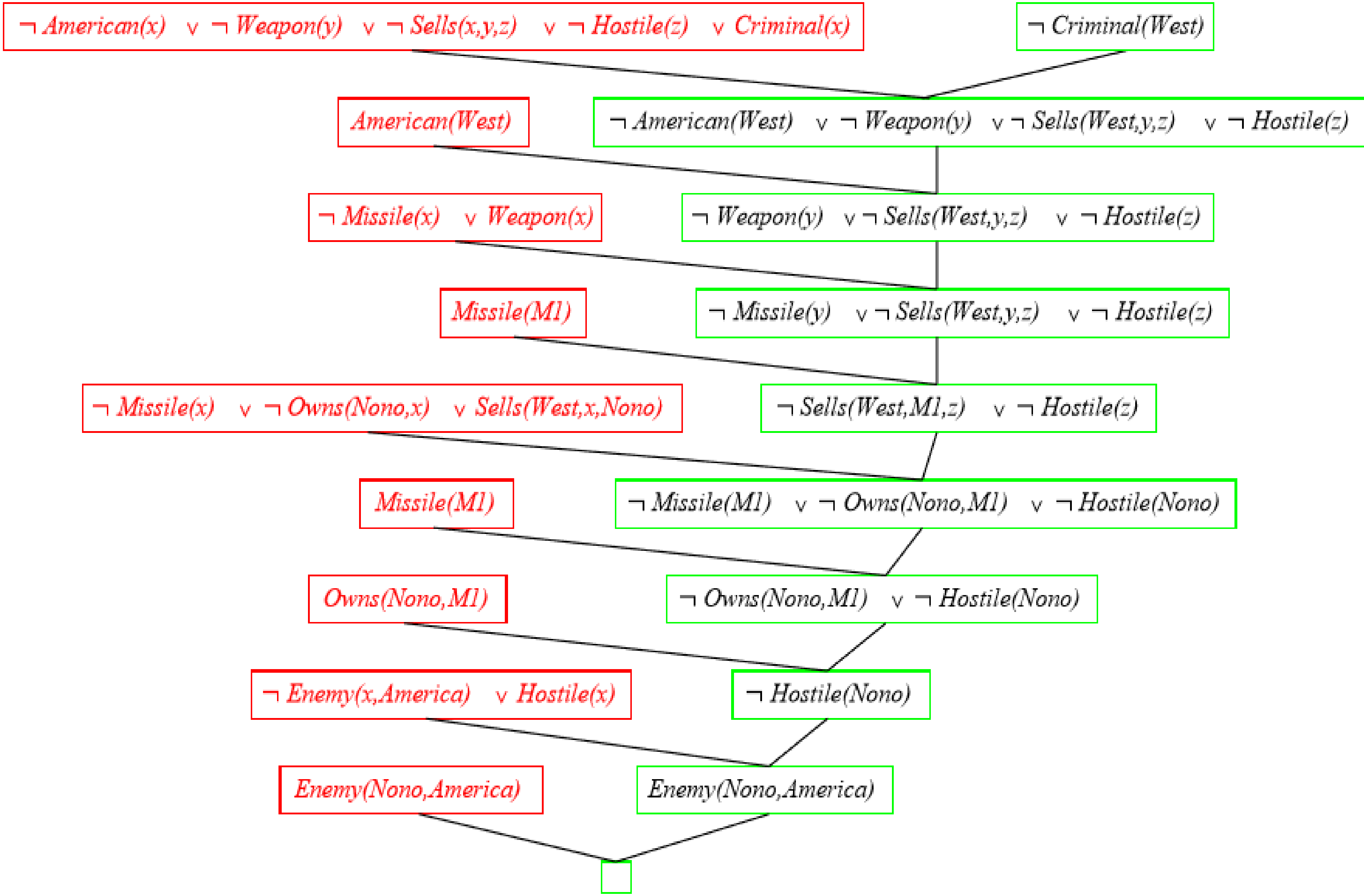
- Converted to CNF
  - $\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$
  - $Missile(M_1)$ and $Owns(Nono, M_1)$
  - $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$
  - $\neg Missile(x) \vee Weapon(x)$
  - $\neg Enemy(x, America) \vee Hostile(x)$
  - $American(West)$
  - $Enemy(Nono, America)$



¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)

¬ Criminal(West)


¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)
¬ Criminal(West)
American(West)
¬ American(West) ∨ ¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ Weapon(x)
¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ ¬ Owns(Nono,x) ∨ Sells(West,x,Nono)
¬ Sells(West,M1,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(M1) ∨ ¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
Owns(Nono,M1)
¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
¬ Enemy(x,America) ∨ Hostile(x)
¬ Hostile(Nono)
Enemy(Nono,America)
Enemy(Nono,America)



¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)

¬ Criminal(West)

American(West)

¬ American(West) ∨ ¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)

¬ Missile(x) ∨ Weapon(x)

¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)

Missile(M1)

¬ Missile(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)

¬ Missile(x) ∨ ¬ Owns(Nono,x) ∨ Sells(West,x,Nono)

¬ Sells(West,M1,z) ∨ ¬ Hostile(z)

Missile(M1)

¬ Missile(M1) ∨ ¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)

Owns(Nono,M1)

¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)

¬ Enemy(x,America) ∨ Hostile(x)

¬ Hostile(Nono)

Enemy(Nono,America)

Enemy(Nono,America)

□

Forward Chaining
and
Backward Chaining

Inference engine

- 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**

Reasoning with Forward Chaining

Translate the following set of sentences into FOL.

Use forward chaining to prove “Raja is angry.”

a. Roy is hungry

b. if Roy is hungry, she barks.

c. if Roy is barking then Raja is angry.

FOL representation-

a. Roy is hungry

Hungry(Roy)

b. if Roy is hungry, she barks.

Hungry(Roy)→Bark(Roy)

c. if Roy is barking then Raja is angry.

Bark(Roy)→Angry(Raja)

- Forward Chaining-
- From (a), (b) and modus ponens,
- **d. Bark(Roy)**
- From (d), (c) and modus ponens,
- e. Angry(Raja)
  - Hence Proved.

Forward Chaining

- Forward deduction/ Forward reasoning
- Form of reasoning which starts 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.
- **Starts** from known **facts**, triggers all rules whose premises are satisfied, and add their conclusion to the known facts.
- Process repeats until the problem is solved.

Properties of Forward-Chaining

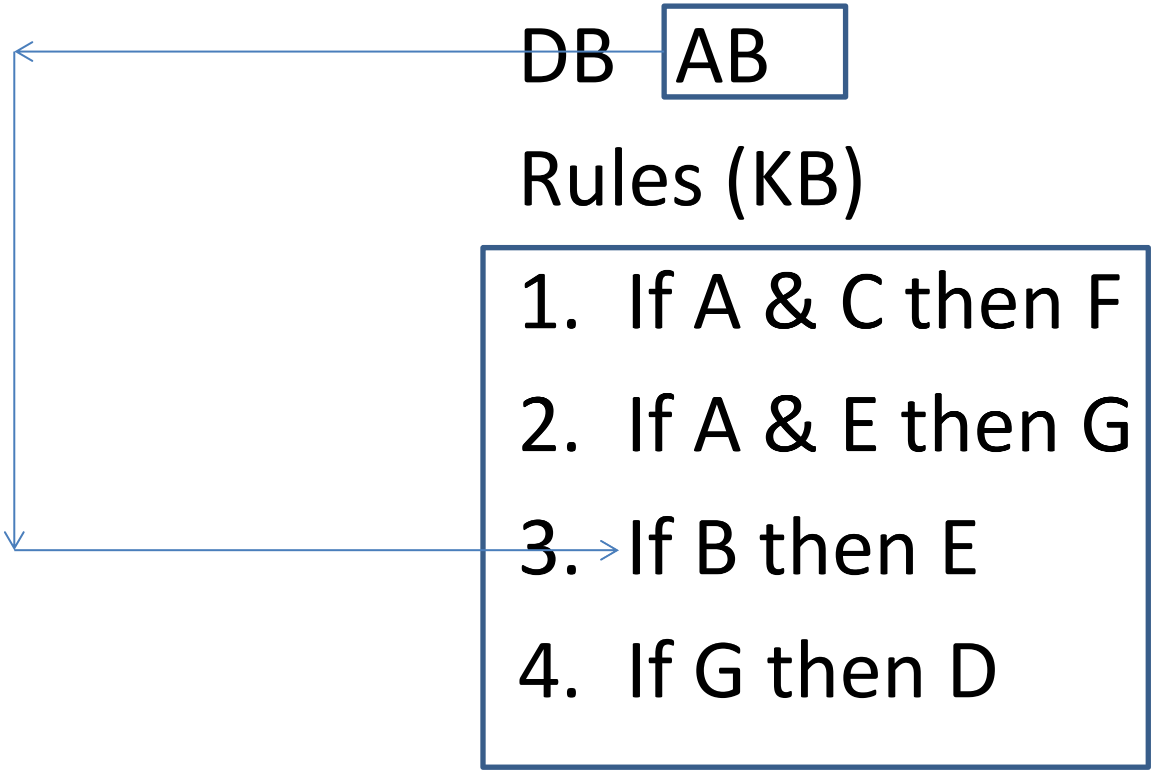
- Bottom-up approach.
- Process of making a **conclusion** based on known facts or data, by starting from the initial state and reaches the goal state.
  - Also called as **data-driven** as we reach to the goal using available data.
- Used in the expert system, such as CLIPS-**C Language Integrated Production System**." , business, and production rule systems.

