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

Example:

a) The Sun rises from West (False proposition)

b) 3+3= 7(False proposition)

c) 5 is a prime number.

Propositional logic in Artificial intelligence

- Propositional logic is also called Boolean logic as it works on 0 and 1.
- In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.
- Propositions can be either true or false, but it cannot be both.
- Propositional logic consists of an object, relations or function, and logical connectives. These connectives are also called logical operators.
- The propositions and connectives are the basic elements of the propositional logic.

- Connectives can be said as 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 Rohini", "How are you", "What is your name", are not propositions.

Syntax of propositional logic:

The syntax of propositional logic defines the allowable sentences for the knowledge representation. There are two types of Propositions:

- Atomic Propositions
- Compound propositions

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

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

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

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

Example:

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

b) "Ankit is a doctor, and his clinic is in Mumbai."

Logical Connectives:

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

	Symbol	Meaning
Negation	~	NOT
Conjunction	∧	AND
Disjunction	∨	OR
Conditional	→	If/Then
Biconditional	⟷	IF and ONLY IF

Logical Connectives:

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

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

Example: Rohan is intelligent and hardworking. It can be written as,

P= Rohan is intelligent,

Q= Rohan is hardworking. → P∧ Q.

3. **Disjunction:** A sentence which has ∨ connective, such as P ∨ Q. is called disjunction, where P and Q are the propositions.

Example: "Ritika is a doctor or Engineer",

Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as P ∨ Q

Logical Connectives:

4. **Implication:** A sentence such as P → Q, is called an implication. Implications are also known as if-then rules. It can be represented as

If it is raining, then the street is wet.

Let P= It is raining, and Q= Street is wet, so it is represented as $P \rightarrow Q$

5. **Biconditional:** A sentence such as P⇔ Q is a Biconditional sentence, example If I am breathing, then I am alive

P= I am breathing, Q= I am alive, it can be represented as $P \Leftrightarrow Q$.

Logical Connectives:

Ram can play tennis (let's take it as variable **X**)

Ram cannot play tennis – There is a negation in the sentence, so symbolic representation will be **˜ X**

Ram can play tennis and badminton – Note, there is a new addition 'Badminton', let's take it as variable **Y**. Now, this sentence has a Conjunction, so symbolic representation will be **X** ∧ **Y**

Logical Connectives:

Ram can play tennis or badminton – Here is a Disjunction, so symbolic representation will be **X** ∨ **Y**

If Ram can play tennis then he can play badminton – There is a condition, so symbolic representation will be **X** → **Y**

Ram can play tennis if and only if he can play badminton – It is a biconditional sentence, so symbolic representation will be **X** ↔ **Y**

Truth Table

P	Q	Negation		Conjunction	Disjunction	Implication	Bi-conditional
		¬ P	¬ Q	P ∧ Q	P ∨ Q	P → Q	P⇔Q
True	True	False	False	True	True	True	True
True	False	False	True	False	True	False	False
False	True	True	False	False	True	True	False
False	False	True	True	False	False	True	True

Properties of Propositional Logic Statements:

A statement is a sentence the two terms have been used interchangeably. When a statement cannot be logically broken into smaller statement. It is called atomic.

Satisfiable:

A atomic propositional formula is satisfiable if there is a interpretation for which it is true. In the table P v Q is satisfiable for first three propositional formula (statements).

Tautology:

A propositional formula is valid or a tautology it is true for all possible interpretations.

Properties of Propositional Logic Statements:

Contradiction:

A propositional formula is considered to be contradictory or unsatisfiable if there no interpretation exists for which it is true. For example, Amritsar is the capital of India

In table P v Q is unsatisfiable for the row 4.

Contingent:

A contingent statement is one which is neither a tautology nor a contradiction.

Precedence of connectives:

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

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

For better understanding use parenthesis to make sure of the correct interpretations. Such as ¬R∨ Q, It can be interpreted as (¬R) ∨ Q.

Truth table with three propositions:

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

P	Q	R	¬R	P∨Q	P∨Q→¬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

Limitations of Propositional logic:

- We cannot represent relations like ALL, some, or none with propositional logic. Example:
  a. **All the girls are intelligent.**
  b. **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.

