Knowledge-based Agents Planning & Logic

Dr. Pulak Sahoo

Associate Professor

Silicon Institute of Technology

Knowledge-Based Agents - Topics

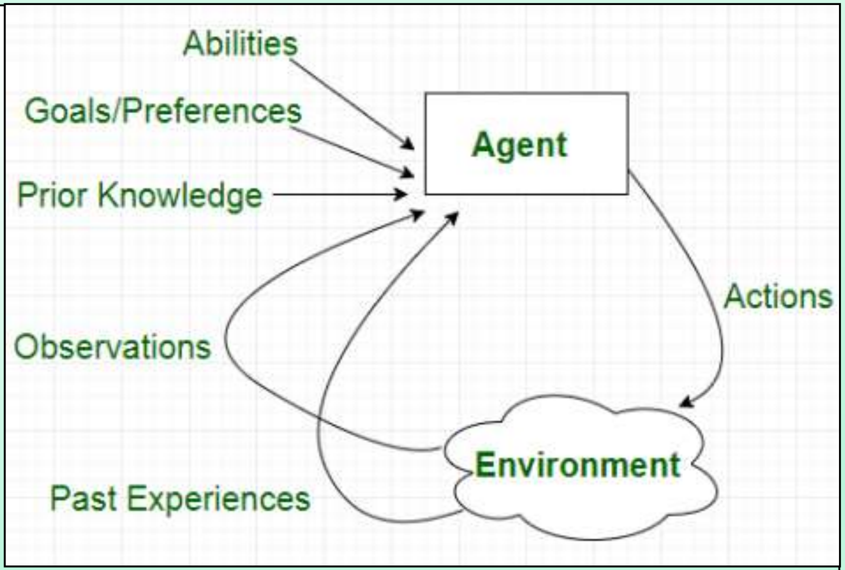
- **Introduction**
- **Knowledge-Based Agents**
- **WUMPUS WORLD Environment**
- **Propositional Logic**
- **First Order Predicate Logic**
- **Forward and Backward Chaining**

Planning - Topics

- **Planning: Introduction to Planning**
- **Planning with State Space Search**
- **Partial Ordered planning**
- **Hierarchical Planning**
- **Conditional Planning**
- **Planning with Operators**

Introduction

- Human beings know things
- This helps them to do things intelligently based on reasoning



- Process of reasoning operates based on **internal representation (storage) of knowledge**
- Same approach is followed by **Knowledge-based Agents**
- **Logic** is a class of representation that supports Knowledge-based Agents
- They can adopt to changes in env. by updating knowledge

Knowledge-based Agents (Design)

- **Central component** – **knowledge base (KB)**
- **Knowledge Base** – Set of sentences expressed in **Knowledge Representation Language**
- **Operations**
  - **TELL** – **Add** new sentence to KB
  - **ASK** – **Query** what is known
- An **KB Agent program** takes a **percept** as **input** & **returns** an **action**
- The **KB** initially contains some **"background knowledge"**
- The **Agent program** does 3 things
  - **TELLs** the **KB** what it **perceives**
  - **ASKs** the **KB** what **action should be performed**
  - **TELLs** the **KB** what **action was chosen** **&** **executes the action**

KB Agents Program

```
Agent KB-Agent (Percept) Returns an action

   Persistent: KB – a knowledge base  // Maintain a KB
   t (time) = 0                       //time is initialized to 0

   // Input percept sequence & time to KB
   TELL ( KB, Make-Percept-Sentence ( percept, t ) )

   // Find suitable action to be taken from KB
   action = ASK (KB, Make-Action-Query ( t ) )

    // Update KB with action corresponding to the percept seq at time t
    TELL (KB, Make-Action-Sentence ( percept, t)

    t = t + 1   // Increment time

   return action  // Return action
```

KB Agents Program

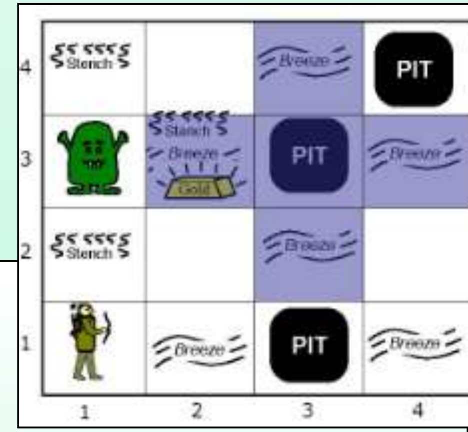
- **Two System building approaches** employed by a **designer** to an empty KB

1. **Declarative approach**
   - **TELL** sentences one-by-one until the agent knows how to operate

2. **Procedural approach**
   - **Encodes** desired behavior directly into program code

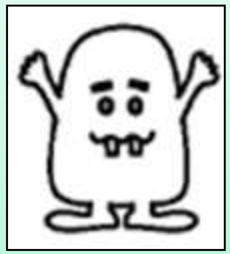
- A successful agent must combine both approaches

The Wumpus World Environment



- **Wumpus World**

- A **cave** containing rooms connected by passageways

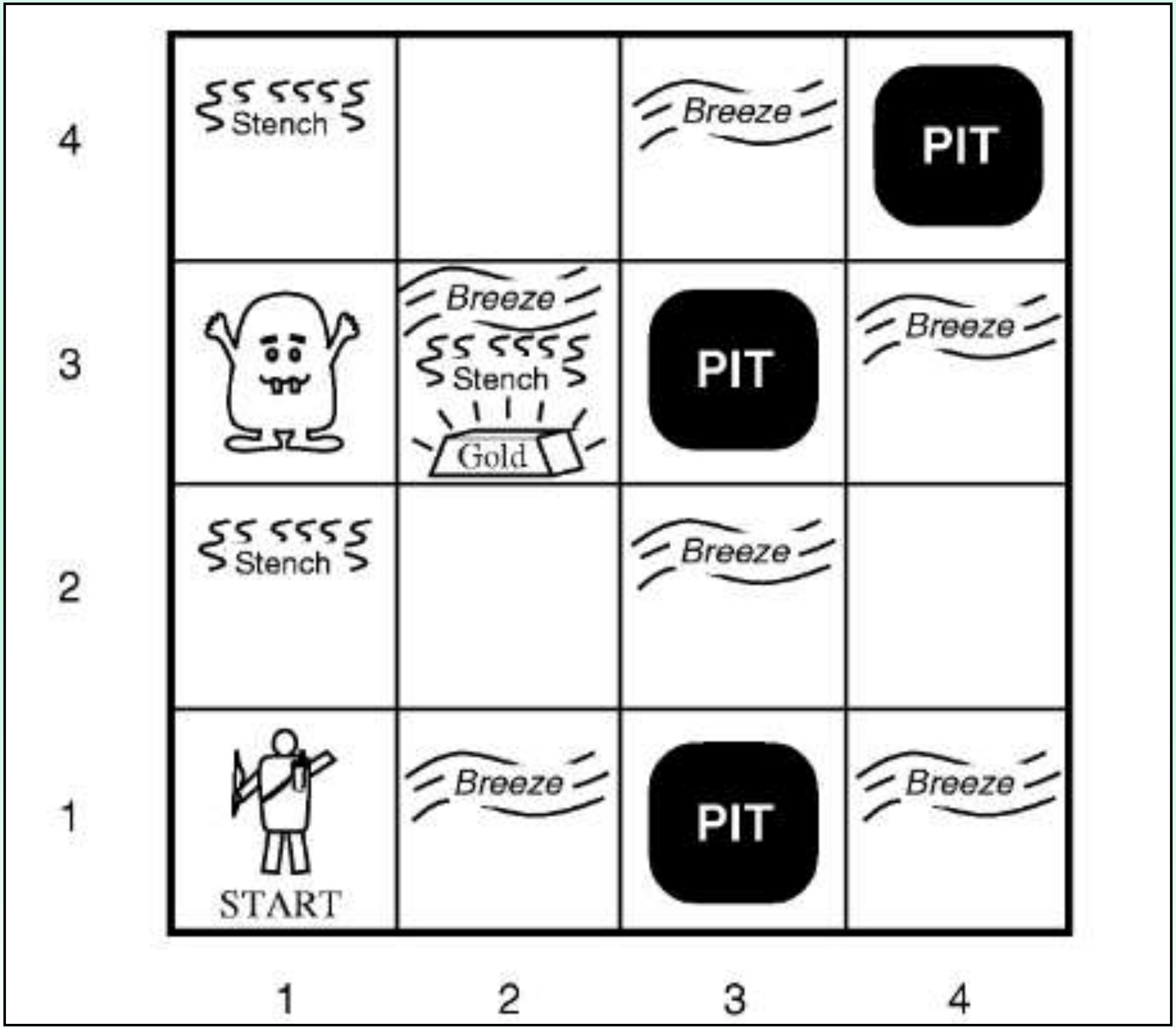
- The **Wumpus (beast)**  hidden in the cave – Eats anyone entering the room

- The **Agent**  has only one arrow to shoot

- Some rooms has bottom-less **pits** 

 to trap anyone entering

- **Only Reward** - Possibility of finding a gold heap 


Stench
Breeze
PIT
Breeze
Breeze
Stench
Gold
PIT
Breeze
Stench
Breeze
START
Breeze
PIT
Breeze
4
3
2
1
1
2
3
4

Task Environment Description - PEAS

- **Performance measure**
  - **+1000** – Coming out of cave with gold
  - **-1000** – Falling into Pit or Eaten by Wumpus
  - **-1** – For each action
  - **-10** – For using the arrow
  - **End of game** – Agent dies or climbs out of cave
- **Env**
  - A **4X4 grid** of **rooms**
  - **Agent** starts in **[1,1]**
  - Location of **Gold** & **Wumpus** chosen randomly (except starting one)
  - Each square (except starting one) can be a pit with probability 0.2

Task Environment Description - PEAS

- **Actuators**
  - **Agent Moves** – ***Forward, TurnLeft, TurnRight***
  - **Death** – Falling into Pit or Eaten by Wumpus
  - **Forward move against wall**– Not allowed
  - **Actions**– **Grab** (pickup gold), **Shoot** (one Arrow), **Climb** (outof cave from [1,1])
  - **End of game** – Agent dies or climbs out of cave

- **Sensors**
  - **Stench**: Perceived in squares **containing** & **adjacent** to **wumpus**
  - **Breeze**: Perceived in squares **adjacent** to a **pit**
  - **Glitter**: Perceived in squares containing **Gold**
  - **Bump**: Perceived when walking into a **Wall**
  - **Kill Wumpus**: Perceived **Scream** anywhere in the cave