DB AB

Rules (KB)

1. If A & C then F
2. If A & E then G
3. If B then E
4. If G then D

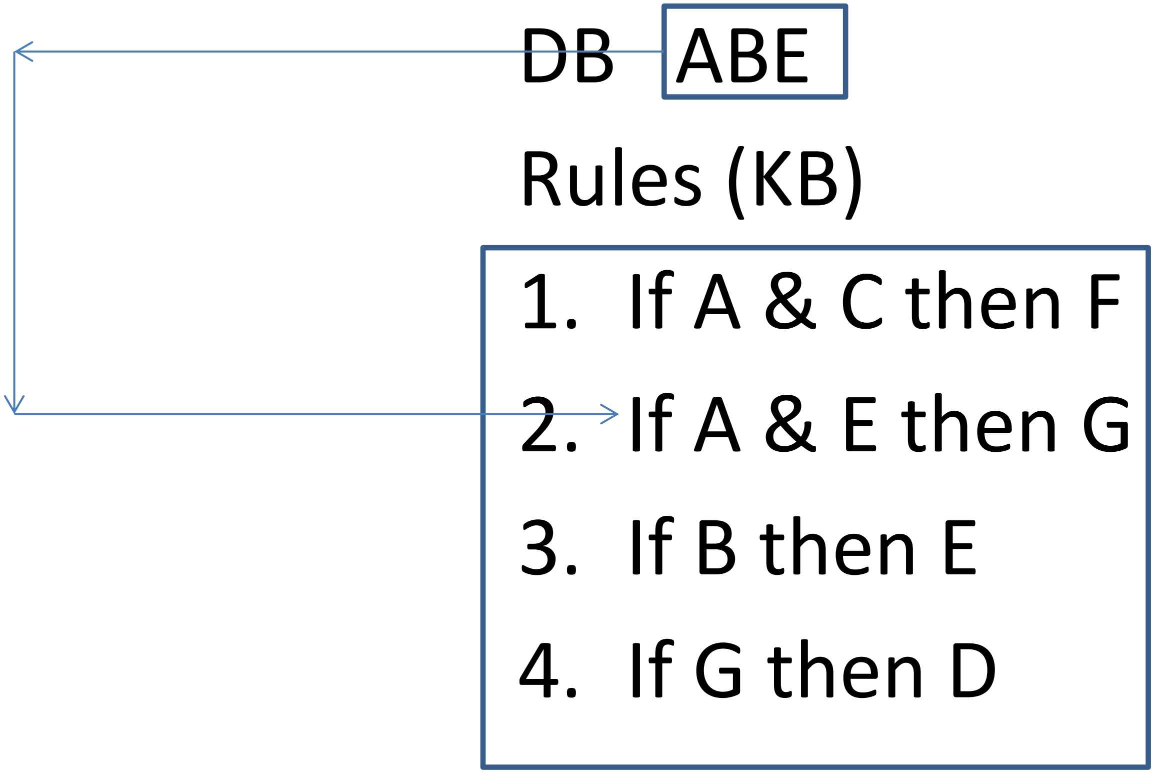
Prove: - If A & B true then D is true



DB AB

Rules (KB)

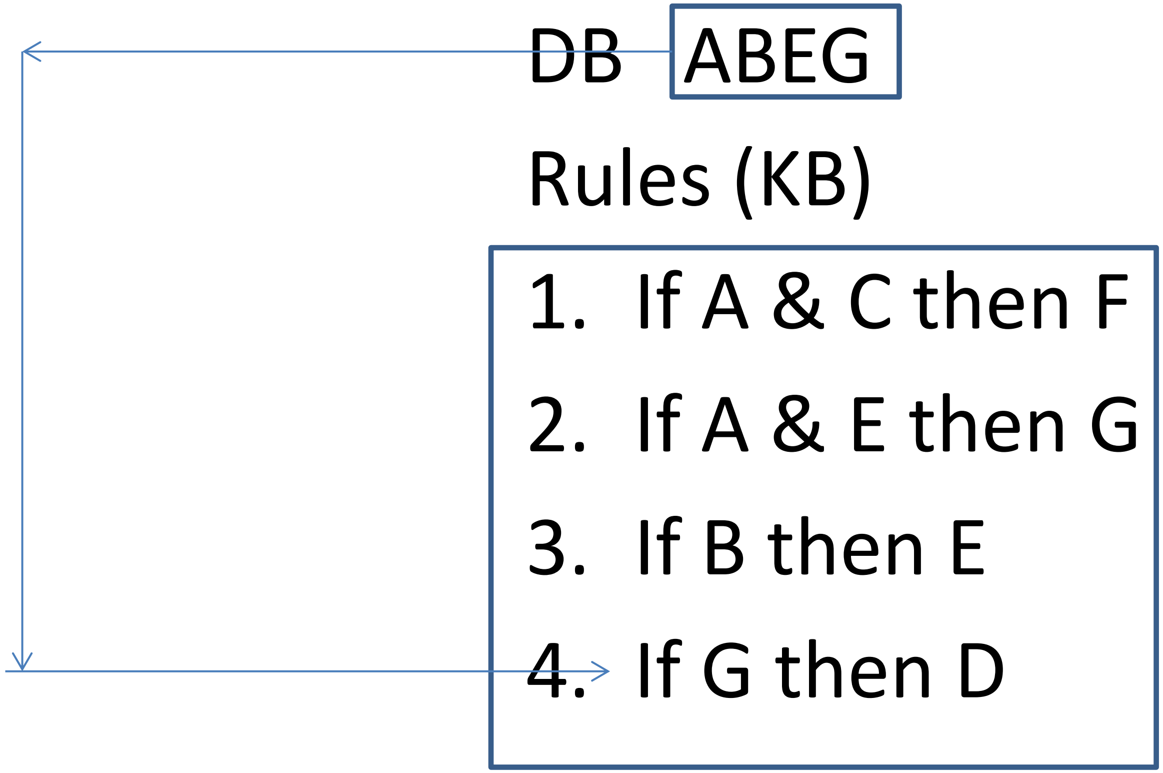
1. If A & C then F
2. If A & E then G
3. If B then E
4. If G then D



DB ABE

Rules (KB)

1. If A & C then F
2. If A & E then G
3. If B then E
4. If G then D



DB ABEG

Rules (KB)

1. If A & C then F
2. If A & E then G
3. If B then E
4. If G then D

DB ABEGD

Rules (KB)

1. If A & C then F
2. If A & E then G
3. If B then E
4. If G then D

Hence Proved

Consider the following famous example which we will use in both approaches:

"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."**

Facts Conversion into FOL:

- 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.
- **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.
- **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)**

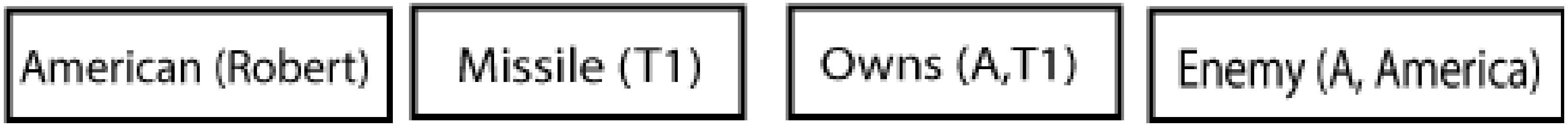
Forward chaining proof

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:

American(Robert), Enemy(A, America), Owns(A, T1), and Missile(T1).

All these facts will be represented as below.



American (Robert)

Missile (T1)

Owns (A,T1)

Enemy (A, America)

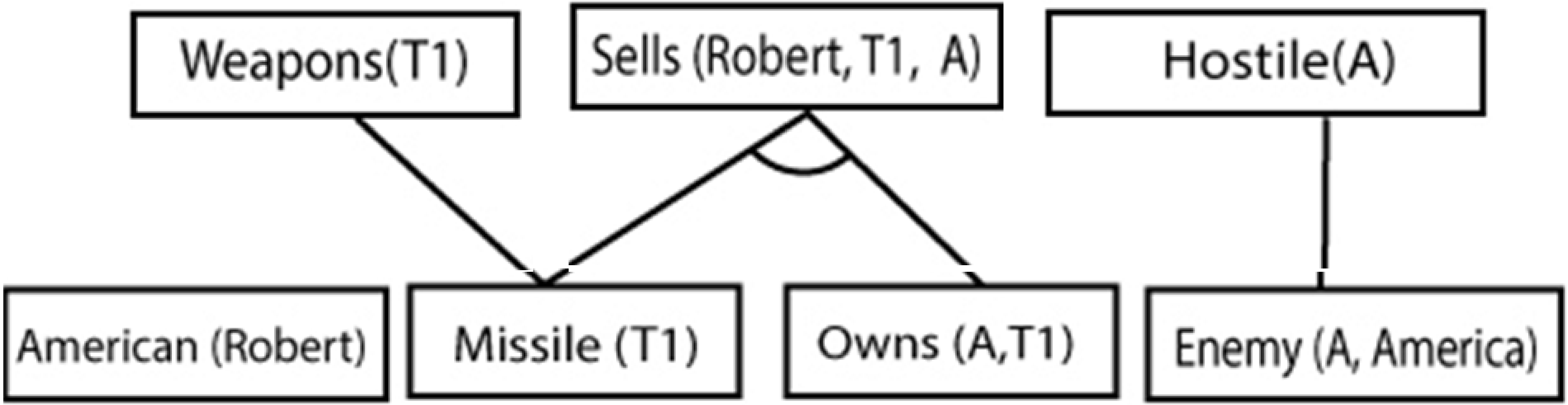
Step-2

- **Facts which we can 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).

Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A)

Missile(p) → Weapons (p)

Enemy(p, America) →Hostile(p)



Weapons(T1)

Sells (Robert, T1, A)

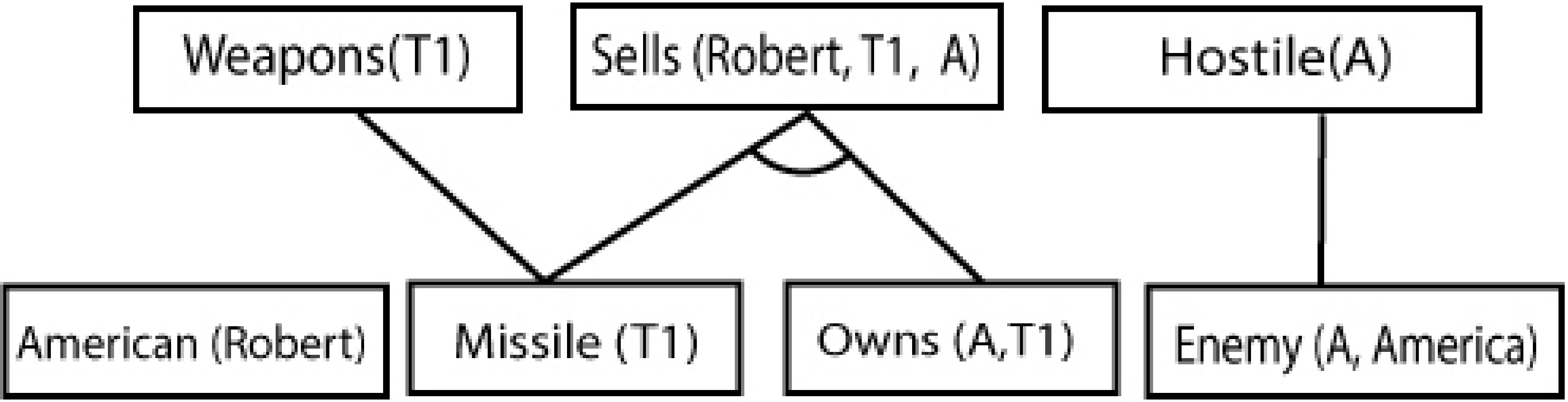
Hostile(A)

American (Robert)

Missile (T1)

Owns (A,T1)