1. Which is created by using single propositional symbol?

a) Complex sentences

b) Atomic sentences

c) Composition sentences

d) None of the mentioned

2. Which is used to construct the complex sentences?

a) Symbols

b) Connectives

c) Logical connectives

d) All of the mentioned

First-Order Logic in Artificial intelligence

In the topic of Propositional logic, we have seen that how to represent statements using propositional logic. But unfortunately, in propositional logic, we can only represent the facts, which are either true or false. PL is not sufficient to represent the complex sentences or natural language statements. The propositional logic has very limited expressive power. Consider the following sentence, which we cannot represent using PL logic.

"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.

First-Order logic:

•First-order logic is also known as **Predicate logic or First-order predicate logic**. First-order logic is a powerful language that develops information about the objects in a more easy way and can also express the relationship between those objects.

•First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:

- **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, ...

•As a natural language, first-order logic also has two main parts:

- **Syntax**
- **Semantics**

Syntax of First-Order logic:

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

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

- Constants
  - Names of specific objects
  - E.g. doreen, gord, william, 32
- Functions
  - Map objects to objects
  - E.g. father(doreen), age(gord), max(23,44)
- Variables
  - For statements about unidentified objects or general statements
  - E.g. x, y, z, …

- Predicate symbols represent relations between zero or more objects
- The number of objects define a predicate's aritiy
- Examples:
  - Likes(george, kate)
  - Likes(x,x)
  - Likes(joe, kate, susy)
  - Friends (father_of(david), father_of(andrew))

Connectivity

- FOL formulas are joined together by logical operators to form more complex formulas (just like in propositional logic)
- The basic logical operators are the same as in propositional logic as well:
  - Negation: ¬p („it is not the case that p“)
  - Conjunction: p ∧ q („p and q“)
  - Disjunction: p ∨ q („p or q“)
  - Implication: p → q („p implies q“ or “q if p“)
  - Equivalence: p ↔ q („p if and only if q“)

Quantifiers

- Two quantifiers: Universal (∀) and Existential (∃)
- Allow us to express properties of collections of objects instead of enumerating objects by name
  - Apply to sentence containing variable
- Universal ∀: true for **all** substitutions for the variable
  - "for all": ∀<variables> <sentence>
- Existential ∃: true for **at least one** substitution for the variable
  - "there exists": ∃<variables> <sentence>
- Examples:
  - ∃ x: Mother(art) = x
  - ∀ x ∀ y: Mother(x) = Mother(y) → Sibling(x,y)
  - ∃ y ∃ x: Mother(y) = x

Syntax of First-Order logic:

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.

•We can represent atomic sentences as **Predicate (term1, term2, ..., term n)**.

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

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

Complex Sentences:

Complex sentences are made by combining atomic sentences using connectives.

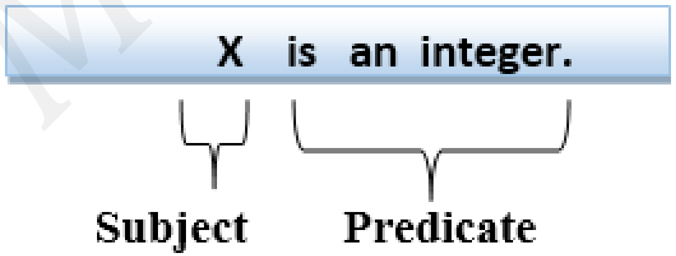
Syntax of First-Order logic:

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:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:

  i. *Universal Quantifier, (for all, everyone, everything)*
  ii. *Existential quantifier, (for some, at least one).*

i. Universal Quantifier, (for all, everyone, everything)

Example:

All man drink coffee.

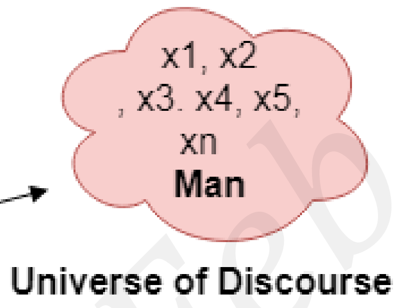
- x1 drinks coffee

  ∧
- x2 drinks

  ∧
- x3 drinks milk

  ∧
- .
- .

  ∧
- xn drinks milk



So in shorthand notation, we can write it as :

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

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

Existential quantifier, (for some, at least one).

It is denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.

In Existential quantifier we always use AND or Conjunction symbol (∧).