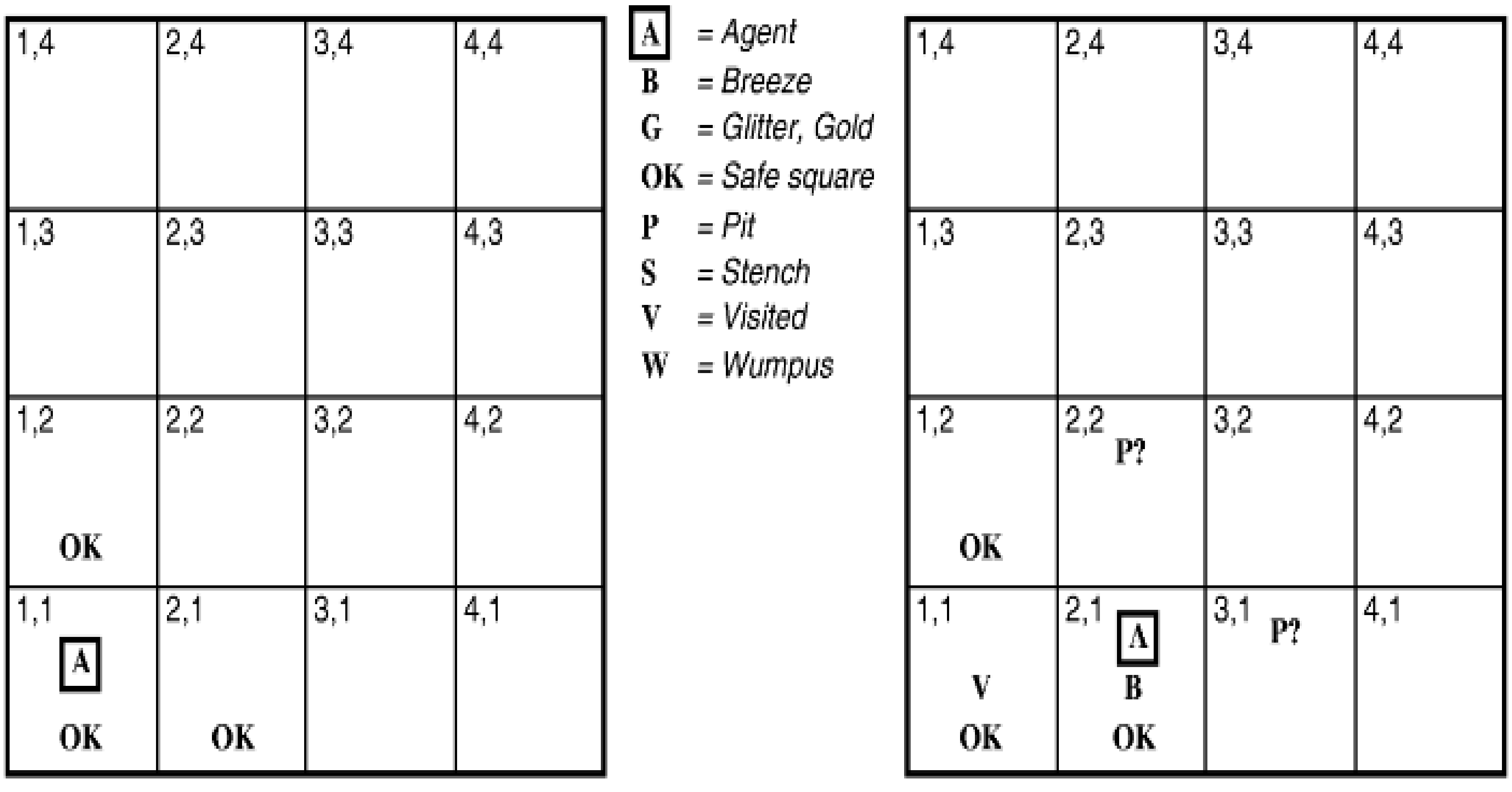
Wumpus World - Steps

- **Challenges for Agent** - Initial ignorance of env configuration (require logical reasoning)

- *Good possibility of agent getting out **with gold***
- *Sometimes, agent will have to **choose** between **empty-hand return** or **death***
- ***21%** times **gold** is in a **pit** or **surrounded by pits***

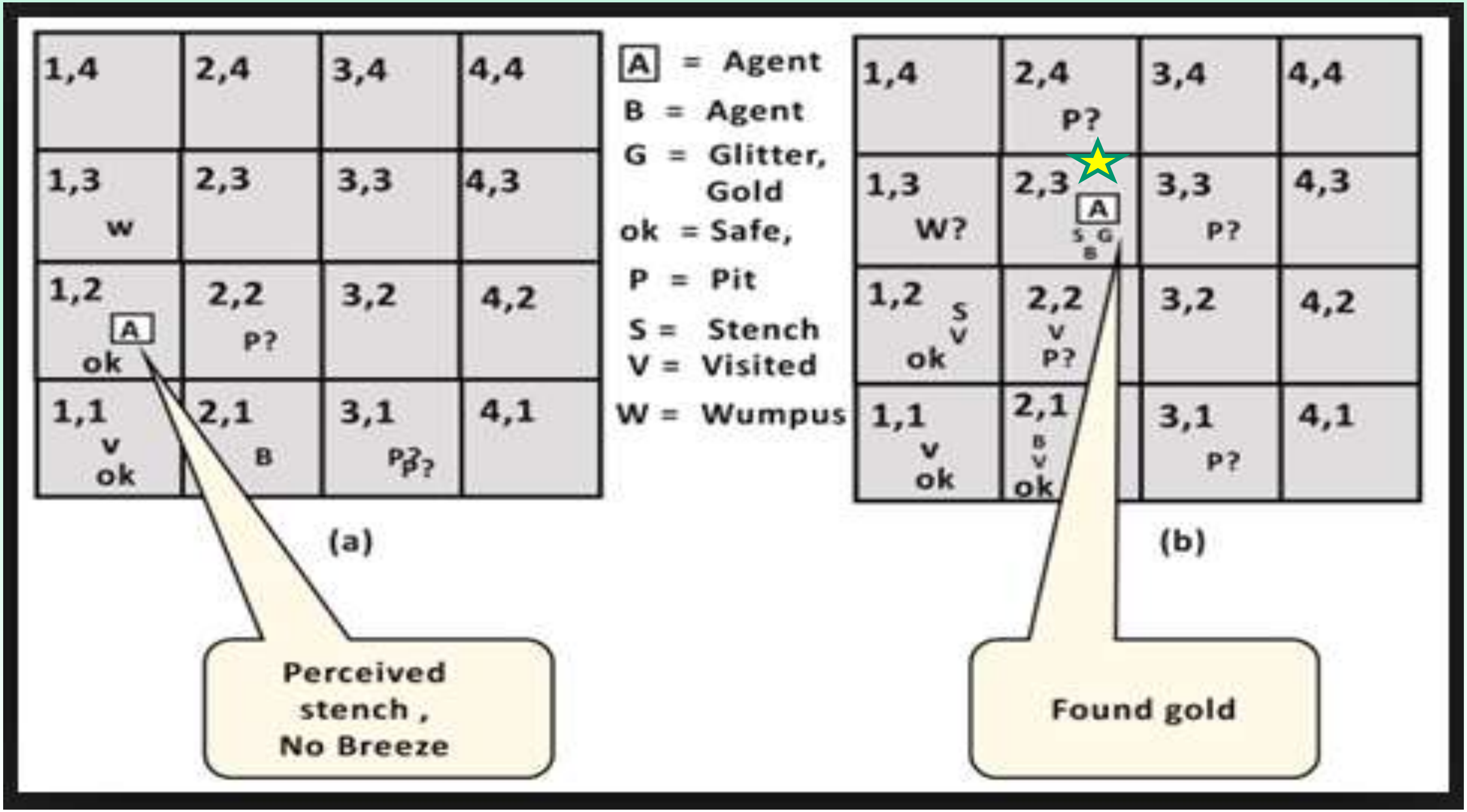
- **Knowledge Representation Language (KRL) used** – writing **symbols** in the **grids**
- **Initial KB** – contains **rules** of the game

- **Start** grid [1,1] & it is **safe** – denoted by **A** (agent) & **OK**
- 1st percept is **[None, None, None, None]** => neibouring grids **[1,2]** & **[2,1]** are safe (**OK**)
- If **Agent** moves to **[2,1]** => Perceives **breeze** (**B**)=>**Pit(s)** present in **[2,2]** or **[3,1]** or **both (P?)**
- Only safe square is **[1,2].** Hence agent should move back to **[1,1]** & then to **[1,2]**



(a) (b)

- In [1,2] perceived **Stench** & **No breeze** – denoted by **S - [Stench, None, None, None, None]**
- After 5th move perceived **[Stench, Breeze, Glitter, None, None]** => **Found Gold**



Logic & Deduction

- A **formal system** for describing **states of affairs**, consisting of:
  - The **syntax** of the language describing *how to make sentences*
  - The **semantics** of the language describing *the relation between the sentences & the states of affairs*
  - A **proof theory** – *a set of rules for logically deducing entailments* of *a set of sentences*
- Improper definition of logic or incorrect proof theory can result in absurd reasoning

Types of Logics

Language	What exists	Belief of agent
Propositional Logic	Facts	True/False/Unknown
First-Order Logic	Facts, Objects, Relations	True/False/Unknown
Temporal Logic	Facts, Objects, Relations, Times	True/False/Unknown
Probability Theory	Facts	Degree of belief 0..1
Fuzzy Logic	Degree of truth	Degree of belief 0..1

1. Propositional Logic

- **Simple** but **Powerful**
- Contains a set of **atomic propositions** $\mathcal{AP}$
- It contains **Syntax, Semantics & Entailment -**
  - **Syntax** - Defines allowable sentences
  - **Sentences** – 2 types
  - **Atomic sentence** – Single symbol that can be **True** | **False** | $\mathcal{AP}$

    Ex- *P, Q, R, $W_{1,3}$ (means Wumpus in [1,3]), North…*
  - **Complex sentence** - (Sentence) | [Sentence]
    - | : Logical Connective like
    - ($\neg$ *(negation),* $\wedge$ *(and) ,* $\vee$*(or),* $\Leftrightarrow$ *(if & only if),* $\Rightarrow$*(implies))*

      Ex: $W1,3 \Leftrightarrow \neg W2,2$
  - **Semantics –** *Defines the <u>rules for determining the truth</u> of the statement in a given model*
  - **Entailment** – *<u>Relation</u> between 2 successive sentences*

Inference Rules

- **Inference rule** - transformation rule - is a logical form that takes premises, analyzes their syntax, and returns a conclusion

1. **Modus Ponens** or Implication Elimination:

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

- **Premise-1** : "If α then β " $\alpha \Rightarrow \beta$,
- **Premise-2** : α ,
- **Conclusion** : β

$\alpha \Rightarrow \beta$, Given α
Conclusion β

=> if the premises are true, then so is the conclusion.

Inference Rules

2. Unit Resolution:

$$\frac{\alpha \vee \beta, \quad \neg\beta}{a}$$

- If $\alpha \vee \beta$ is True & $\neg\beta$ is True, Then α is True

3. Resolution:

$$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{a \vee \gamma} \quad \text{or} \quad \frac{\neg\alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg a \Rightarrow \gamma}$$

- The 2 premises are said to be resolved and the variable β is said to be resolved away.

…. and several other rules

- A **sentence/premise** may have:
  - **Validity** (always true)
  - **Satisfiability** (sometimes true)
  - **No Satisfiability** (always false)

Propositional Logic

- **Semantics** *(Defines the rules for determining the truth of the statement)*
- **Atomic sentence –** 2 rules (True & False)
- **Complex sentence –** 5 rules
  - *$\neg P$ is true iff P is false*
  - *$P \wedge Q$ is true iff both P & Q are true*
  - *$P \vee Q$ is true iff either P or Q is true*
  - *$P \Rightarrow Q$ is true unless P is true & Q is false*
  - *$P \Leftrightarrow Q$ is true iff P & Q are both true or both false*
- **Truth Tables -** Specify truth value of complex sentence for each possible value

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

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

A square is breezy if the neighboring squares have pit and vice versa

- Find if the following is valid, satisfactory or invalid?

$$((P \land Q) \Rightarrow R) \lor (\neg Q \Rightarrow \neg R)$$

KNOWLEDGE REPRESENTATION

Q.1 Prove that $((P \wedge Q) \rightarrow R) \vee (\neg Q \rightarrow \neg R)$ is valid

P	Q	R	$P \wedge Q$	$\neg(P \wedge Q)$		Result
						T
F	F	F	F	T		T
F	F	T	F	T		T
F	T	F	F	T		T
F	T	T	F	T		T
T	F	F	F	T		T
T	F	T	F	T		T
T	T	F	T	F		T
T	T	T	T	F		

Slide 24

Q: (a) "Steve" likes "easy" questions

(b) "Science" is a "hard" course

(c) All courses in "Basket weaving" (BW) Dept are "easy"

(d) "BK301" is a "Basket Weaving" Course.