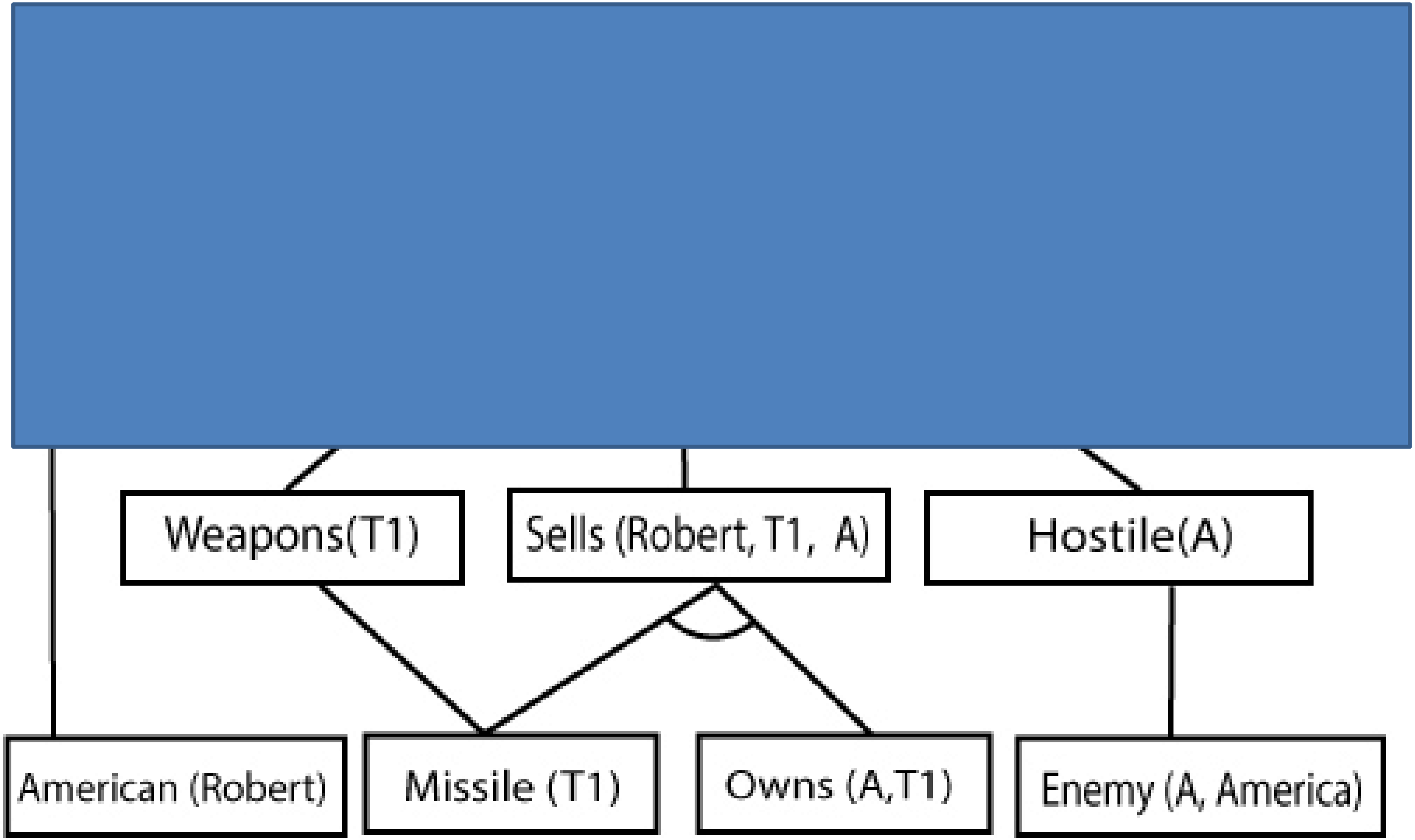
Enemy (A, America)


Weapons(T1)
Sells (Robert, T1, A)
Hostile(A)
American (Robert)
Missile (T1)
Owns (A,T1)
Enemy (A, America)

Step-3

At 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.

$\text{American }(p) \land \text{weapon}(q) \land \text{sells }(p, q, r) \land \text{hostile}(r) \rightarrow \text{Criminal}(p)$ ...(1)



Translate the following set of sentences into FOL. Use forward chaining to

prove Fred is at museum.

- a. Fred is a collie.
- b. Sam is Fred’s master.
- c. The day is Saturday
- d. It is cold on Saturday.
- e. Fred is trained.
- f. Trained collies are good dogs.
- g. If a dog is a good dog and has a master and at some place, then he will be with his master.
- h. If the day is Saturday and the day is cold, then Sam is at the museum.

FOL Representation-

a. Fred is a collie.

Collie(Fred)

b. Sam is Fred’s master.

Master(Sam, Fred)

c. The day is Saturday

Day(Saturday)

d. It is cold on Saturday.

Cold(Saturday)

e. Fred is trained.

Trained(Collie)

f. Trained collies are good dogs.

∀ x Trained(x) ∩ Collie(x) →GoodDog(x)

g. Good dogs who have a master and is at some place, **then** they are always with their maste

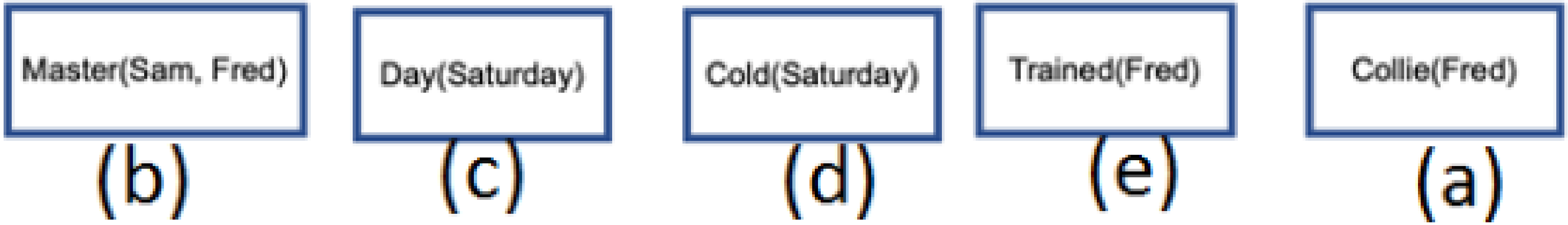
∀ x,y,z GoodDog(x) ∩ Master(y,x) ∩ At(y,z)→ At(x,z),good dog &master has a dog&dog is with the master

h. If the day is Saturday and the day is cold, then Sam is at the museum.

Day(Saturday) ∩ Cold(Saturday) → At(Sam, Museum)

$\forall\, x\ \text{Trained}(x) \cap \text{Collie}(x) \rightarrow \text{GoodDog}(x)$

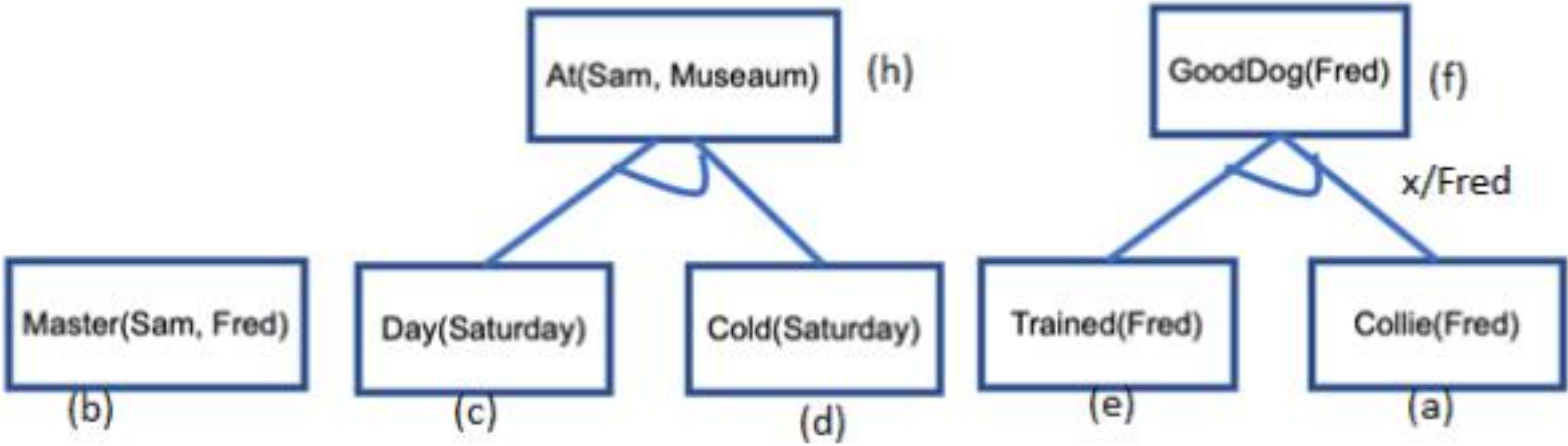
$\text{Day}(\text{Saturday}) \cap \text{Cold}(\text{Saturday}) \rightarrow \text{At}(\text{Sam}, \text{Museum})$



∀ x Trained(x) ∩ Collie(x) →GoodDog(x)

