PES UNIVERSITY ONLINE

Automata Formal Languages & Logic

Preet Kanwal

Department of Computer Science & Engineering

Automata Formal Languages & Logic

Unit 5

Preet Kanwal

Department of Computer Science & Engineering

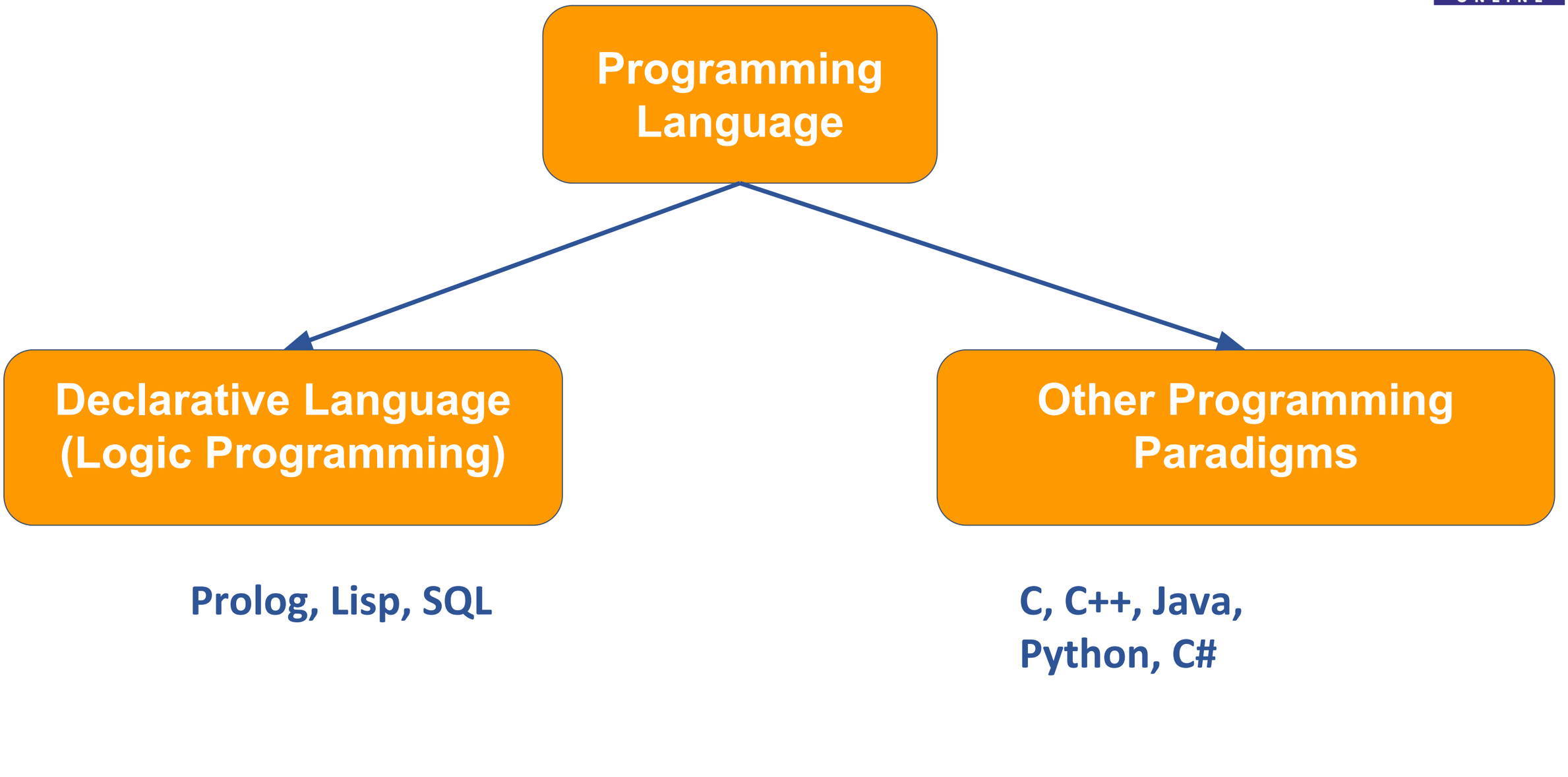
Automata Formal Languages and Logic

Unit 5 - Formal Languages and Logic

PES UNIVERSITY ONLINE

In this unit, we look at the application of what we have studied so far.