(a) $\forall x$ course$(x, D) \wedge$ easy$(x) \Rightarrow$ Likes(steve, x)

(b) course(Science, D) $\Rightarrow$ hard(Science)

(c) $\forall x$ Course$(x$, BWD$) \Rightarrow$ easy(x)

(d) Course(BK301, BWD)

(Q) "steve" likes "BK301" $\Rightarrow$ $\neg$ likes(steve, BK301)

---

(1) Remove $\Rightarrow$ from (a)

(a) $\neg$ course(x, D) $\vee$ $\neg$easy(x) $\vee$ Likes(steve, x)

(d) course(BK301, BWD)

x = BKD301

$\neg$ easy(BKD301) $\vee$ Likes(steve, BKD301)

(c) $\neg$ Course(BK301, BWD) $\vee$ easy(BK301)

Likes(Steve, BKD301) $\vee$ $\neg$Course(BK301, BWD)

(d) Course(BK301, BWD)

Likes(Steve, BK301)

Propositional Logic – Example – Wumpus world

- A Simple **Knowledge Base**
  - Example KB for Wumpus world
    - ***Px,y*** *– True, if* ***pit*** *is there in [x,y]*
    - ***Wx,y*** *- True, if* ***wumpus*** *is there in [x,y]*
    - ***Bx,y*** *- True, if* ***breeze*** *is there in [x,y]*
    - ***Sx,y*** *- True, if* ***stench*** *is there in [x,y]*
  - Sentences (Enumerate)
    - $R_1 : \neg P_{1,1}$ *// There is* ***no pit*** *in [1,1]*
    - $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{1,2})$ // A square is breezy if the neighboring squares have pit & vice versa

Propositional Logic

- A Simple **Inference Procedure**
  - Models are assignments of True or False to every symbol
  - Check the sentences are true in every model
  - Example Inference Procedure (Wumpus world)
    - *Seven symbols – $B_{1,1}, B_{2,2}, P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}, P_{3,1}$*
    - *2^7 = 128 possible models*
    - *In three cases, KB is true*
    - Time Complexity = $O(2^n)$ n = no. of symbols in KB
    - Space Complexity = $O(n)$

Examples - Automated Reasoning

Example-1: *Deducing the position of the wumpus based on information like Stench, Breeze etc..*

Example-2:

- If the **unicorn** is **mythical**, then it is **immortal**, (premises)
- But, if it is **not mythical**, then it is a **mortal mammal**.
- If the **unicorn** is either **immortal** (P_1) or a **mammal** (P_2), then it is **horned** (**Q**) .
- The **unicorn** is **magical** if it is **horned**

Q: Can we prove that the unicorn is **mythical**? **Magical**? **Horned**?

- In general, the inference problem is NP-complete (Cook's Theorem)
- If we restrict ourselves to Horn sentences, then repeated use of Modus Ponens gives us a polytime procedure. Horn sentences are of the form:

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow Q$$

Conjunctive Normal Form (CNF)

- **Conjunctive normal form (CNF)** is an approach to Boolean logic that expresses **formulas** as :

- **Conjunctions of clauses with an AND or OR**

- Each **clause** connected by a **conjunction**, **(AND)** must be either a **literal** or **contain a disjunction** (**OR**) operator.

$$a \wedge b$$

$$(a \vee \neg b) \wedge (c \vee d)$$

$$\neg a \wedge (b \vee \neg c \vee d) \wedge (a \vee \neg d)$$

- **CNF** is useful for automated theorem proving

Conjunctive Normal Form (CNF)

Conversion Procedure to Normal Form

STEP I: Eliminate implication and biconditionals. We use the following laws

$$(P \Rightarrow Q) = \neg P \vee Q$$

$$(P \Leftrightarrow Q) = (P \Rightarrow Q) \wedge (Q \Rightarrow P)$$

$$= (\neg P \vee Q) \wedge (\neg Q \vee P)$$

// Replace all $\Rightarrow$

STEP II: Reduce the NOT symbol by the formula $(\neg(\neg P)) = P$ and apply De Morgan's theorem to bring negations before the atoms.

$$\neg(P \vee Q) = \neg P \wedge \neg Q$$

$$\neg(P \wedge Q) = \neg P \vee \neg Q$$

// De Morgan's theorem

STEP III: Use Distributive laws and other equivalent formula given in table III to obtain the normal form

$$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$$

$$P \vee (Q \wedge R) = (P \vee R) \wedge (P \vee R)$$

// in normal form

Propositional Logic – Example-1

3rd inference rule (resolution)

Resolution Rule:

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

// B is resolved away

example: $C1 \land C2$.

C1: $P \lor Q \lor \neg R$.

C2: $\neg Q \lor W$

$P \lor Q \lor \neg R$ $\quad$ $\neg Q \lor W$

// Q is resolved away

$P \lor \neg R \lor W$

(Resolution tree)

Propositional Logic – Example-2

Problem:

- If it is "Hot", Then it is "Humid"
- If it is "Humid", Then it will "Rain"

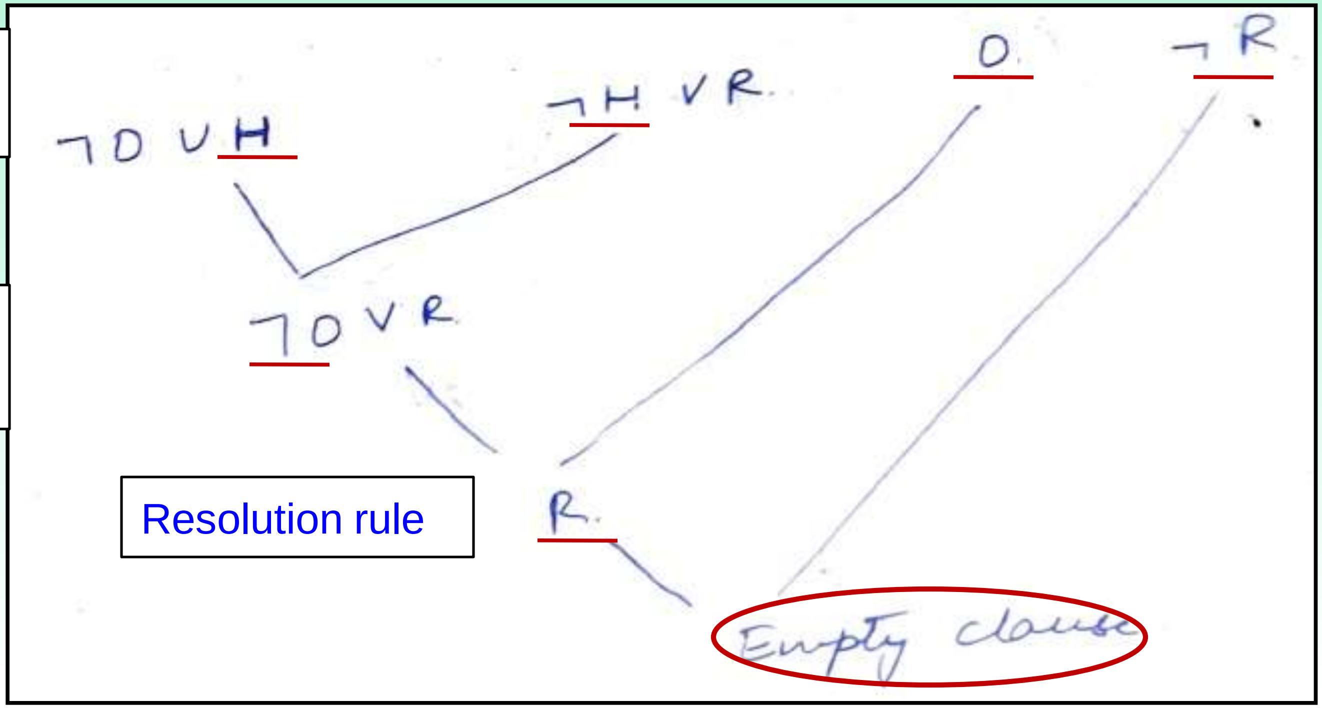
Q: If it is "Hot", Show that it will "Rain"

Solution: H: It is "Humid" (sentences)
R: it will "Rain"
O: It is "Hot"

- If it is "Hot", Then it is "Humid": **O => H**
- If it is "Humid", Then it will "Rain : **H => R**
- It is "Hot" : **O**
- Add "**Negation of Goal**": $\neg R$

CNF: **Step-1**
(eliminate =>)

Apply Resolution Inference rule on **H , O & R**



★ Propositional Logic – Example-3

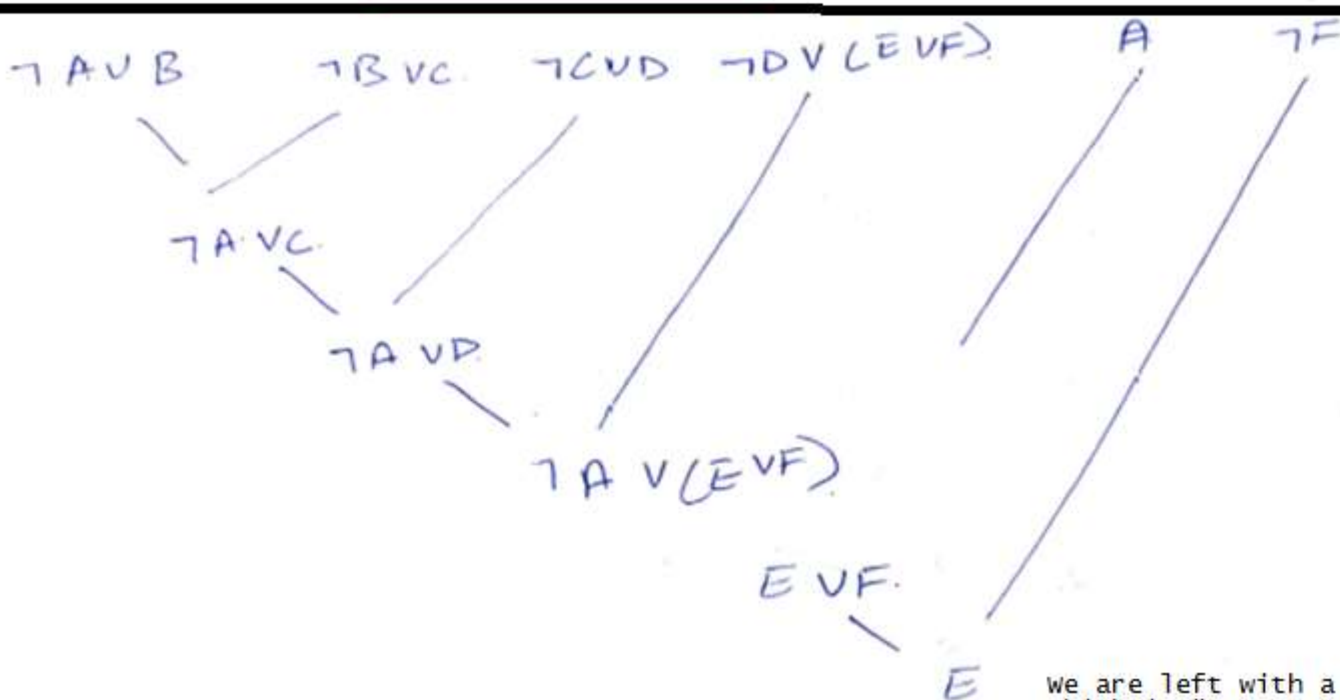
Example 3

Consider the following statements

$A \to B$, $B \to C$, $C \to D$, $D \to E \vee F$

Conclude $A \to F$

Add "**Negation of Goal**": $\neg(A \to F) = A \wedge \neg F$



We are left with a single clause 'E' which is "Not Empty Clause"

Hence A -> F is not valid

2. First-order Predicate Logic (FOPL)

- **FOPL** is a **symbolized reasoning s**ystem in which each sentence is broken down into **(1) a subject** (a variable) & **(2) a predicate** (a function)

- The **predicate** modifies or defines the properties of the **subject**

- A **predicate** can only refer to a single subject

- A **sentence** in **FOPL** is written in the form
  - **Px** or **P(x)**, where **P** is the predicate & **x** is the subject (a variable)

- Complete sentences are logically combined & manipulated as done in Boolean algebra

First-order Predicate Logic (FOPL)

- **Sentence** $\rightarrow$ Atomic Sentence *(P(x) or x = y)*
  | Sentence *Connective* Sentence $(\Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow)$
  | Quantifier Variable $(\forall \mid \exists)$, … Sentence
  | $\neg$ Sentence

- **Atomic Sentence** $\rightarrow$ Predicate(Term, …)| Term = Term
- Term $\rightarrow$ Function(Term, …) | Constant | Variable
- Connective $\rightarrow \Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow$
- Quantifier $\rightarrow$ $\forall \mid \exists$
- Constant $\rightarrow$ A | 5 | Kolkata | …
- Variable $\rightarrow$ a | x | s | …
- Predicate $\rightarrow$ Before | HasColor | …
- Function $\rightarrow$ Is-Prof () | Is_Person () | Is_Dean ()| …

First-order Predicate Logic (FOPL)

- Consider a **subject** as a variable represented by **x**
  - Let **A** be a predicate "is an apple"
  - **F** be a predicate "is a fruit"
  - **S** be a predicate "is sour“
  - **M** be a predicate "is mushy"

$\forall x$	=>	For All x
$\exists x$	=>	Some x

- Then we can say -

$\forall x : Ax \Longrightarrow Fx$

which translates to "For all x, if x is an apple, then x is a fruit." We can also say such things as

$\exists x : Fx \Longrightarrow Ax$

$\exists x : Ax \Longrightarrow Sx$

$\exists x : Ax \Longrightarrow Mx$

where the existential quantifier translates as "For some."