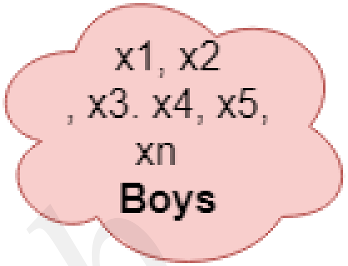
Some boys are intelligent.

- **x1 is intelligent**

  V
- **x2 is intelligentV**
- **x3 is intelligent**

  V
- .
- .

  V
- **xn is intelligent**



Universe of Discourse

So in short-hand notation, we can write it as:

∃x: boys(x) ∧ intelligent(x)

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

Points to remember:

- The main connective for universal quantifier ∀ is implication →.
- The main connective for existential quantifier ∃ is and ∧.

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 $\forall y \exists x$.

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) $\rightarrow$ play(x, cricket).

4. *Not all students like both Mathematics and Science.*

In this question, the predicate is "like(x, y)," where x= student, and y= subject.

Since there are not all students, so we will use ∀ with negation, so following representation for this:

$$\neg\forall (x) [\ student(x) \rightarrow like(x, Mathematics) \wedge like(x, Science)].$$

5. *Only one student failed in Mathematics.*

In this question, the predicate is "failed(x, y)," where x= student, and y= subject.

Since there is only one student who failed in Mathematics, so we will use following representation for this:

$$\exists(x) [\ student(x) \rightarrow failed\ (x, Mathematics) \wedge\forall (y) [\neg(x==y) \wedge student(y) \rightarrow \neg failed\ (x, Mathematics)].$$

Among the given options, which search algorithm requires less memory?
a) Optimal Search
b) Depth First Search
c)Breadth-First Search
d)Linear Search

Search Algorithm in AI is classified as
a. Un-informed search
b. Informed Search
c. Both a & b
d. None of the above

In which algorithm downhill move is allowed.

a) Simple hill climbing
b) Steepest ascent hill climbing
c) Stochastic hill climbing
d) Stimulat*ed* annealing

A.M. turing developed a technique for determining whether a computer could or could not demonstrate the artificial Intelligence, Presently, this technique is called

a) Turing Test

b) Algorithm

c) Boolean Algebra

d) All of the above

__________________ are mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations.

A. constraints satisfaction problems

B. uninformed search problems

C. local search problems

D. all of the mentioned

What among the following constitutes to the incremental formulation of CSP?
a) Path cost
b) Goal test
c) Successor function
d) Al/ of the mentioned

Which of the Following problems can be modeled as CSP?
a) 8-Puzzle problem
b) 8-Queen problem
c) Map coloring problem
d) All of the mentioned

Among the given options, which is not the required property of Knowledge representation?

a) Inferential Efficiency
b) Inferential Adequacy
c) Representational Verification
d) Representational Adequacy

Two simultaneous search from an initial node to goal node and backward from goal node to initial node, stopping when two meet is an example of

a) Depth First Search
b) Best First Search
c) Bidirectional Search
d) All answer are correct

What is the heuristic function of greedy best-first search?

A. f(n) != h(n)

B. f(n) < h(n)

C. f(n) = h(n)

D. f(n) > h(n)

The advantage of bidirectional search in artificial intelligence is

a. Faster

b. Reduced exploration

c. Both a & *b*

d. None of the above

The metric used to measure the performance of the Bidirectional Search is

a. Completeness

b. Optimality

c. Time and space complexity

d. All of the Above

Program generator is present in _____

a) Goal based agent
b) Simple reflex agent
c) Learning agen*t*
d) Utility based agent

An algorithm A is admissible if ___________

a) It is not guaranteed to return an optimal solution when one exists

b) It is guaranteed to return an optimal solution when one exists

c) It returns more solutions, but not an optimal one

d) It guarantees to return more optimal solutions

A search algorithm takes _________ as an input and returns ________ as an output.
a.
Input, output
b.
Problem, *solution*
c.
Solution, problem
d.
Parameters, sequence of actions

A technique that was developed to determine whether a machine could or could not demonstrate the artificial intelligence known as the___

A)Boolean Algebra

B)Turing Test

C)Logarithm

D)Algorithm

1) Marcus is a man

Solution:

Man (Marcus)

2) Marcus was a Pompien

Solution:

Pompien (Marcus)

This presentation does not take care whether sentence is a past or present.