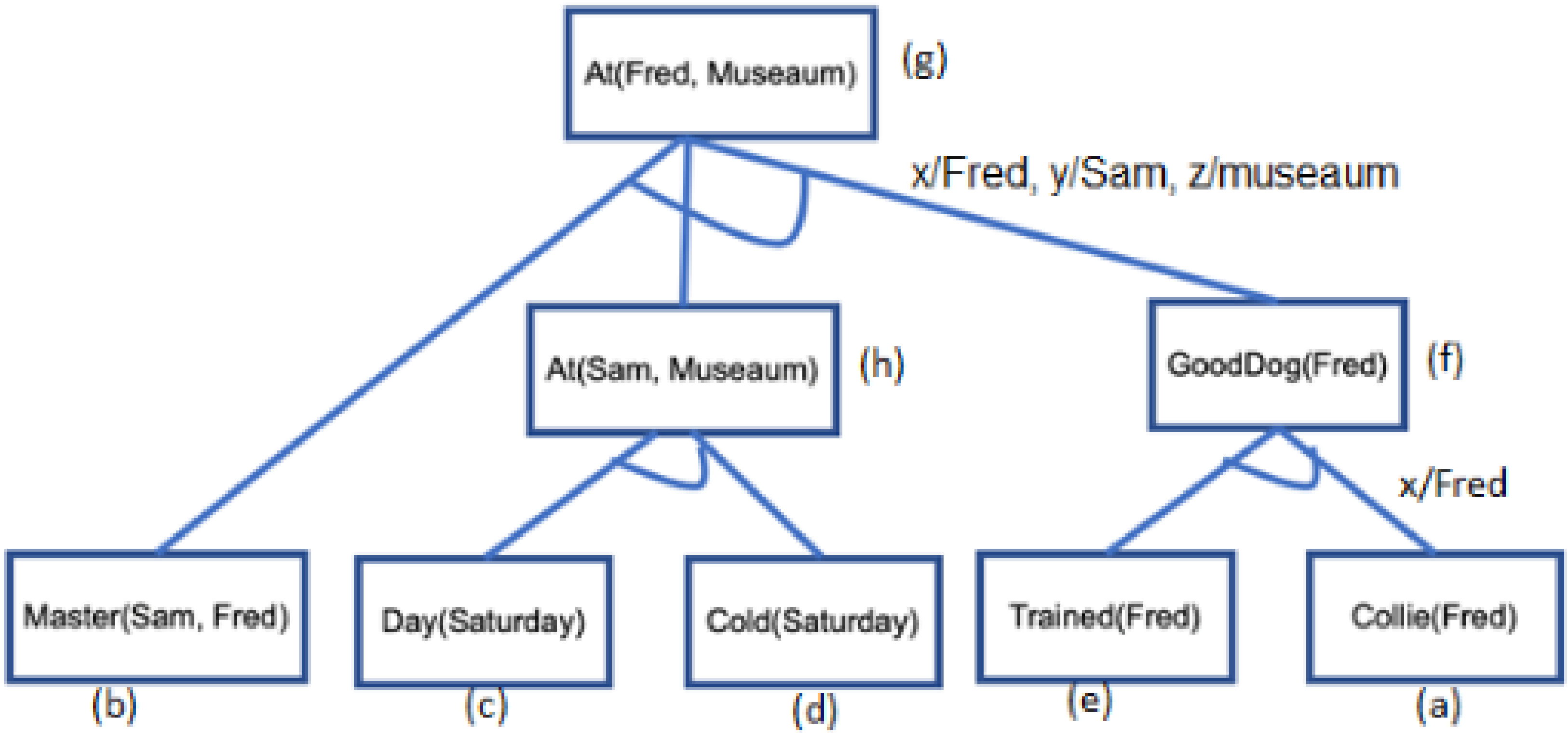
Day(Saturday) ∩ Cold(Saturday) → At(Sam, Museum)

Step 2-



∀ x,y,z GoodDog(x) ∩ Master(y,x) ∩ At(y,z)→ At(x,z),good dog &master has a dog&dog is with the master

Step 3-



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.***

Example:

In backward-chaining, we will use the same above example, and will rewrite all the rules.

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

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

Missile(T1) ....(3)

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

Missile(p) → Weapons (p) ....(5)

Enemy(p, America) →Hostile(p) ....(6)

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

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

Backward-Chaining proof:

- In Backward chaining, we will start with our goal predicate, which is **Criminal(Robert)**, and then infer further rules.

Step-1:

- At the first step, we will take the goal fact.
  - Goal fact → Infer other facts,
  - Lastly, we will prove those facts true.

Goal fact is "Robert is Criminal," so following is the predicate of it.

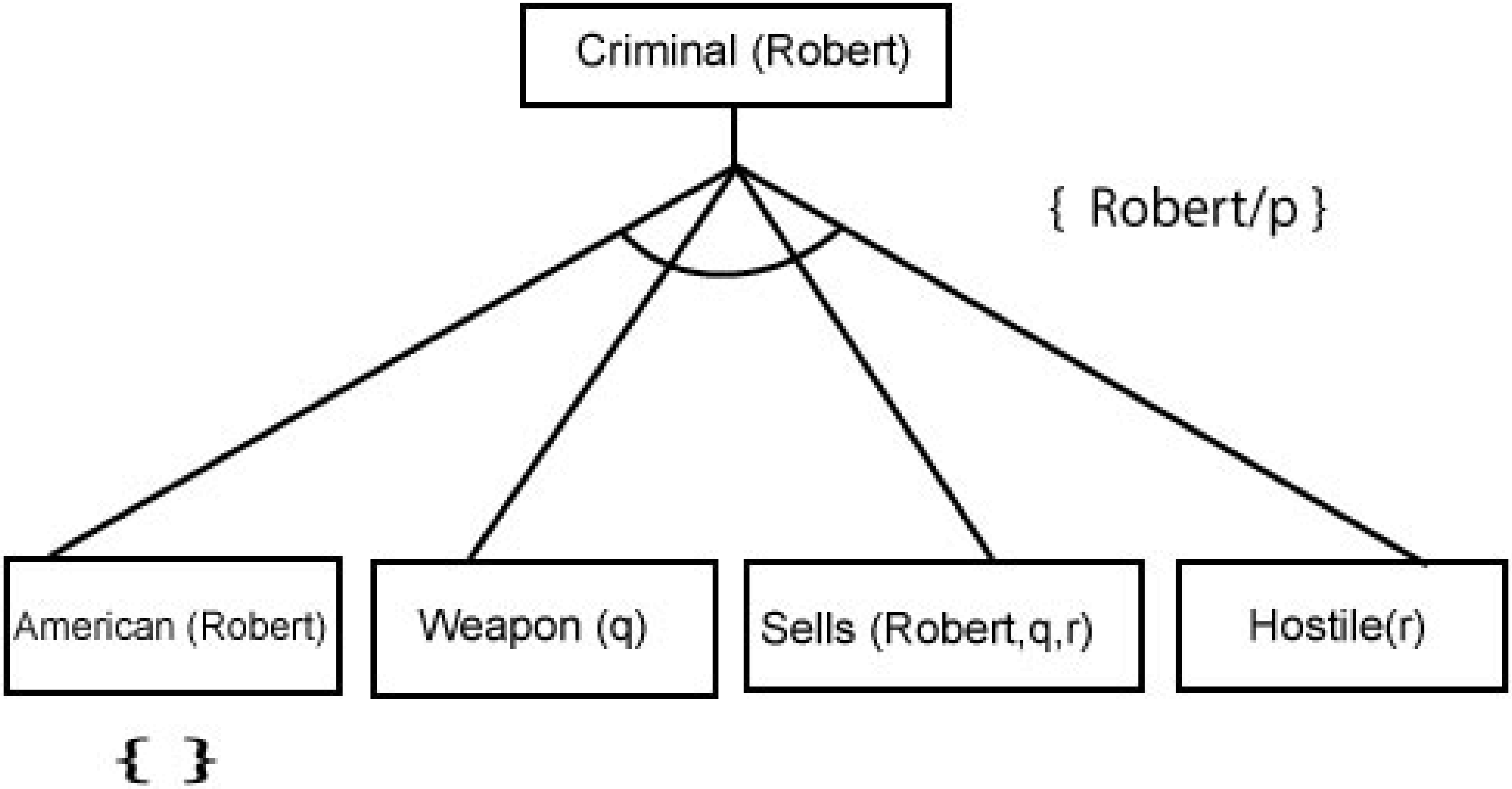
Criminal (Robert)

Step-2

- Infer other facts from goal fact which satisfies the rules.
  - Rule-1, the goal predicate Criminal (Robert) is present with substitution {Robert/P}.
  - Add all the conjunctive facts below the first level and will replace p with Robert.

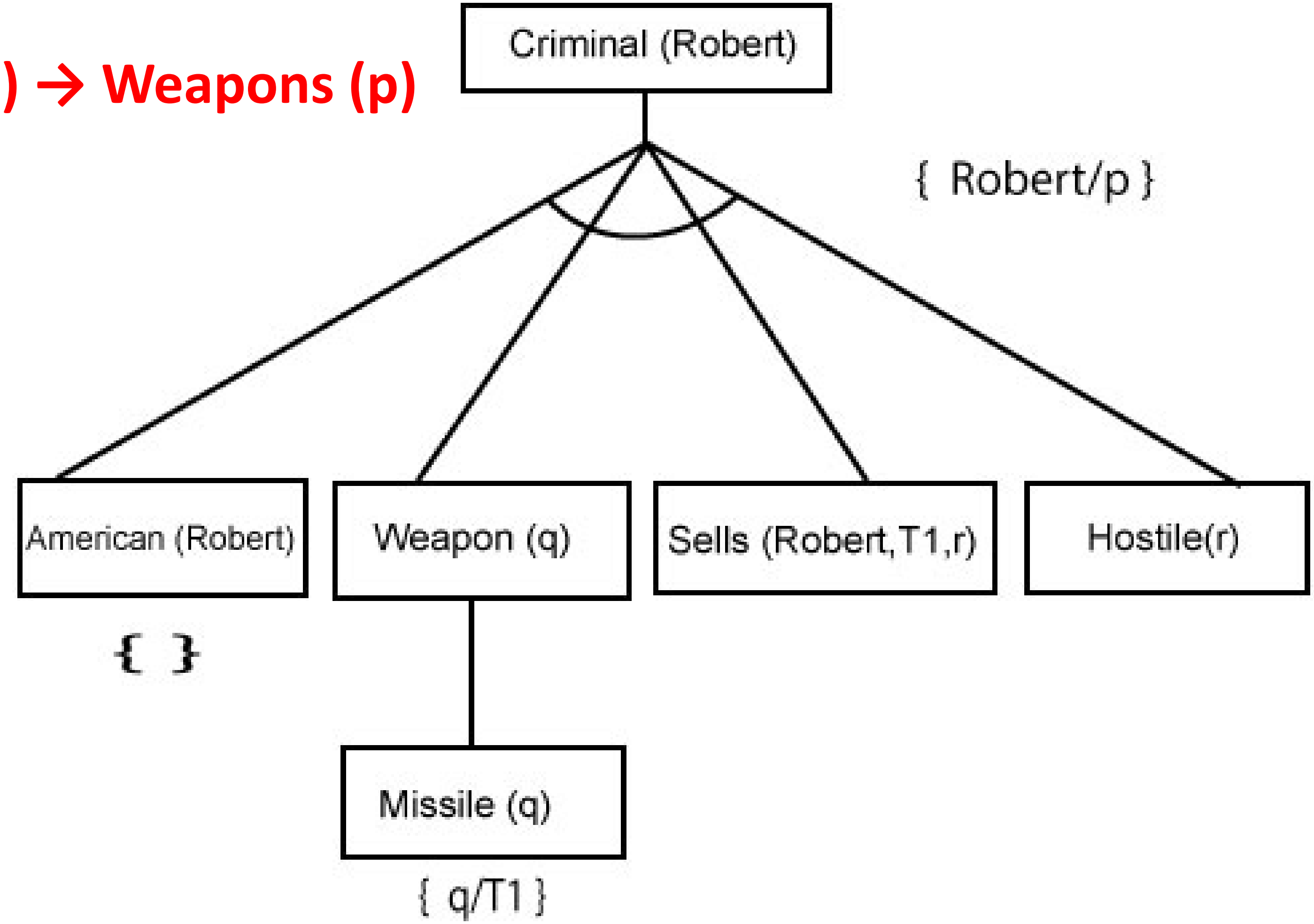
American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p)

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


Criminal (Robert)
{ Robert/p }
American (Robert)
Weapon (q)
Sells (Robert,q,r)
Hostile(r)
{ }

Step-3:t 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.