Example-1

$\forall x : Ax \Longrightarrow Fx$ which translates to "For all x, if x is an apple, then x is a fruit." We can also say such things as

$\exists x : Fx \Longrightarrow Ax$ where the existential quantifier translates as "For some."

$\exists x : Ax \Longrightarrow Sx$

$\exists x : Ax \Longrightarrow Mx$

1st – If x is a apple => All x are fruits,

2nd – If x is a fruit => **some** x are apple

3rd – Some apples are sour

4th – Some apples are mushy

$\forall x$	=>	For All	x
$\exists x$	=>	Some	x

Examples-2

Formal definition – using FOPL

1. Lucy* is a professor   is-prof(lucy)
2. All professors are people. $\forall x\,(\text{is-prof}(x) \rightarrow \text{is-person}(x))$
3. John is the dean.   is-dean(John)
4. Deans are professors. $\forall x\,(\text{is-dean}(x) \rightarrow \text{is-prof}(x))$
5. All professors consider the dean a friend or don't know him.
   $\forall x\,(\forall y\,(\text{is-prof}(x) \wedge \text{is-dean}(y) \rightarrow \text{is-friend-of}(y,x) \vee \neg \text{knows}(x, y)))$
6. Everyone is a friend of someone. $\forall x\,(\exists y\,(\text{is-friend-of}(y, x)))$
7. People only criticize people that are not their friends.
   $\forall x\,(\forall y\,(\text{is-person}(x) \wedge \text{is-person}(y) \wedge \text{criticize}(x,y) \rightarrow \neg \text{is-friend-of}(y,x)))$
8. Lucy criticized John .   criticize(lucy, John)

Question: Is John no friend of Lucy?

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

FOPL – Example-3

Problem:

- Show the validity of the following sentence
- All men are mortal. John is a man. Therefore John is Mortal.

Man(x) – x is a man

Mortal(x) – x is mortal

$(\forall x)(Man(x) \rightarrow Mortal(x))$ // For all 'x', if 'x' is man => 'x' is mortal

$\neg(\forall x)\, Man \lor Mortal(x)$ // Replacing =>

$((\forall x)\neg Man(x) \lor Mortal) \land Man(John)$ // All men are mortal. John is a man

$(\neg \cancel{Man}(X) \lor \cancel{Mortal}(X)) \land \cancel{Man}(\cancel{John}) \land \neg \cancel{Mortal}(John)$

// Negation of Goal

if $X = John$ Empty Clause

Hence true

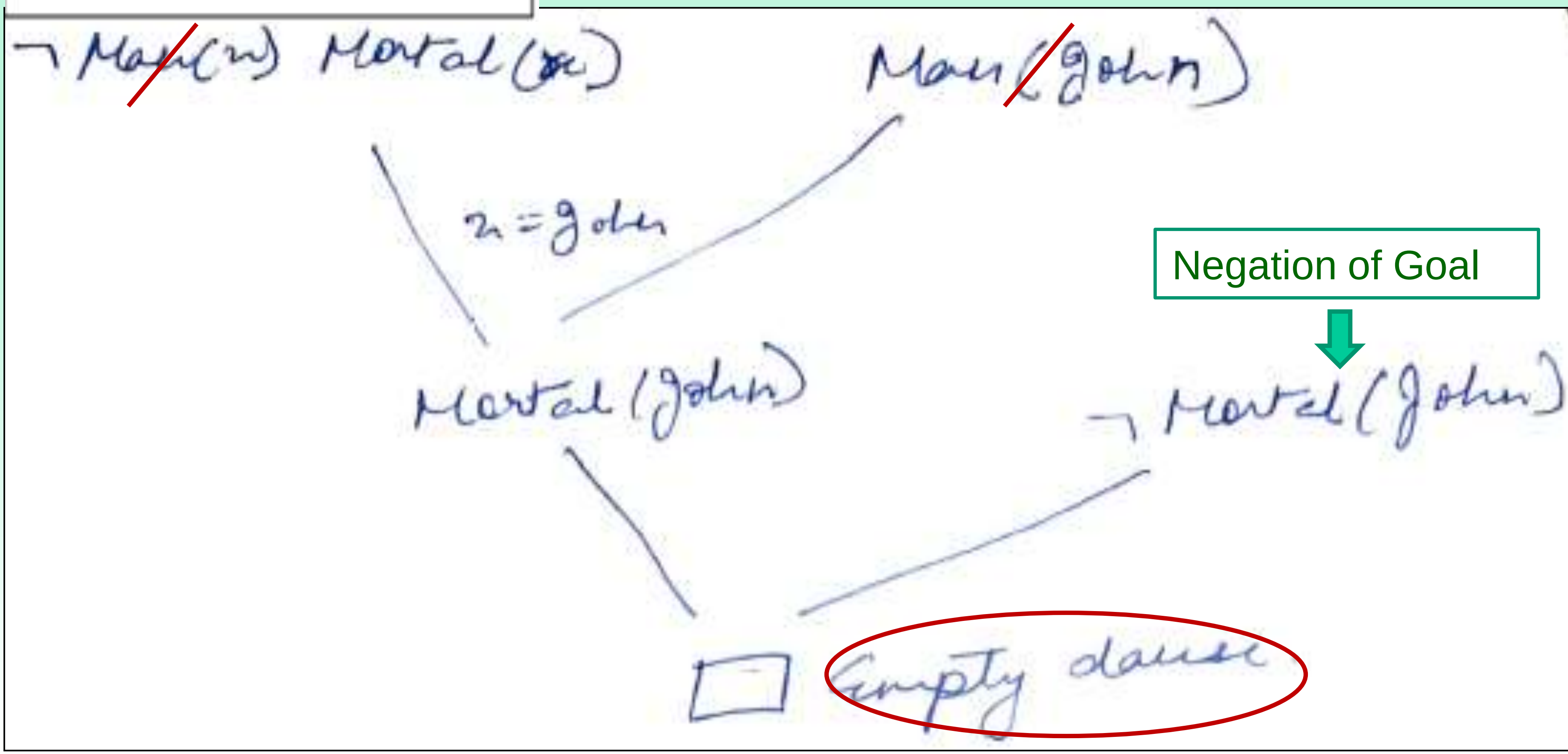
FOPL – Example-3

Problem:

- Show the validity of the following sentence
- All men are mortal. John is a man. Therefore John is Mortal.

$(\forall x)(Man(x) \rightarrow Mortal(x));$

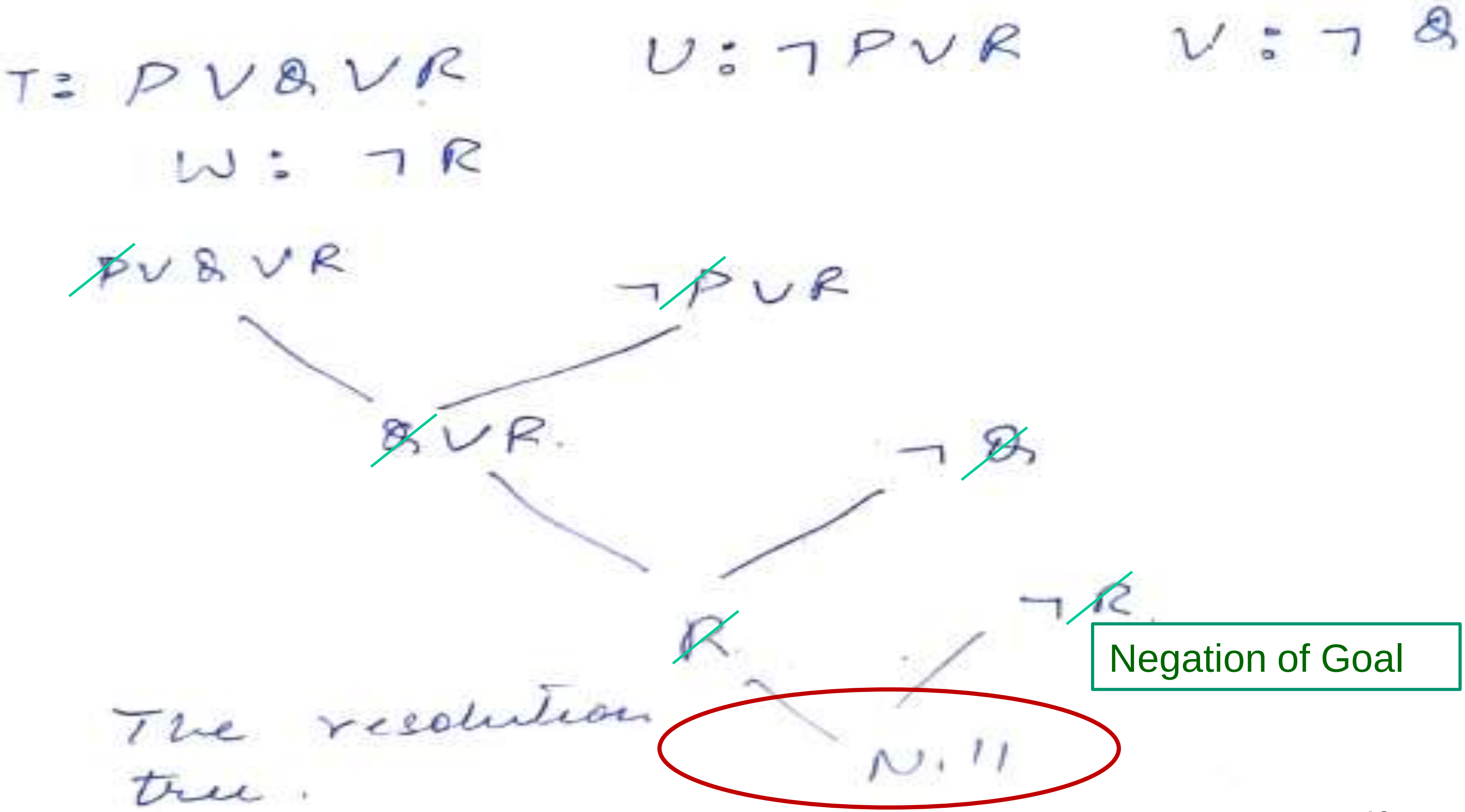
Replace =>



FOPL – Example-4

Problem:

- Given the following predicate show how resolution process can be applied



FOPL - Example – Knowledge Base

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

Sentences in FOPL

- **It is a crime for an American to sell weapons to hostile nations:**

  $American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$

  < *x – person, y – weapon, z – country* >

- **Nono…has some missiles**

  $\exists x\ Owns(Nono, x) \wedge Missiles(x)$ *// Some weapons owned by Nono are missiles*