3) All Pompiens were Romans

Solution:

$\forall x : Pompien(x) \Rightarrow Roman(x)$

4) Every Gardener Likes Sun

Solution:

$\forall x : Gardener(x) \Rightarrow likes(x,Sun)$

5) All purple Mushrooms are poisonous

Solution:

$\forall x : Mushroom(x) \land purple(x) \Rightarrow poisonous(x)$

6) Everyone is Loyal to Someone

Solution:

$\forall x \exists y : loyal(x,y)$

7) Everyone loves everyone

Solution:

$\forall x \forall y : loves(x,y)$

8) Everyone loves everyone except himself

Solution:

$\forall x \forall y : \text{loves}(x,y) \land \neg \text{loves}(x,x)$

9) All Romans were either loyal to Ceasar or hated him.

Solution:

$\forall x : \text{Roman}(x) \Rightarrow \text{Loyal}(x,\text{Ceasar}) \lor \text{Hated}(x,\text{Ceasar})$

10) People only try to assasinate rulers they are not loyal to.

Solution:

$$\forall x \forall y : Person(x) \land Ruler(y) \land \neg loyal(x,y) \Rightarrow tryassasinate(x,y)$$

Wrong Presentation….Why?

10) People only try to assasinate rulers they are not loyal to.

$$\forall x \forall y : Person(x) \wedge Ruler(y) \wedge \neg loyal(x,y)$$

$$\Rightarrow tryassasinate\ (x,y)$$

$$\forall x \forall y : Person(x) \wedge Ruler(y) \wedge tryassasinate\ (x,y)$$

$$\Rightarrow \neg loyal(x,y,)$$

11) Anyone who is married and has more than one spouse is a Bigamist

Analyse this

$$\forall x : Married(x) \wedge (no.of.spouse(x) > 1) \Rightarrow bigamist(x)$$

Is it correct…….No…Why?
Actually it said every ‘x’ has more than 1 spouse…
But we have to check it as it is not true for all…
So correct presentation is…

11) Anyone who is married and has more than one spouse is a Bigamist

Solution:

$$\forall x : Married(x) \wedge gt(no.of.spouse(x),1) \Rightarrow bigamist(x)$$

- Shows the use of predicate logic as a way of representing knowledge.
- 1. Marcus was a man.
- 2. Marcus was a Pompeian.
- 3. All Pompeians were Romans.
- 4. Caesar was a ruler.
- 5. All Pompeians were either loyal to Caesar or hated him.
- 6. Everyone is loyal to someone.
- 7. People only try to assassinate rulers they are not loyal to.
- 8. Marcus tried to assassinate Caesar.

The facts described by well-formed formulas (*wffs*)
as follows:

1. Marcus was a man.
- man(Marcus)

2. Marcus was a Pompeian.
- Pompeian(Marcus)

3. All Pompeians were Romans.
- $\forall x$: Pompeian(x) $\rightarrow$ Roman(x)

4. Caesar was a ruler.
- ruler(Caesar)

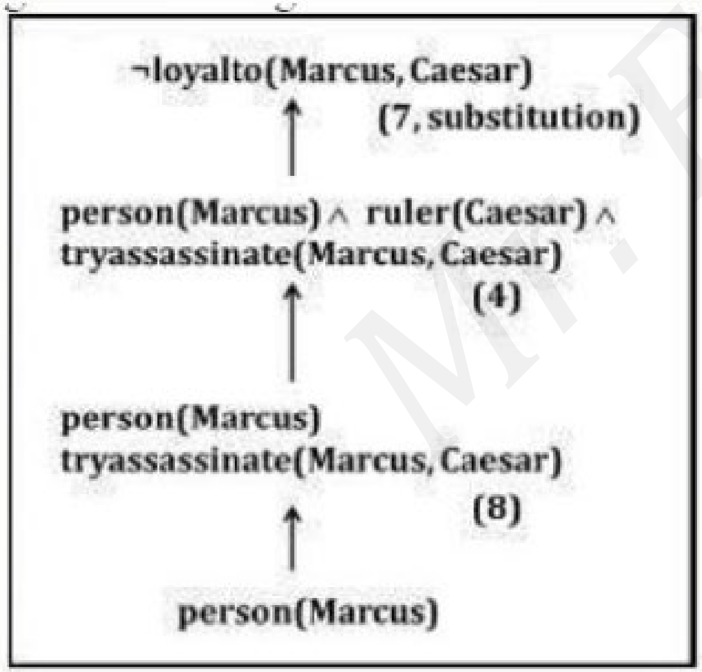
5. All Pompeians were either loyal to Caesar or hated him.
- inclusive-or
- $\forall x$: Roman(x) $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar)
- exclusive-or
- $\forall x$: Roman(x) $\rightarrow$ (loyalto(x, Caesar) $\wedge \neg$ hate(x, Caesar)) $\vee$ ($\neg$loyalto(x, Caesar) $\wedge$ hate(x, Caesar))