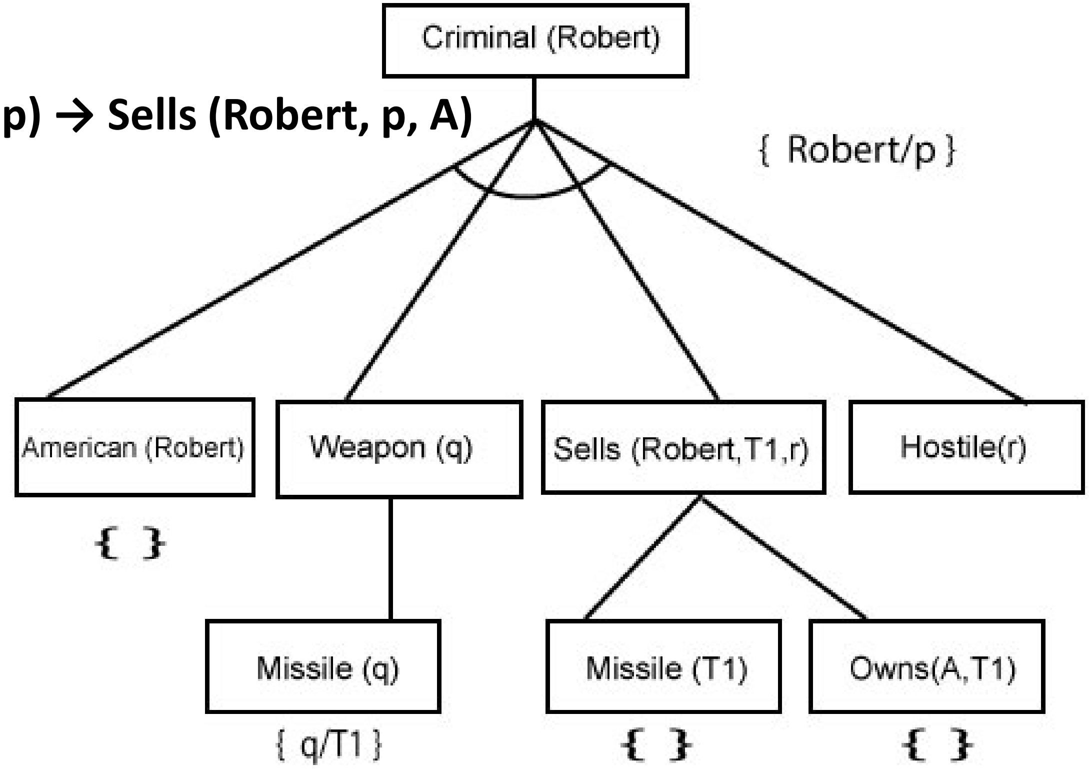
Missile(p) → Weapons (p)



Step-4:

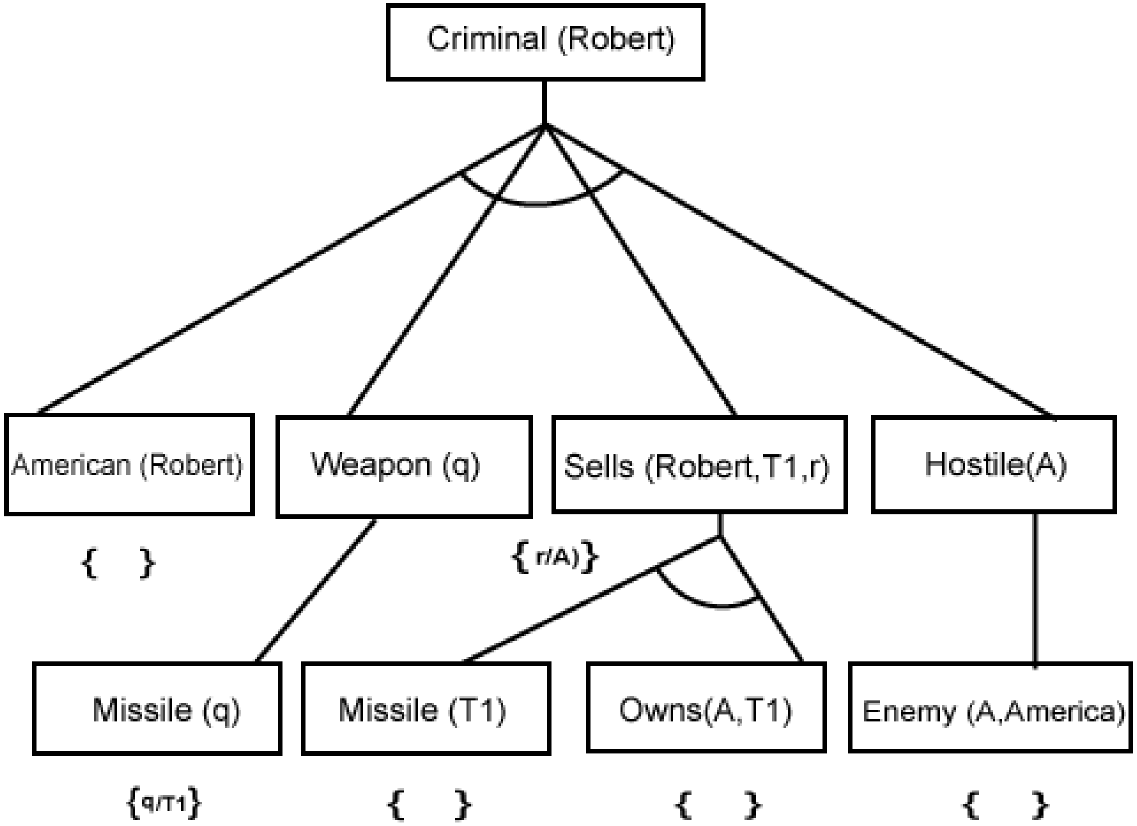
- At step-4, we can infer facts Missile(T1) and Owns(A, T1) form Sells(Robert, T1, r) which satisfies the **Rule- 4**, with the substitution of A in place of r. So these two statements are proved here.

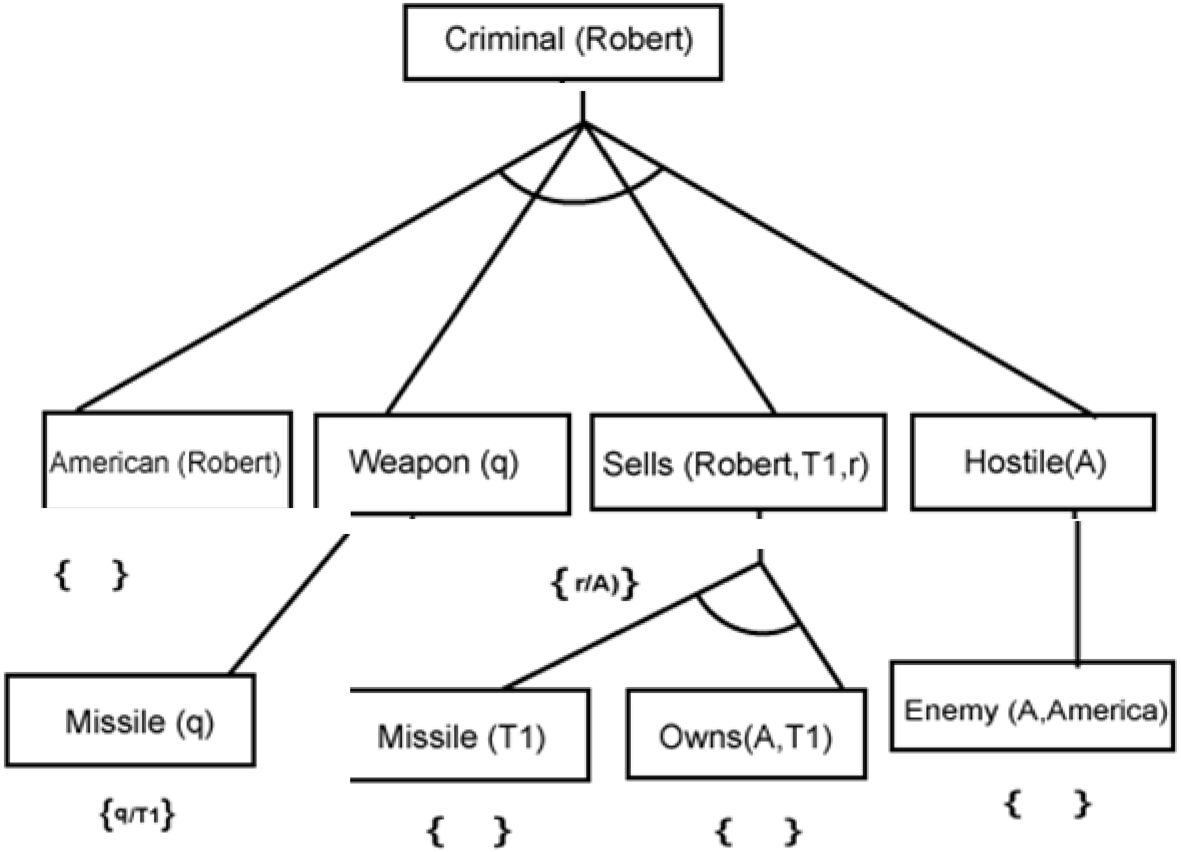
∀ p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A)