  $Owns(Nono, M_1)$ *and* $Missile(M_1)$

  < *x – weapon, y – missile* >

- **All of its missiles were sold to it by Col. West**

  $\forall x\ Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

  < *x – missile* >

- **Missiles are weapons**

  $Missile(x) \Rightarrow Weapon(x)$

- **An enemy of America counts as “hostile”**

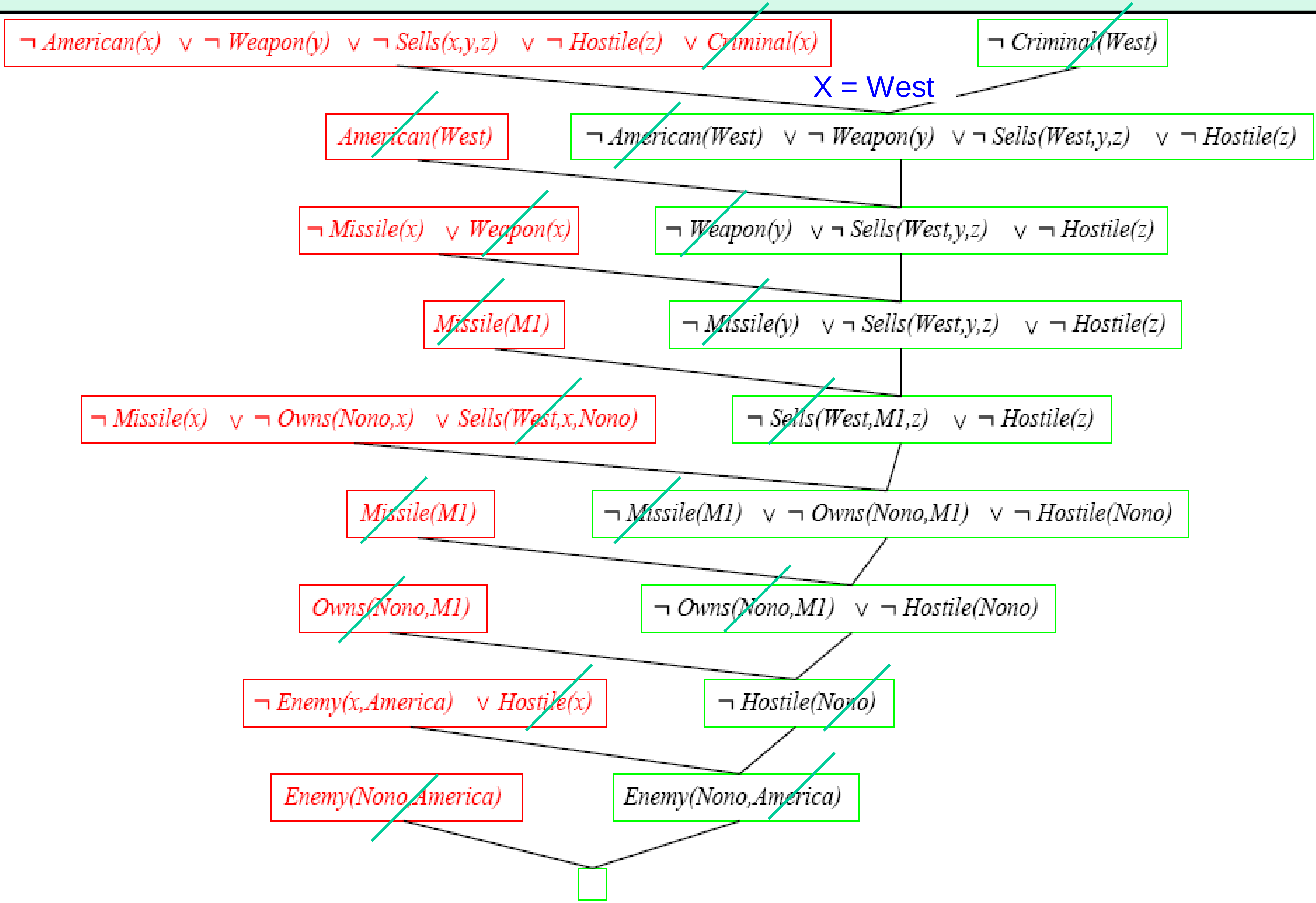
  $Enemy(x, America) \Rightarrow Hostile(x)$

- **Col. West who is an American**

  American(Col. West)

- **The country Nono, an enemy of America**

  Enemy(Nono, America)

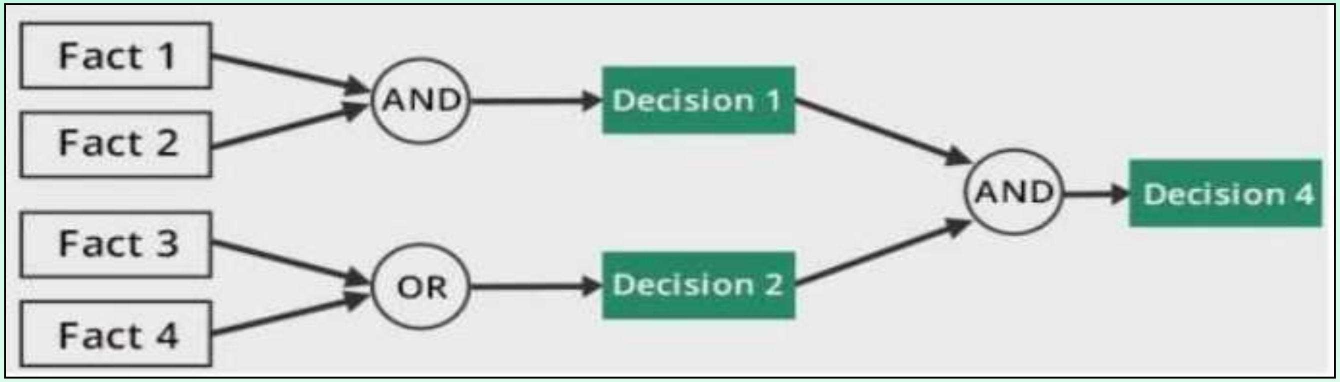

¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)
¬ Criminal(West)
X = West
American(West)
¬ American(West) ∨ ¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ Weapon(x)
¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(y) ∨ ¬ Sells(West,y,z) ∨ ¬ 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)

Knowledge-Based Agents - Topics

- **Introduction**
- **Knowledge-Based Agents**
- **WUMPUS WORLD Environment**
- **Propositional Logic**
- **First Order Predicate Logic**
- **Forward and Backward Chaining**

Forward Chaining

- **Forward chaining** is a **data driven** method of
  - Deriving a particular goal
    - from a given knowledge base & a set of inference rules
- The application of inference rules results in new knowledge
  - which is then added to the knowledge base
- Used to answer the question "**What can happen next**"
- The inference engine applies *chain of conditions, facts & rules*
  - to arrive at a *solution* (**decision** or **goal**)



Forward Chaining

- The system **starts** from a set of **facts** & a set of **rules**
  - Tries to find ways of using them to **deduce** a **conclusion** (goal)

- Called **data-driven reasoning** because the reasoning starts from a set of data and ends up at the goal

- **1st Step** – Take the facts from the *fact database* & see if any combination of these matches any of the components of rules in the *rule database*

- **2nd Step** – In case of a match, the rule is **triggered (fired)**

- **3rd Step** – Then it's **conclusion** is added to the *facts database*

- **4th Step** - If the **conclusion** is an **action**, then the system causes that action to take place

Forward Chaining – Example - Elevator

Rule 1

IF on first floor and button is pressed on first floor

THEN open door

Rule 2

IF	on first floor
AND	button is pressed on second floor
THEN	go to second floor

Rule 3

IF	on first floor	// Fact-1
AND	button is pressed on third floor	// Fact-2
THEN	go to third floor	// Conclusion added to KB

Forward Chaining - Example - Elevator

Rule 4

IF on second floor
AND button is pressed on first floor
AND already going to third floor
THEN remember to go to first floor later

// Fact-4 added to KB

Let us imagine that we start with the following facts in our database:

Fact 1

At first floor

Fact 2

Button pressed on third floor

Fact 3

Today is Tuesday

Forward Chaining - Example - Elevator

- The system examines the rules & finds that Facts 1 & 2 match the components of Rule 3

- Rule 3 fired & its conclusion "Go to 3rd floor" is added to the facts database

- This results in the elevator heading to the 3rd floor

- Note that Fact 3 (*today is Tuesday*) was ignored because it did not match the components of any rules

- Assuming the elevator is going to the 3rd floor & has reached the 2nd floor, when the button is pressed on the 1st floor

- The fact "Button pressed on first floor" Is now added to the database, which results in Rule 4 firing (*remember to go to first floor*)

Forward Chaining - Example – Knowledge Base

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

Sentences in FOPL

- **It is a crime for an American to sell weapons to hostile nations:**

  $American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$

  < x – person, y – weapon, z – country >

- **Nono…has some missiles**

  $\exists x\ Owns(Nono, x) \wedge Missiles(x)$ *// Some weapons owned by Nono are missiles*

  $Owns(Nono, M_1)$ and $Missile(M_1)$

  < x – weapon, y – missile >

- **All of its missiles were sold to it by Col. West**

  $\forall x\ Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

  < x – missile >

- **Missiles are weapons**

  $Missile(x) \Rightarrow Weapon(x)$

- **An enemy of America counts as "hostile"**

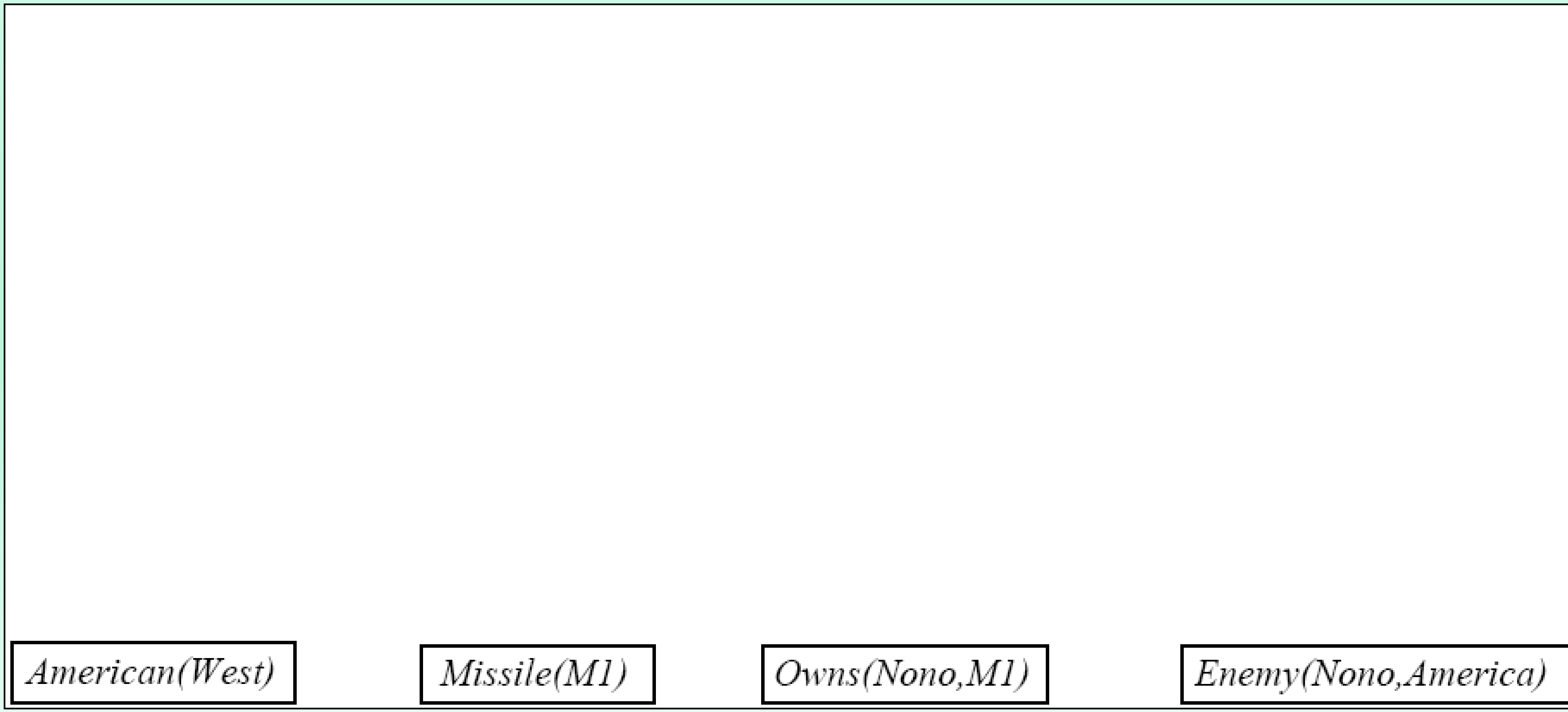
  $Enemy(x, America) \Rightarrow Hostile(x)$