6. Everyone is loyal to someone.
- $\forall x$: $\exists y$: loyalto(x, y)

7. People only try to assassinate rulers they are not loyal to.
- $\forall x$: $\forall y$: person(x) $\wedge$ ruler(y) $\wedge$ tryassassinate(x, y) $\rightarrow \neg$loyalto(x, y)

8. Marcus tried to assassinate Caesar.
- tryassassinate(Marcus, Caesar)

- Now suppose if we want to use these statements to answer the question: ***Was Marcus loyal to Caesar?***
- Also, Now let’s try to produce a formal proof, reasoning backward from the desired goal:
- ¬Ioyalto(Marcus, Caesar)



Representing instance and isa relationship

INSTANCE AND ISA RELATIONSHIPS

- Instance relationships :
  - Relationship between a class and its object
  - eg., Marcus is an instance of man
- Isa relationship:
  - Relationship between a base class and a derived class
  - Derived class isa base class
  - e.g., Pompeian is a Roman
- Axioms can also be represented with isa and instance relationships as predicates!
  - man(Marcus) : instance(Marcus, man)
  - Pompeian(Marcus) : instance(Marcus, Pompeian)
  - $\forall x: Pompeian(x) \rightarrow Roman(x)$: isa(Pompeian, Roman)
  - ruler(Caesar) : instance(Caesar, ruler)

Representing instance and isa relationship

Representing instance and isa relationship

Marcus was a man

Marcus was a Pompeian

All Pompeian were roman

Caesar was a ruler

All roman were either loyal to caser or hated him

Representing instance and isa relationship

1.Marcus was a man

2.Marcus was a Pompeian

3. All Pompeian were roman

4. Caesar was a ruler

5. All roman were either loyal to

caser or hated him

1. Man(Marcus).
2. Pompeian(Marcus).
3. $\forall x$: Pompeian(x) $\rightarrow$ Roman(x).
4. ruler(Caesar).
5. $\forall x$: Roman(x) $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

Representing instance and isa relationship

1.Marcus was a man

2.Marcus was a Pompeian

3. All Pompeian were roman

4. Caesar was a ruler

5. All roman were either loyal to

caser or hated him

1. Man(Marcus).
2. Pompeian(Marcus).
3. $\forall x$: Pompeian(x) $\rightarrow$ Roman(x).
4. ruler(Caesar).
5. $\forall x$: Roman(x) $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

1. instance(Marcus, man).
2. instance(Marcus, Pompeian).
3. $\forall x$: instance(x, Pompeian) $\rightarrow$ instance(x, Roman).
4. instance(Caesar, ruler).
5. $\forall x$: instance(x, Roman). $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

Representing instance and isa relationship

1.Marcus was a man

2.Marcus was a Pompeian

3. All Pompeian were roman

4. Caesar was a ruler

5. All roman were either loyal to caser or hated him

1. instance(Marcus, man).
2. instance(Marcus, Pompeian).
3. $\forall x$: instance(x, Pompeian) $\rightarrow$ instance(x, Roman).
4. instance(Caesar, ruler).
5. $\forall x$: instance(x, Roman). $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

$\downarrow$

1. instance(Marcus, man).
2. instance(Marcus, Pompeian).
3. isa(Pompeian, Roman)
4. instance(Caesar, ruler).
5. $\forall x$: instance(x, Roman). $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).
6. $\forall x$: $\forall y$: $\forall z$: instance(x, y) $\wedge$ isa(y, z) $\rightarrow$ instance(x, z).

Computable Function and Predicates

- • To express simple facts, such as the following greater-than and less-than relationships:
  - gt(1,O) It(0,1) gt(2,1) It(1,2) gt(3,2) It(2,3)

- • It is often also useful to have computable functions as well as computable predicates.
  - Thus we might want to be able to evaluate the truth of gt(2 + 3,1)

- • To do so requires that we first compute the value of the plus function given the arguments 2 and 3, and then send the arguments 5 and 1 to gt.

Computable Function and Predicates

1.Marcus was a man.