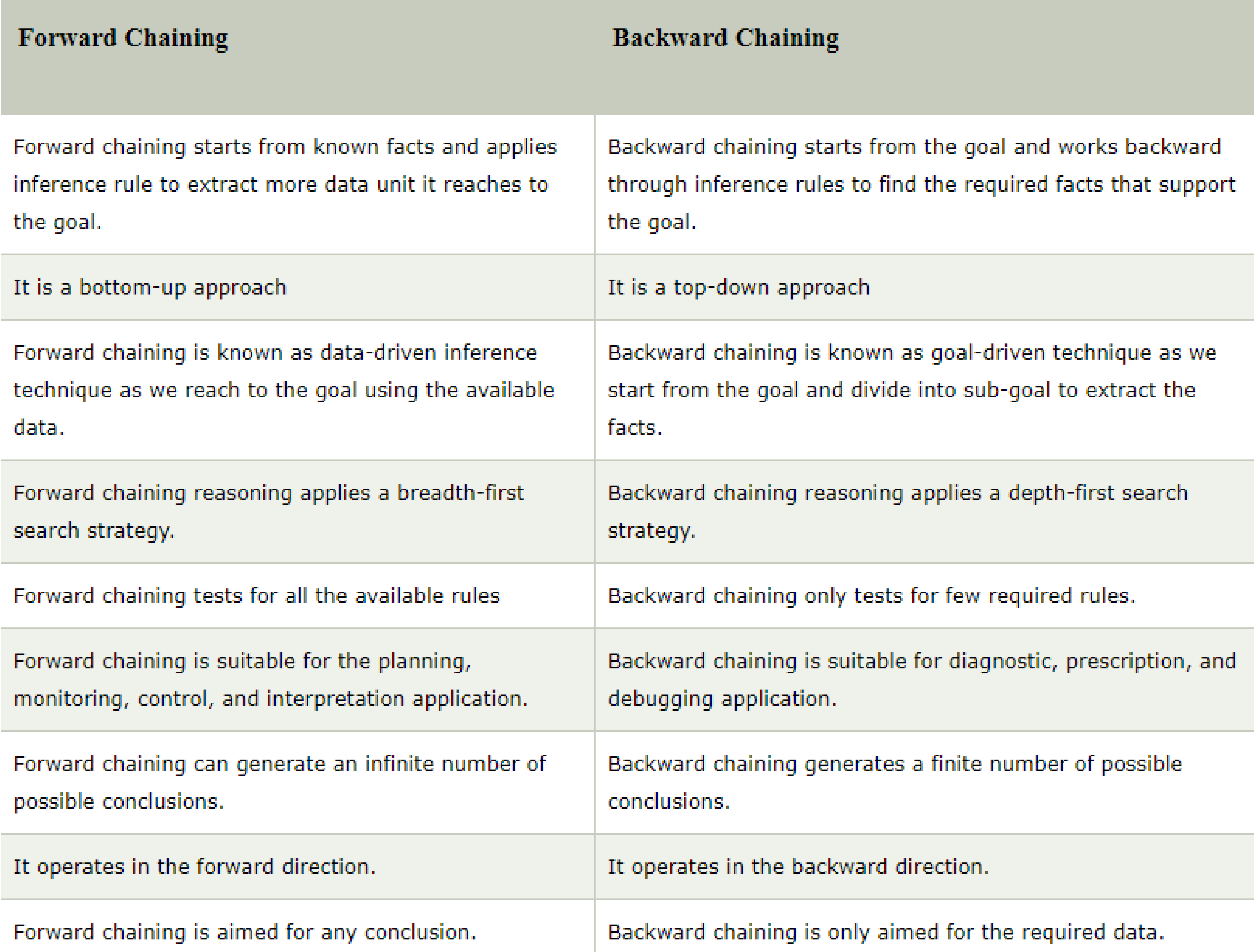


Step-5:

Infer the fact **Enemy(A, America)** from **Hostile(A)** which satisfies Rule- 6. Hence all the statements are proved true using backward chaining.



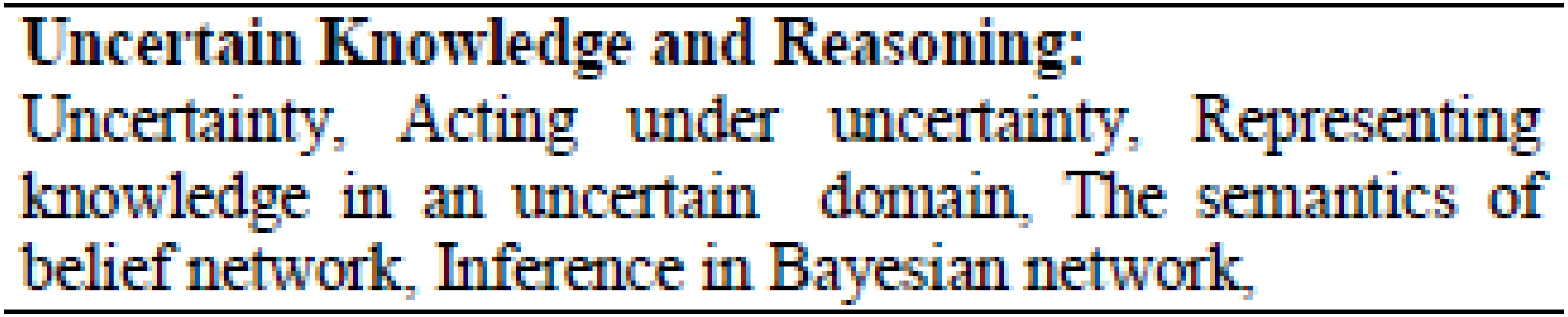

Criminal (Robert)
American (Robert)
Weapon (q)
Sells (Robert,T1,r)
Hostile(A)
{ }
{ r/A)}
Missile (q)
Missile (T1)
Owns(A,T1)
Enemy (A,America)
{q/T1}
{ }
{ }
{ }



Forward Chaining	Backward Chaining
Forward chaining starts from known facts and applies inference rule to extract more data unit it reaches to the goal.	Backward chaining starts from the goal and works backward through inference rules to find the required facts that support the goal.
It is a bottom-up approach	It is a top-down approach
Forward chaining is known as data-driven inference technique as we reach to the goal using the available data.	Backward chaining is known as goal-driven technique as we start from the goal and divide into sub-goal to extract the facts.
Forward chaining reasoning applies a breadth-first search strategy.	Backward chaining reasoning applies a depth-first search strategy.
Forward chaining tests for all the available rules	Backward chaining only tests for few required rules.
Forward chaining is suitable for the planning, monitoring, control, and interpretation application.	Backward chaining is suitable for diagnostic, prescription, and debugging application.
Forward chaining can generate an infinite number of possible conclusions.	Backward chaining generates a finite number of possible conclusions.
It operates in the forward direction.	It operates in the backward direction.
Forward chaining is aimed for any conclusion.	Backward chaining is only aimed for the required data.

Answer Set Programming

- (ASP) is a form of declarative programming oriented towards difficult (primarily NP-hard) search problems.
- An answer set program consists of given knowledge formulated as facts, rules **(head(X) :- body(X).)** or constraints.
- The program is loaded by a **solver** and returns a “stable model”.
- This so called ***answer set*** consists of all facts that can be derived using the given rules and constraints.
- A **finite** amount of stable models are generated as solutions, of which one is finally selected.



Uncertain Knowledge and Reasoning:

Uncertainty, Acting under uncertainty, Representing knowledge in an uncertain domain, The semantics of belief network, Inference in Bayesian network,

Problem and plans

Consider a grocery delivery system like BigBasket which is supposed to deliver goods as per the chosen slots to the customers during lockdown phase.

Give some plans for their procurement and delivery work considering uncertainties in the real world.

Uncertainty

- car breaks down
- runs out of fuel,
- Agent gets into an accident,
- there are accidents on the bridge,
- the plane leaves early
- None of these conditions can be deduced, so the plan's success cannot be inferred.

Uncertainty

- Knowledge representation → first-order logic and propositional logic with certainty
  - which means we were sure about the predicates.
  - A→B, which means **if A is true then B is true,**
  - A is not true ?? we cannot express this statement
- **UNCERTAINTY!**
- Represent uncertain knowledge, where we are not sure about the predicates, we need **UNCERTAIN REASONING OR PROBABILISTIC REASONING.**

Causes of uncertainty

- Following are some leading causes of uncertainty to occur in the real world.
  - Information occurred from unreliable sources.
  - Experimental Errors
  - Equipment fault
  - Temperature variation
  - Climate change.

Reasons which summarize uncertainty

- **Laziness**: It is too much work to list the complete set of antecedents or consequents needed to ensure an exceptionalness rule and too hard to use such rules.

- **Theoretical ignorance**: There may not be any complete theory for the domain. E.g. Medical Science

- **Practical ignorance**: Even if we know all the rules, we might be uncertain about the problem instance in hand. E.g. some particular patient is tested because not all the necessary tests have been or can be run.

Eg:- In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players."

- **NEED OF PROBABILISTIC REASONING IN AI:**
  - When there are unpredictable outcomes.
  - When specifications or possibilities of predicates becomes too large to handle.
  - When an unknown error occurs during an experiment.
- In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge:
- **Bayes' rule**
- **Bayesian Statistics**

Probability

- Chance that an uncertain event will occur.
- Numerical measure of the likelihood that an event will occur.
  - The value of probability always remains between 0 and 1 that represent ideal uncertainties.
- $0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.
  - P(A) = 0, indicates **total uncertainty** in an event A.
  - P(A) =1, indicates **total certainty** in an event A.

$$\textbf{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

Conditional probability:

- Probability of occurring an event when another event has already happened.
  - Ex:- Calculate the event A when event B has already occurred, "**the probability of A under the conditions of B**", it can be written as:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

- **Where P(*A*⋀*B*)= Joint probability of A and B**
- **P(B)= Marginal probability of B.**
- **If the probability of A is given and we need to find the probability of B,**

$$P(B|A) = \frac{P(A \wedge B)}{P(A)}$$

Example

- In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?
- **Solution:**
- Let, A is an event that a student likes Mathematics
- B is an event that a student likes English.

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{0.4}{0.7} = 57\%$$

- **Hence, 57% are the students who like English also like Mathematics.**

Bayes' theorem in Artificial intelligence

Bayes' theorem/ Bayes' rule/ Bayes' law/Bayesian reasoning, which **determines the probability of an event with uncertain knowledge.**

- Way to calculate the value of P(B|A) with the knowledge of P(A|B).
- Allows updating the probability prediction of an event by observing new information of the real world.

Bayesian network

- Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network.

Following are some applications of Bayes' theorem:

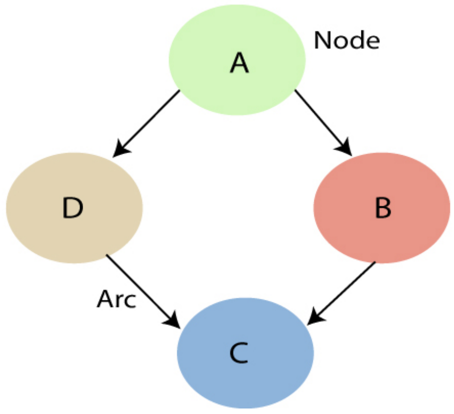
- It is used to calculate the next step of the robot when the already executed step is given.
- Bayes' theorem is helpful in weather forecasting.
- It can solve the Monty Hall problem(brain teaser puzzle)
- It can also be used in various tasks including
  - **prediction, anomaly detection,**
  - **diagnostics, automated insight,**
  - **reasoning, time series prediction**, and
  - **decision making under uncertainty**

Bayesian networks are probabilistic, because these networks are built from a **probability distribution**, and also use probability theory for prediction and anomaly detection.