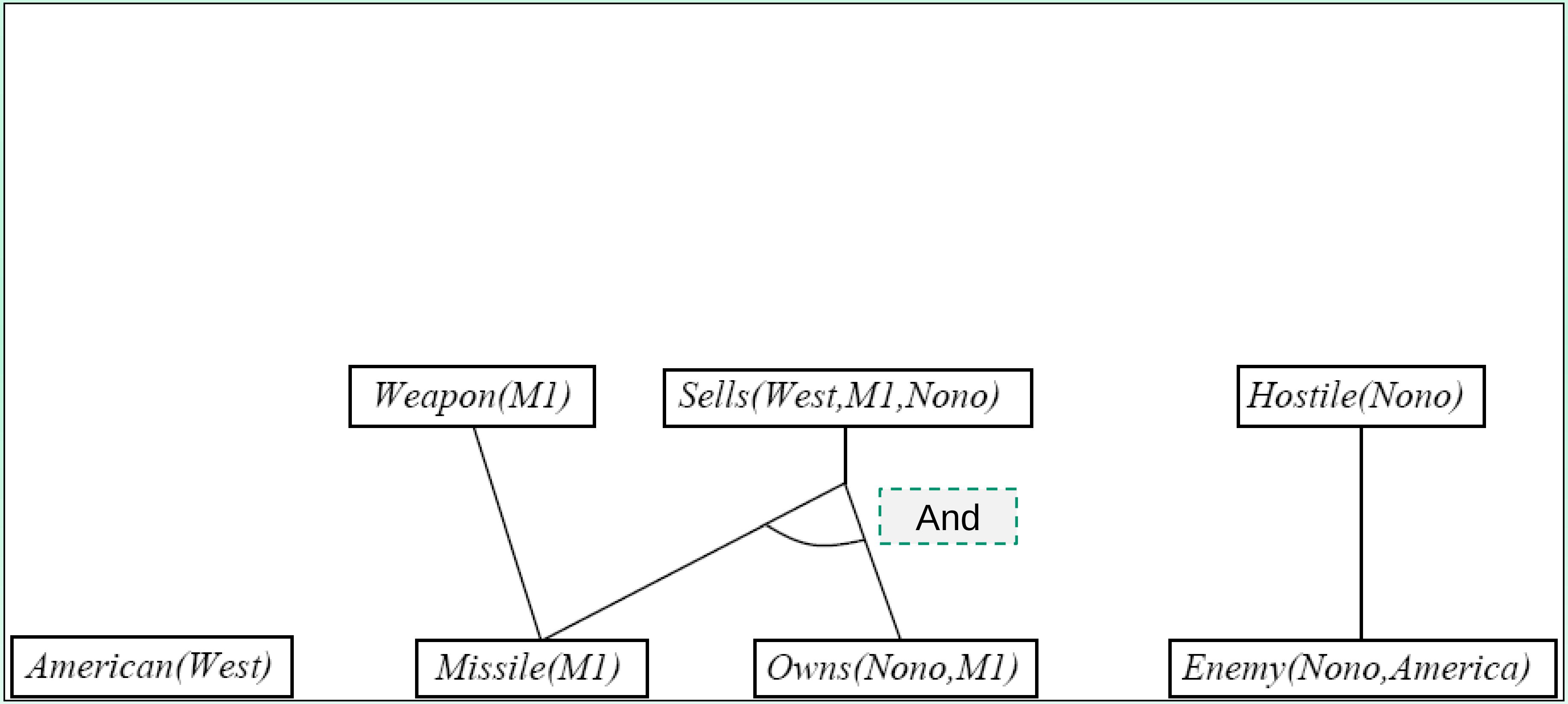
- **Col. West who is an American**

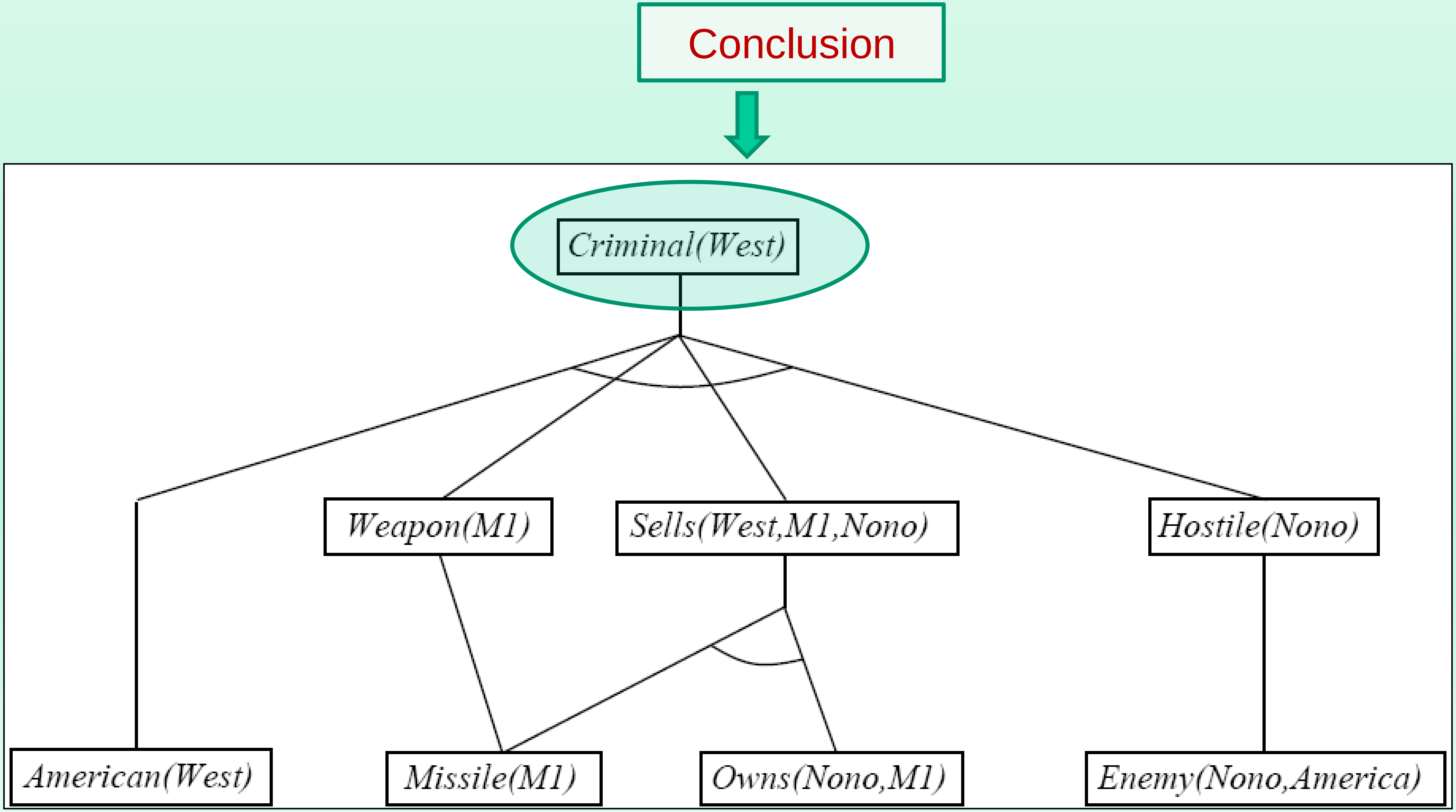
  $American(Col.\ West)$

- **The country Nono, an enemy of America**

  $Enemy(Nono, America)$

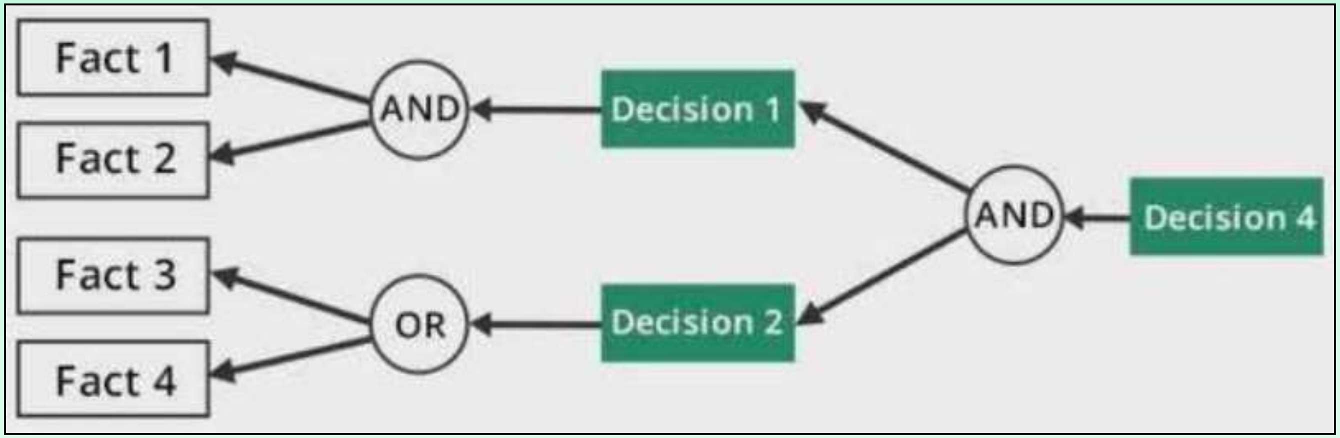

American(West)
Missile(M1)
Owns(Nono,M1)
Enemy(Nono,America)


Weapon(M1)
Sells(West,M1,Nono)
Hostile(Nono)
And
American(West)
Missile(M1)
Owns(Nono,M1)
Enemy(Nono,America)


Conclusion
Criminal(West)
Weapon(M1)
Sells(West,M1,Nono)
Hostile(Nono)
American(West)
Missile(M1)
Owns(Nono,M1)
Enemy(Nono,America)

Backward Chaining

- **Backward chaining** is a **goal driven method** of
  - Deriving a particular goal from a given knowledge base & a set of inference rules
- Inference rules are applied by matching the **goal** to the **results of the relations** stored in the knowledge base
- Used to answer the question "**Why this happened**"
- Based on what has already happened, the inference engine tries to find out which conditions (***causes*** *or* ***reasons***) could have happened for this result



Backward Chaining

- The system starts from a conclusion (hypothesis to prove or goal)
  - Tries to show how the conclusion has been reached from the rules & facts in the database
- Reasoning in this way is called as **goal-driven reasoning**
- **Steps** – Start with the goal state & see what actions could lead to it of the components of rules in the *rule database*
- **Ex:**
  - If the goal state is "blocks arranged on a table"
  - Then one possible action is to "place a block on the table"
  - This action may not be possible from the start state
  - Further actions need to be added before this action
  - In this way, a plan is formulated starting from the goal & working back toward the start state

Backward Chaining

- Backward chaining ensures that each action that is taken is one that will definitely lead to the goal

- In many cases, Backward Chaining will make the planning process far more efficient compared to Forward Chaining

Example - WEATHER FORECAST SYSTEM

Rule I

If we suspect temperature is less than 20° // Premise-1 – R1 (conclusion of rule-2)

AND there is humidity in the air

Then there are chances of rain 3 // Rule-1

Rule II

If Sun is behind the clouds 1 // Premise-1 – R2 (This is ‘Known’)

AND air is very cool. // Premise-2 – R2 (This is ‘Known’)

Then we suspect temperature is less than 20°. // Fire R-2

Rule III **If** air is very heavy // Premise-1 – R3 (This is ‘Known’)

Then there is humidity in the air. 2 // Conclusion of R-3

- Suppose we have been given the following facts,
  - a) Sun is behind the clouds
  - b) Air is very heavy & cool
- **Problem**: Use **Backward chaining** to conclude there are **chances of rain**

Example - WEATHER FORECAST SYSTEM

Step	Description	Working Memory
1	Goal "There are chances of rain." Not in Working Memory.	
2	Find rules with our goal "There are chances of rain" in conclusion: It is in Rule 1.	
3	Now see if Rule 1, premise 1 is known "we suspect temperature is less than 20^0".	
4	This is conclusion of rule 2. So going to Rule 2. The premise 1 of rule 2 is "Sun is behind the clouds".	
5	This is primitive. We ask from user Response: Yes	Sun is behind the clouds.