2.Marcus was a Pompeian.

3.Marcus was born in 40 A.D.

4.All men are mortal.

5.All Pompeians died when the volcano erupted in 79 A.D.

6.No mortal lives longer than 150 years.

7.It is now 1991.

8.Alive means not dead.

9.If someone dies, then he is dead at all later times.

Computable Function and Predicates

Consider the following set of facts, again involving Marcus:

1. Marcus was a man.

   $man(Marcus)$

   Again we ignore the issue of tense.
2. Marcus was a Pompeian.

   $Pompeian(Marcus)$

3. Marcus was born in 40 A.D.

   $born(Marcus, 40)$

4. All men are mortal.

   $\forall x: man(x) \rightarrow mortal(x)$

5. All Pompeians died when the volcano erupted in 79 A.D.

   $erupted(volcano, 79) \wedge \forall x : [Pompeian(x) \rightarrow died(x, 79)]$

6. No mortal lives longer than 150 years.

$$\forall x : \forall t_1 : \forall t_2 : mortal(x) \wedge born(x, t_1) \wedge gt(t_2 - t_1, 150) \rightarrow dead(x, t_2)$$

7. It is now 1991.

$$now = 1991$$

8. Alive means not dead.

$$\forall x : \forall t : [alive(x, t) \rightarrow \neg dead(x, t)] \wedge [\neg dead(x, t) \rightarrow alive(x, t)]$$

9. If someone dies, then he is dead at all later times.

$$\forall x : \forall t_1 : \forall t_2 : died(x, t_1) \wedge gt(t_2, t_1) \rightarrow dead(x, t_2)$$

A Set of Facts about Marcus:

1. $man(Marcus)$
2. $Pompeian(Marcus)$
3. $born(Marcus, 40)$
4. $\forall x : man(x) \rightarrow mortal(x)$
5. $\forall : Pompeian(x) \rightarrow died(x, 79)$
6. $erupted(yolcano, 79)$
7. $\forall_x : \forall t_1 : \forall t_2 : mortal(x) \wedge born(x, t_1) \wedge gt(t_2 - t_1, 150) \rightarrow dead(x, t_2)$
8. $now = 1991$
9. $\forall x : \forall t: [alive(x, t) \rightarrow \neg dead(x, t)] \wedge [\neg dead(x, t) \rightarrow alive(x, t)]$
10. $\forall x : \forall t_1 : \forall t_2 : died(x, t_1) \wedge gt(t_2, t_1) \rightarrow dead(x, t_2)$

we want to answer the question “Is Marcus alive?”

One Way of Proving That Marcus Is Dead

$$\neg alive(Marcus, now)$$
$\uparrow$ (9, substitution)
$$dead(Marcus, now)$$
$\uparrow$ (10, substitution)
$$died(Marcus, t_1) \wedge gt(now, t_1)$$
$\uparrow$ (5, substitution)
$$Pompeian(Marcus) \wedge gt(now, 79)$$
$\uparrow$ (2)
$$gt(now, 79)$$
$\uparrow$ (8, substitute equals)
$$gt(1991, 79)$$
$\uparrow$ (compute gt)
$$nil$$

The term nil at end of each proof indicate that the list of conditions remaining is empty
So the proof has succeeded.

Another Way of Proving That Marcus Is Dead

$$\neg alive(Marcus, now)$$
$$\uparrow \quad (9,\ substitution)$$
$$dead(Marcus, now)$$
$$\uparrow \quad (7,\ substitution)$$
$$mortal(Marcus) \wedge$$
$$born(Marcus, t_1) \wedge$$
$$gt(now - t_1, 150)$$
$$\uparrow \quad (4,\ substitution)$$
$$man(Marcus) \wedge$$
$$born(Marcus, t_1) \wedge$$
$$gt(now - t_1, 150)$$
$$\uparrow \quad (1)$$
$$born(Marcus, t_1) \wedge$$
$$gt(now - t_1, 150)$$
$$\uparrow \quad (3)$$
$$gt(now - 40, 150)$$
$$\uparrow \quad (8)$$
$$gt(1991 - 40, 150)$$
$$\uparrow \quad (compute\ minus)$$
$$gt(1951, 150)$$
$$\uparrow \quad (compute\ gt)$$
$$nil$$

- Very simple conclusion can require many steps to prove.
- A variety of processes such as Matching, substitution and application of modus ponens are involved in production of proof.