Bayesian Network

Can be used for building models from data and experts opinions, and it consists of two parts:

- **Directed Acyclic Graph (DAG)**
- **Table of conditional probabilities.**



Each node in the Bayesian network has **condition probability distribution $P(X_i \mid Parent(X_i))$**, which determines the **effect of the parent on that node.**

Bayesian network is based on **Joint probability** distribution and conditional probability

- If we have variables x1, x2, x3,....., xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as Joint probability distribution.

$$\mathbf{P(X_i \mid X_{i-1}, \ldots\ldots\ldots, X_1) = P(X_i \mid Parents(X_i))}$$

Explanation of Bayesian network:

Example

Harry installed a new burglar alarm at his home to detect burglary. The alarm reliably responds at detecting a burglary but also responds for minor earthquakes. Harry has two neighbours David and Sophia, who have taken a responsibility to inform Harry at work when they hear the alarm. David always calls Harry when he hears the alarm, but sometimes he gets confused with the phone ringing and calls at that time too. On the other hand, Sophia likes to listen to high music, so sometimes she misses to hear the alarm. Here we would like to compute the probability of Burglary Alarm.

Problem:

- Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called Harry.

List of all events occurring in this network:

- **Burglary (B)**
- **Earthquake(E)**
- **Alarm(A)**
- **David Calls(D)**
- **Sophia calls(S)**

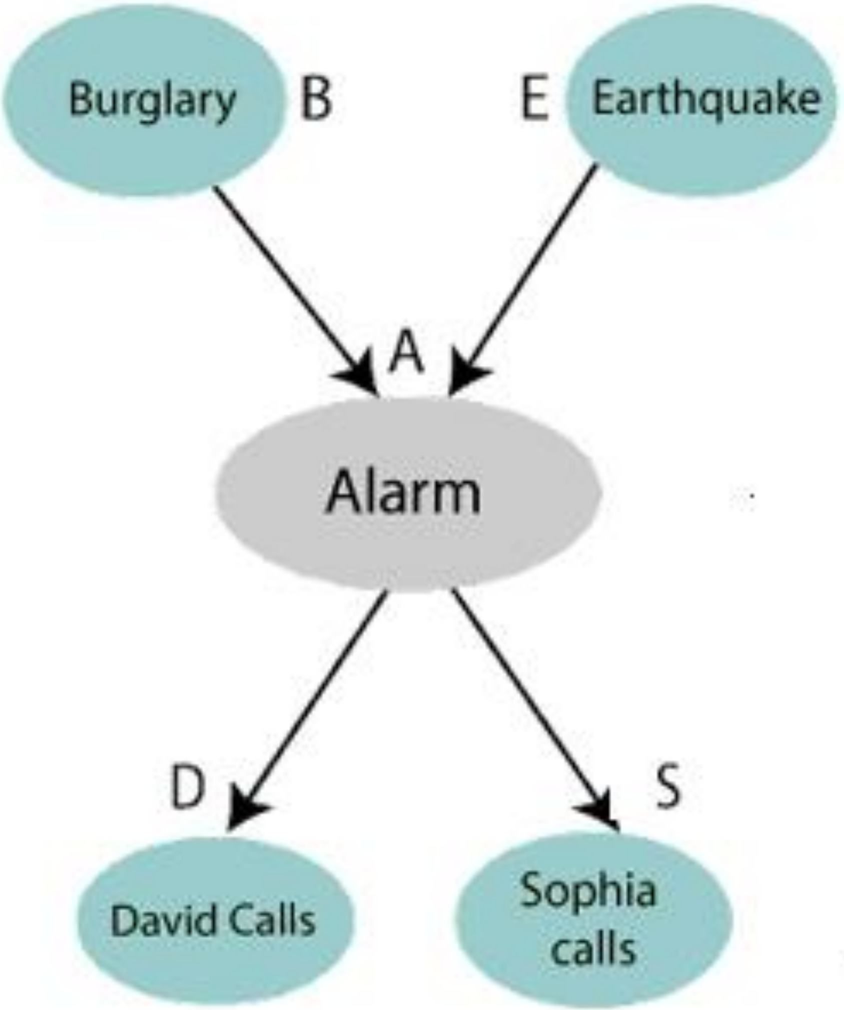
- Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and Davic and Sophia both called Harry.

Find the probability that ‘S’ is true (Sophia has called ‘Harry’), ‘D’ is true (David has called ‘Harry’) when the alarm ‘A’ rang, but no burglary ‘B’ and earthquake ‘E’ has occurred.

- => **P (S, D, A, ~B, ~E)** [where- ,S ,D & A are ‘true’ events and ‘~B’ & ‘~E’ are ‘false’ events]

Solution

- Burglary and earthquake is the parent node of Alarm →Dependencies.
- The network is representing that our assumptions do not directly perceive the burglary



Note: The values mentioned below are neither calculated nor computed. They have observed values.

Let's take the observed probability for the Burglary and earthquake

component:

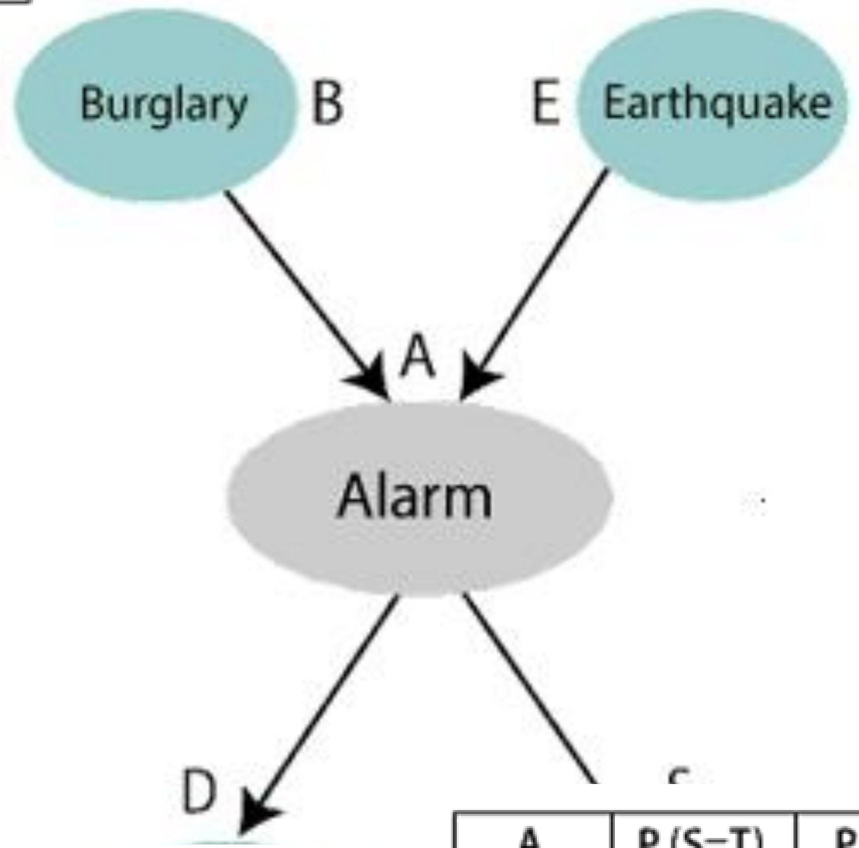
- P(B= True) = 0.002, which is the probability of burglary.
- P(B= False)= 0.998, which is the probability of no burglary.
- P(E= True)= 0.001, which is the probability of a minor earthquake
- P(E= False)= 0.999, Which is the probability that an earthquake not occurred.

- We can provide the conditional probabilities as per the below tables:

- The conditional distributions for each node are given as conditional probabilities table or CPT.
- Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.
- In CPT, a boolean variable with k boolean parents contains 2^K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values

T	0.002
F	0.998

T	0.001
F	0.999



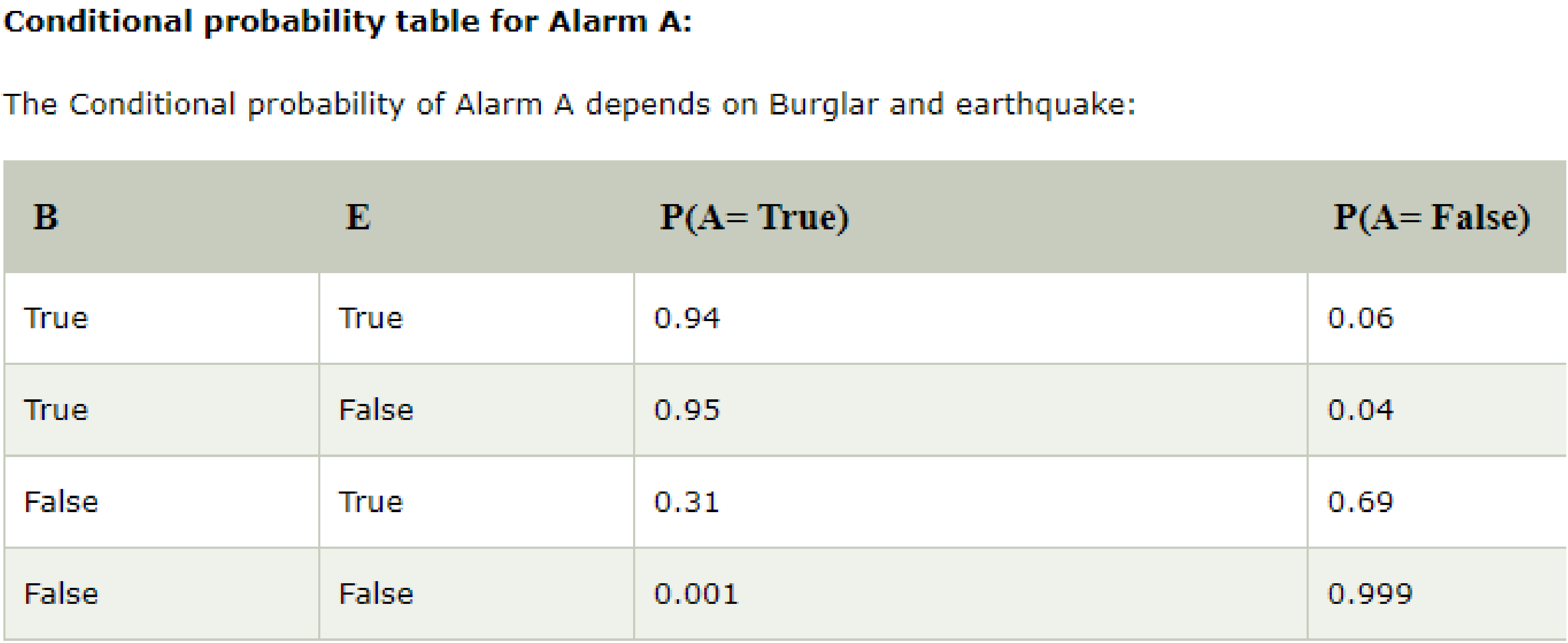
A	P (D=T)	P (D=F)
T	0.91	0.09
F	0.05	0.95