Example - WEATHER FORECAST SYSTEM

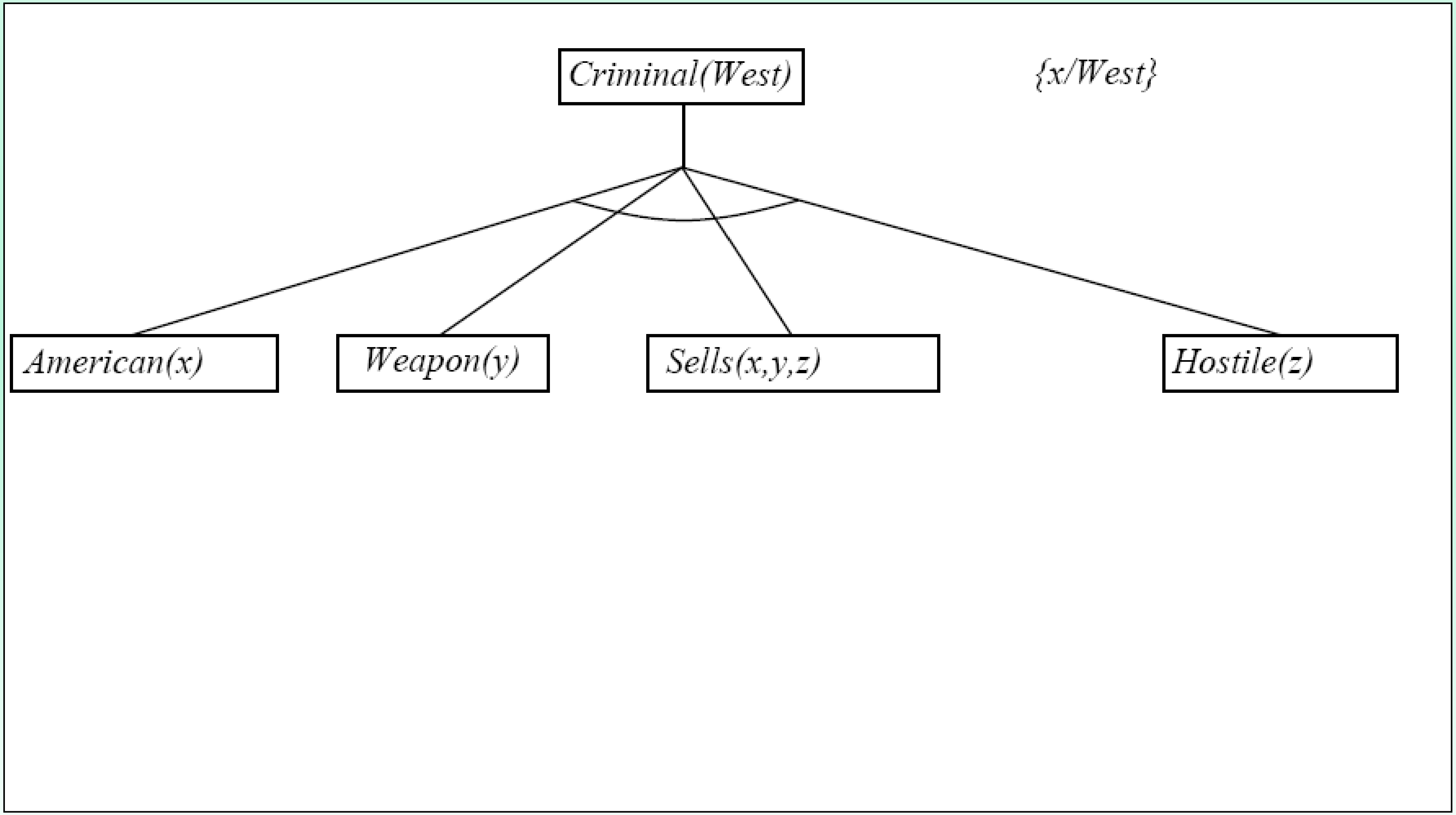
6	See if Rule 2, premise 2 is known "Air is very cool".	
7	This is also primitive. We ask its Response: Yes. Both conditions of Rule 2 are met so Fire rule 2	Sun is behind the clouds. Air is very cool. **We suspect temperature is less than 20^0.**
8	So Rule 1 premise 1 is in working memory, coming to Rule 1, premise 2 "There is humidity in the air"	Sun is behind the clouds. Air is very cool. **We suspect temperature is less than 20^0.**
9	This is conclusion of Rule 3. So see if Rule 3, premise 1 is known "Air is very heavy".	Sun is behind the clouds. Air is very cool. **We suspect temperature is less than 20^0.**

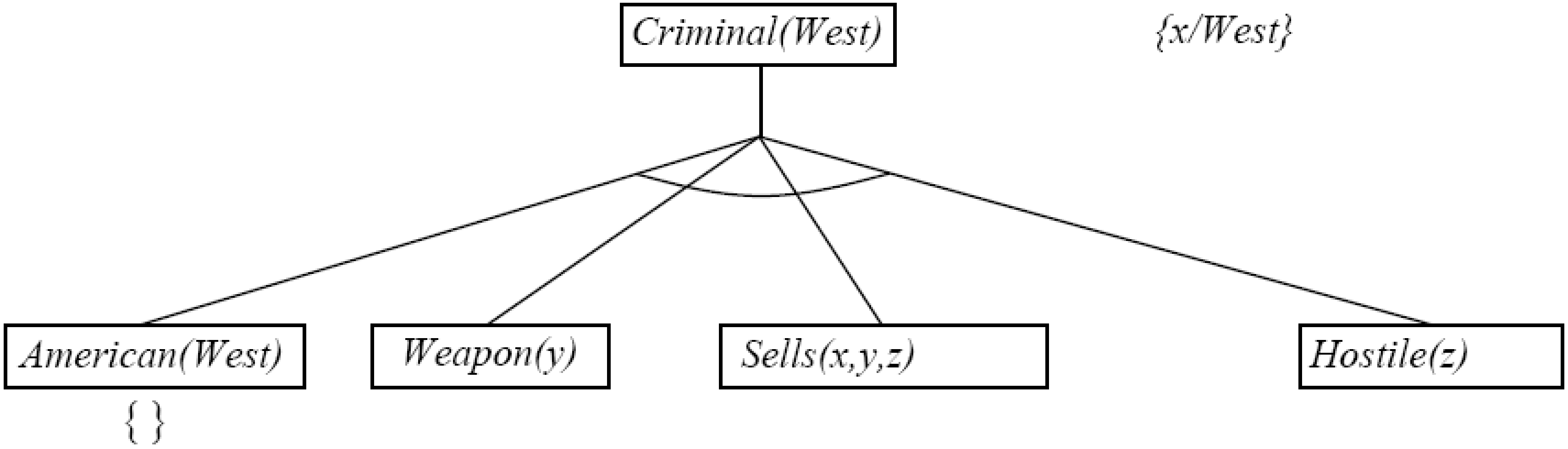
Example - WEATHER FORECAST SYSTEM

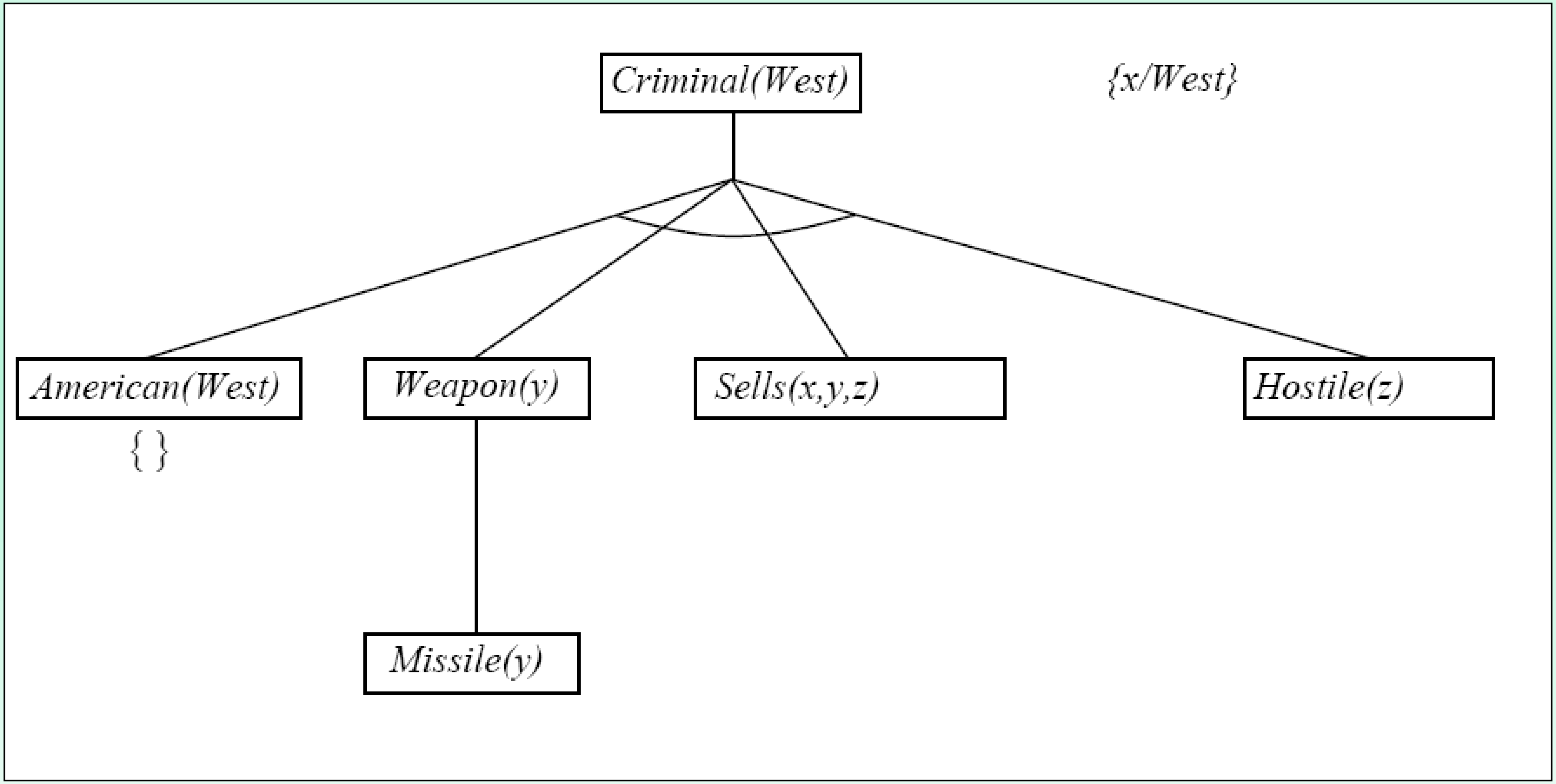
10	This is primitive so asking from user Response: Yes. Fire rule	Sun is behind the clouds. Air is very cool. We suspect temperature is less than 20^0. **There is humidity in the air.**
11	Now Rule 1 premise 1 and 2 both are in working memory so fire Rule 1.	Sun is behind the clouds. Air is very cool. Air is very heavy. We suspect temperature is less than 20^0. There is humidity in the air. **There are chances of rain.**