A	P (S=T)	P (S=F)
T	0.75	0.25
F	0.02	0.98

We can write the events of problem statement in the form of probability: **P[D, S, A, B, E]**, can rewrite the above probability statement using joint probability distribution:

$$P[D, S, A, B, E]$$

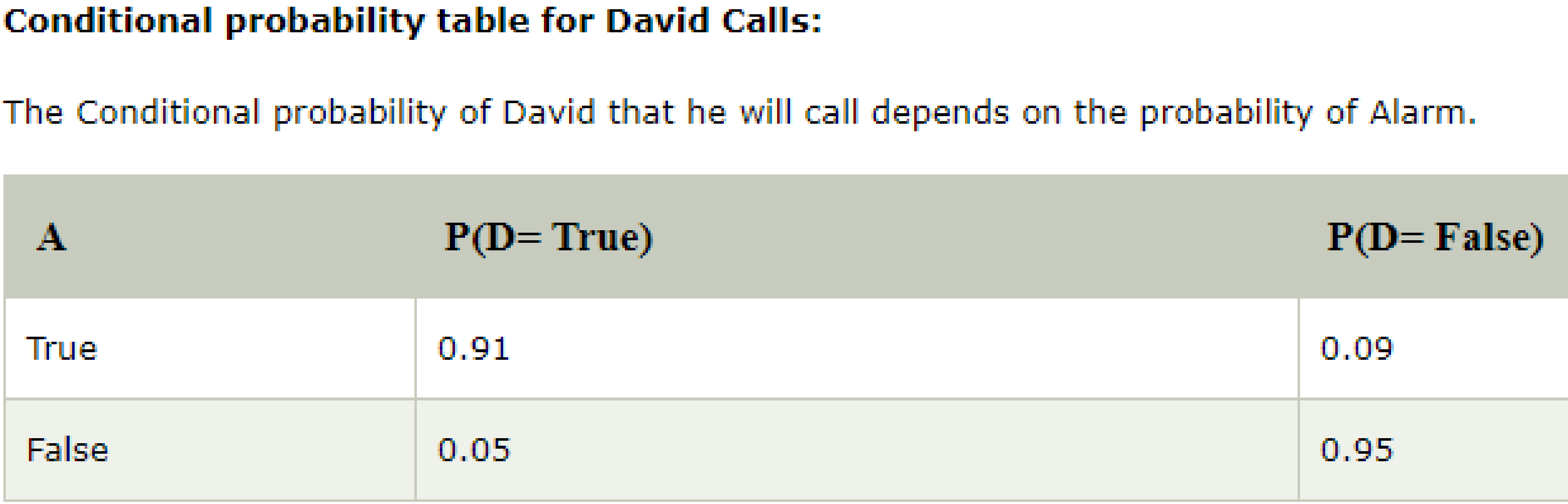
$$= P[D \mid A]. P[S \mid A]. P[A \mid B, E]. P[B]. P[E]$$



Conditional probability table for Alarm A:

The Conditional probability of Alarm A depends on Burglar and earthquake:

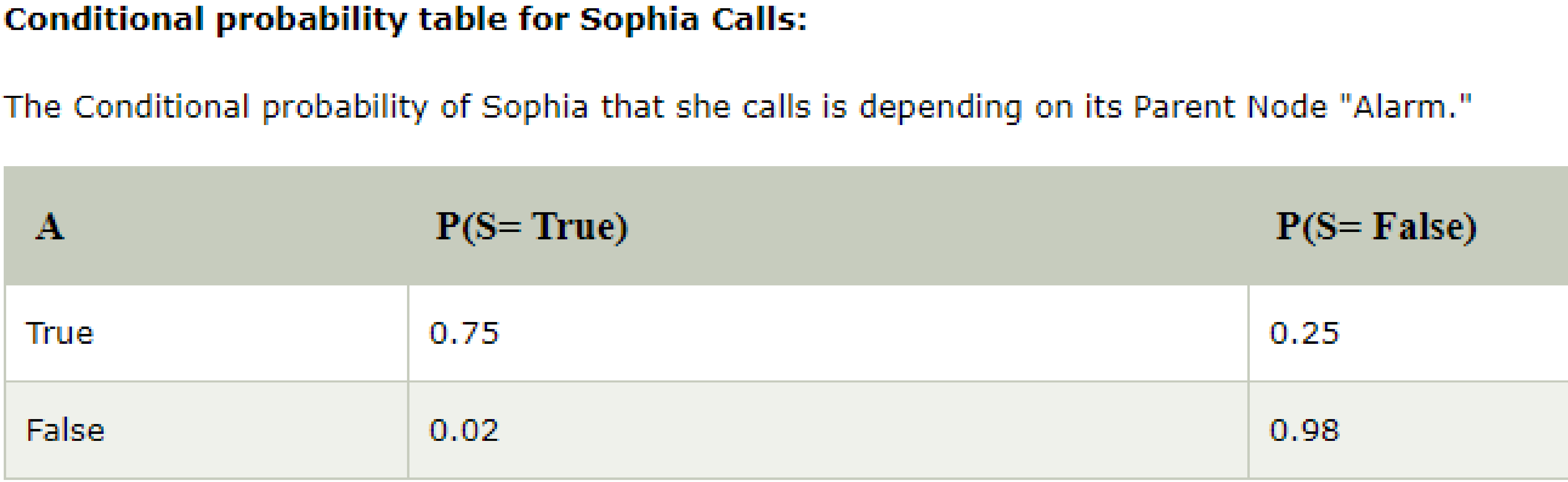
B	E	P(A= True)	P(A= False)
True	True	0.94	0.06
True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999



Conditional probability table for David Calls:

The Conditional probability of David that he will call depends on the probability of Alarm.

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95



Conditional probability table for Sophia Calls:

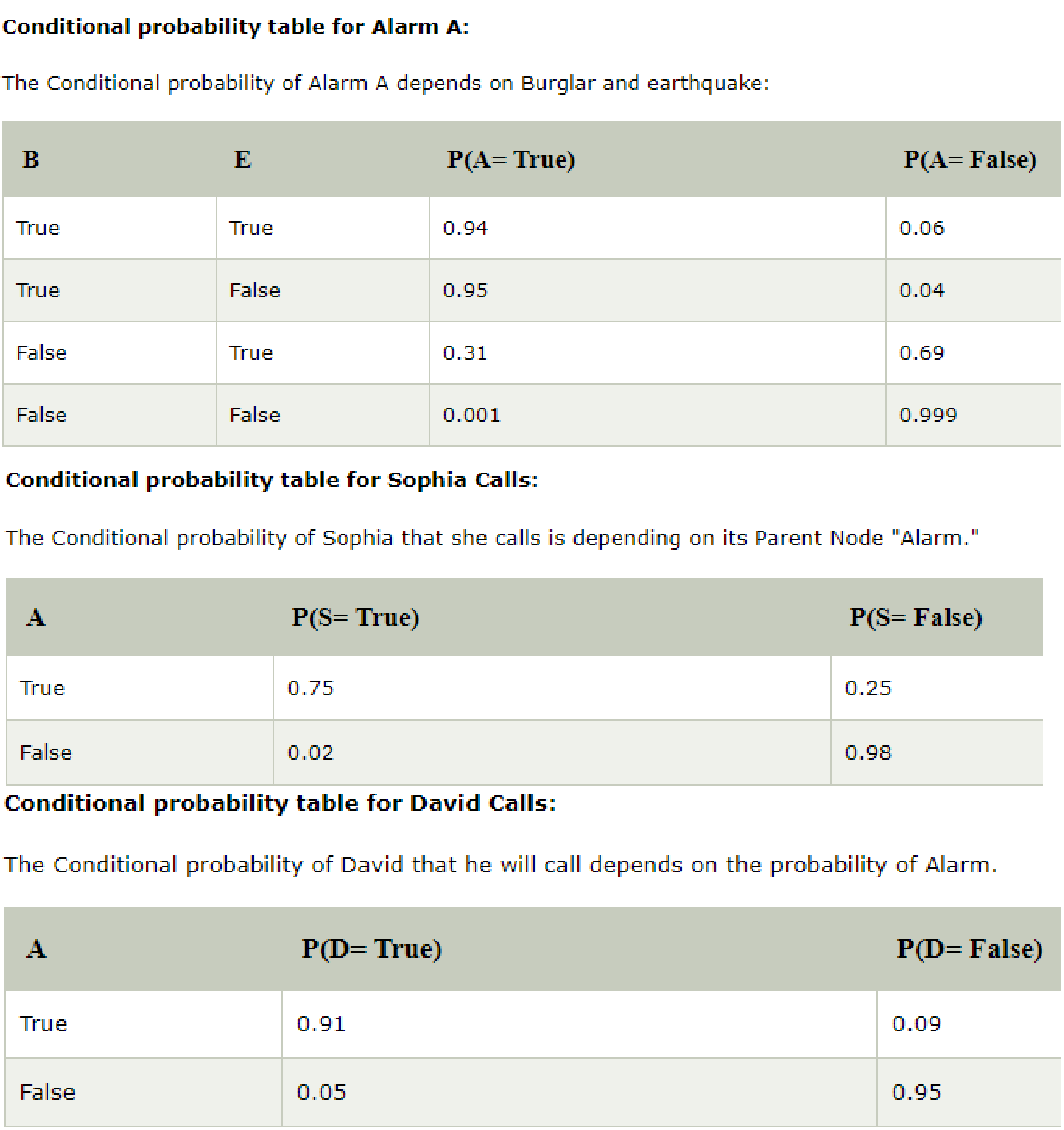
The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm."

A	P(S= True)	P(S= False)
True	0.75	0.25
False	0.02	0.98

Find the probability that ‘S’ is true (Sophia has called ‘Harry’), ‘D’ is true (David has called ‘Harry’) when the alarm ‘A’ rang, but no burglary ‘B’ and earthquake ‘E’ has occurred.

- => **P (S, D, A, ~B, ~E)** [where- ,S ,D & A are ‘true’ events and ‘~B’ & ‘~E’ are ‘false’ events]

- From the formula of joint distribution, we can write the problem statement in the form of probability distribution:
- **P(S, D, A, ¬B, ¬E) = P (S|A) *P (D|A)*P (A|¬B ^ ¬E) *P (¬B) *P (¬E).**
- = 0.75* 0.91* 0.001* 0.998*0.999
- **= 0.00068045.**
- **Hence, a Bayesian network can answer any query about the domain by using Joint distribution.**



Conditional probability table for Alarm A:

The Conditional probability of Alarm A depends on Burglar and earthquake:

B	E	P(A= True)	P(A= False)
True	True	0.94	0.06
True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999

Conditional probability table for Sophia Calls:

The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm."

A	P(S= True)	P(S= False)
True	0.75	0.25
False	0.02	0.98

Conditional probability table for David Calls:

The Conditional probability of David that he will call depends on the probability of Alarm.

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95

The semantics of Bayesian Network:

There are two ways to understand the semantics of the Bayesian network, which is given below:

- **1. To understand the network as the representation of the Joint probability distribution.**
  - It is helpful to understand how to construct the network.
- **2. To understand the network as an encoding of a collection of conditional independence statements.**
  - It is helpful in designing inference procedure