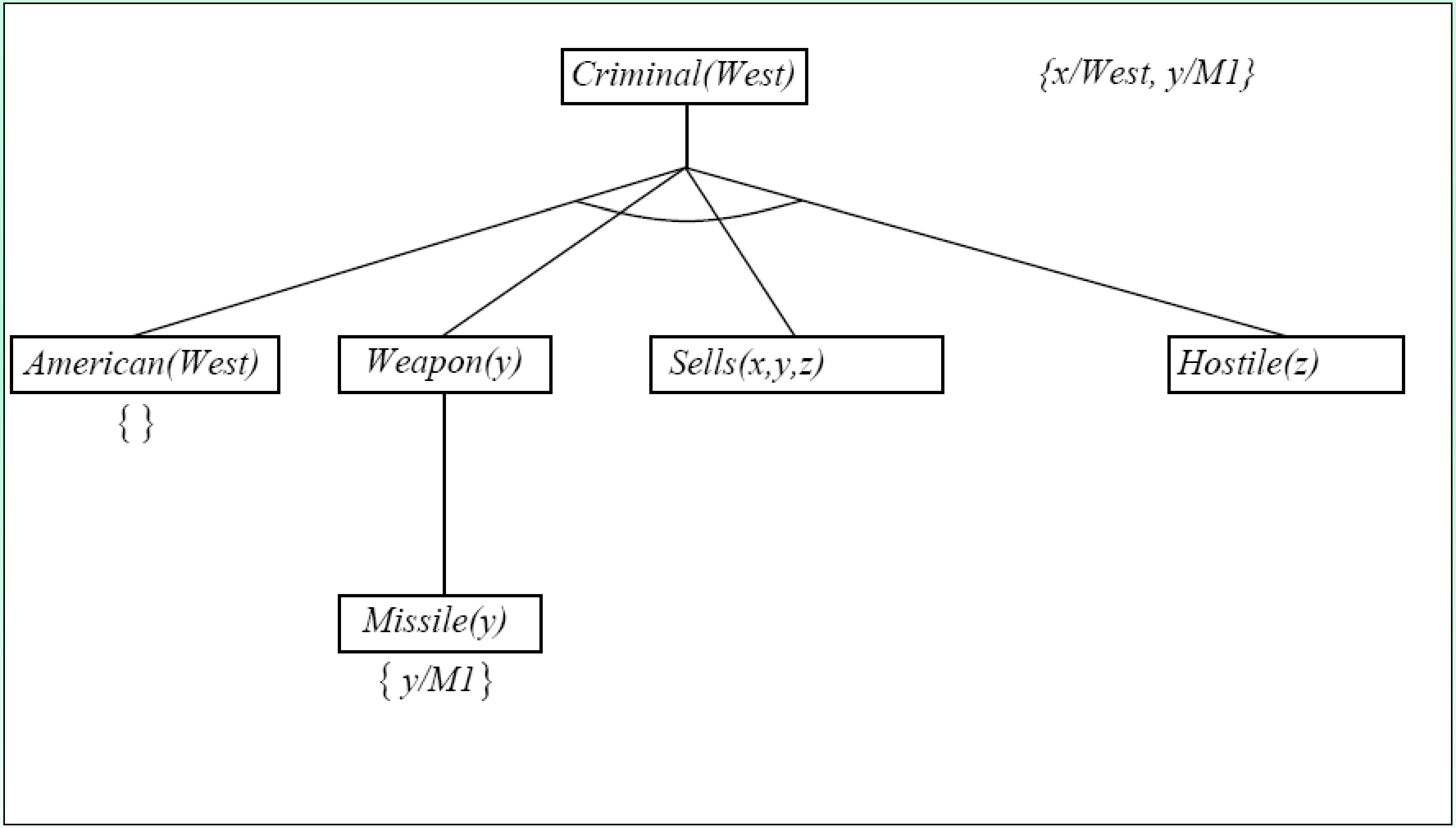
Backward Chaining Example

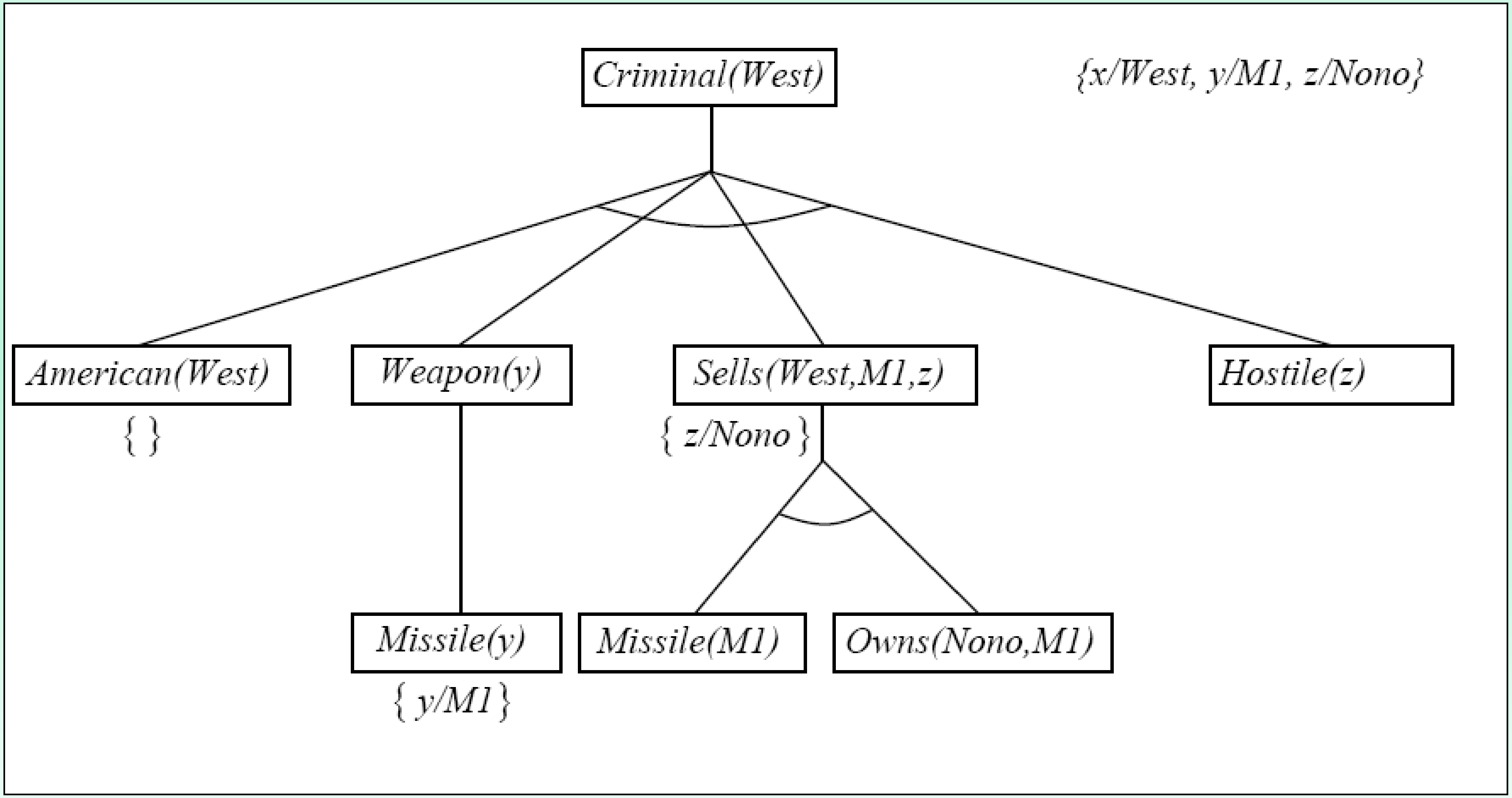
$\boxed{Criminal(West)}$

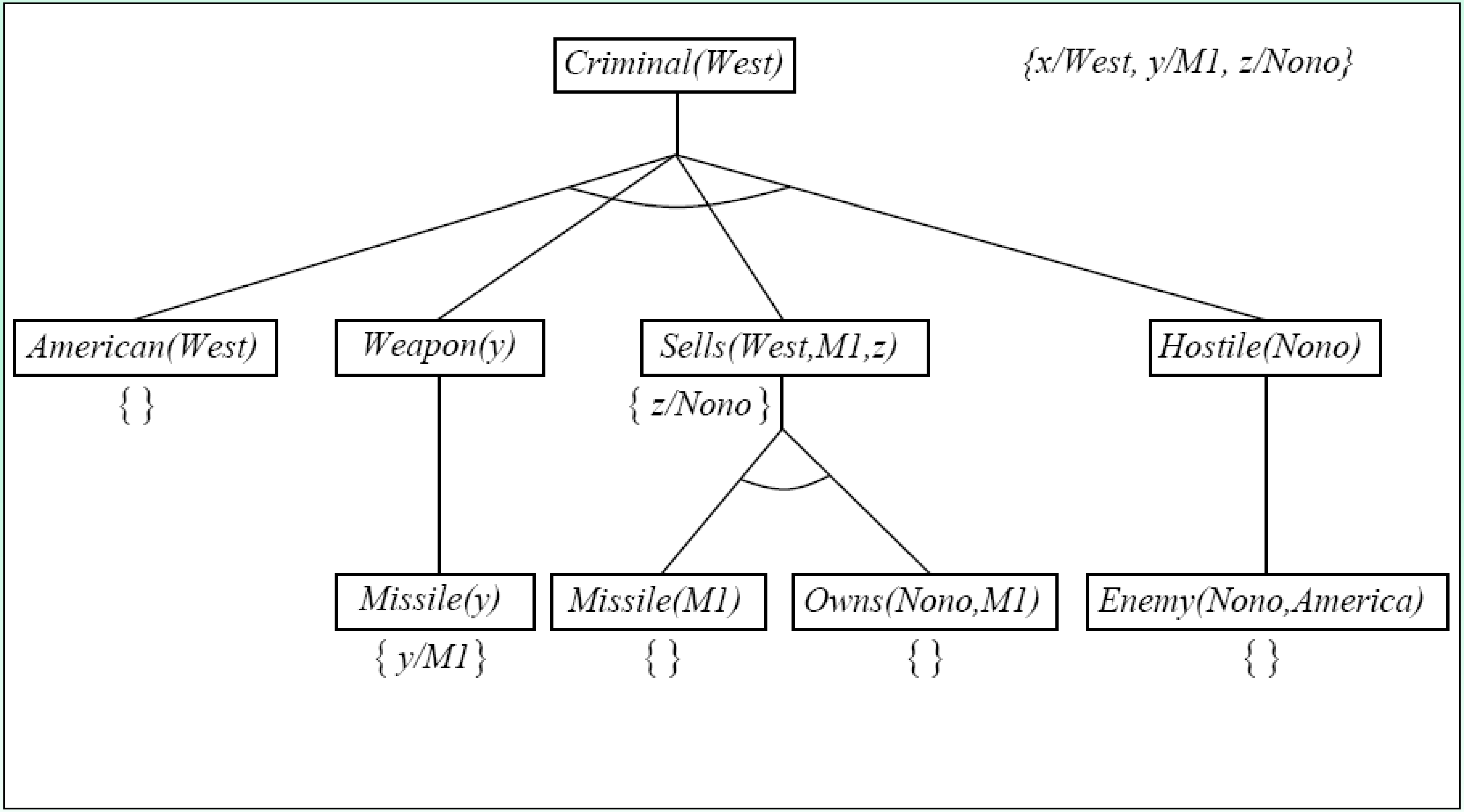

Criminal(West)
{x/West}
American(x)
Weapon(y)
Sells(x,y,z)
Hostile(z)


Criminal(West)
{x/West}
American(West)
{ }
Weapon(y)
Sells(x,y,z)
Hostile(z)


Criminal(West)
{x/West}
American(West)
{ }
Weapon(y)
Sells(x,y,z)
Hostile(z)
Missile(y)


Criminal(West)
{x/West, y/M1}
American(West)
{ }
Weapon(y)
Sells(x,y,z)
Hostile(z)
Missile(y)
{ y/M1}


Criminal(West)
{x/West, y/M1, z/Nono}
American(West)
{ }
Weapon(y)
Sells(West,M1,z)
{ z/Nono }
Hostile(z)
Missile(y)
{ y/M1 }
Missile(M1)
Owns(Nono,M1)


Criminal(West)
{x/West, y/M1, z/Nono}
American(West)
{ }
Weapon(y)
Sells(West,M1,z)
{ z/Nono }
Hostile(Nono)
Missile(y)
{ y/M1 }
Missile(M1)
{ }
Owns(Nono,M1)
{ }
Enemy(Nono,America)
{ }

Backward Chaining Algorithm

function FOL-BC-ASK(KB, goals, θ) **returns** a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query
    θ, the current substitution, initially the empty substitution $\{\ \}$
  local variables: ans, a set of substitutions, initially empty

  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow \text{SUBST}(\theta, \text{FIRST}(goals))$
  for each r **in** KB where STANDARDIZE-APART(r) $= (\ p_1 \ \wedge \ \ldots \ \wedge \ p_n \ \Rightarrow \ q)$
    and $\theta' \leftarrow \text{UNIFY}(q, q')$ succeeds
    $ans \leftarrow \text{FOL-BC-ASK}(KB, [p_1, \ldots, p_n | \text{REST}(goals)], \text{COMPOSE}(\theta', \theta)) \cup ans$
  return ans


The End