We look at the design and working of a programming language.


Programming Language
Declarative Language (Logic Programming)
Other Programming Paradigms
Prolog, Lisp, SQL
C, C++, Java, Python, C#

What is the difference??

- **Declarative programming–**
  - Describes what you want to do, and not how you want to do it.
  - It is left up to the compiler/interpreter to figure out the how.
  - expresses the logic of a computation without describing its control flow.
- **Other Programming Paradigms–**
  - Focuses on how to execute,
  - Defines control flow as statements that change a program state.
  - Programmer understands and has access to each step of the code.
  - Lacks mechanism of deriving facts from other facts. Each update to the data structure is done by a domain-specific procedure provided by the programmer.

- In imperative languages, the description of the problem is incorporated implicitly in this specification, and usually it is not possible to clearly distinguish between the description of the problem, and the method used for its solution.
- In logic programming, the description of the problem and the method for solving it are explicitly separated from each other.

algorithm = logic + control

- The term ‘logic’ in this equation indicates the descriptive component of the algorithm, that is, the description of the problem; the term ‘control’ indicates the component that tries to find a solution, taking the description of the problem as a point of departure.
- So, the logic component defines what the algorithm is supposed to do; the control component indicates how it should be done.

PES UNIVERSITY ONLINE

In order to learn the design and working of a logic programming language like Prolog one must study the following :

- **Basic concepts of logic (via Propositional logic)**
- **First order Predicate Logic (more powerful)**

PES UNIVERSITY ONLINE

Prolog

PROgramming in **LO**gic

- High level interactive language
- Logic Programming Language
- Declarative language
- Emphasis on what rather than how.
- It is widely used in the field of AI.

It is based on the principles of **First Order Predicate Logic.**

https://www.cs.ru.nl/P.Lucas/teaching/CS3510/intro-prol.pdf

Specifying a logical program (Prolog program) amounts to:

- Specifying the facts concerning the objects and relations between objects relevant to the problem at hand;
- Specifying the rules concerning the objects and their interrelationships;
- Posing queries concerning the objects and relations.

A Prolog program consists of :

Solving a problem in PROLOG starts with describing the objects that are relevant to the particular problem, and the relationships that exist between them.

When we have identified all relevant objects and relations, it must be specified which facts and rules hold for the objects and their interrelationships.

When all facts and rules have been identified, then a specific problem may be looked upon as a query concerning the objects and their interrelationships.

Prolog Basic Constructs

fact	A.	
rule	A :- B_1,B_2,B_3,.....B_n.	if B_1,B_2,B_3,....B_n then A or $B_1 \Lambda B_2 \Lambda B_3 \Lambda ....\Lambda B_n \rightarrow A$
	A :- B_1;B_2;B_3,.....B_n.	$B_1 V B_2 V B_3 V ....V B_n \rightarrow A$
query	?- B1, . . . , Bn	

Facts and rules form the *knowledge base*

Note :

Following terms are used interchangeably:

fact, sentence, axiom - data which is given without being derived from other facts/sentences.

Deriving new sentences from existing facts is called inferencing.

```
// A simple logical program
/* facts */
bigger(elephant, horse).
bigger(horse, monkey).
bigger(monkey, cat).
bigger(cat, ant).
bigger(lion, monkey).
```

Query the following:

```
?- find(elephant, cat)
?- find(X, monkey)
?- find(monkey, X)
?- find(X, Y)
```

```
/*rules*/
find(X, Z):- (bigger(X,Y), bigger(Y,Z)) ; bigger(X,Z).
```

Execute here : https://swish.swi-prolog.org/example/examples.swinb

PES UNIVERSITY ONLINE

Procedure View of Prolog:

Like normal *imperative* programming languages, a Prolog program consists of procedures, which are collections of statements. A program is executed by running a sequence of statements. Each of these statements executes by "calling" a procedure, which calls other procedures, etc.

```
fact(0,1).
fact(N,Result) :-
    N > 0,
        M is N - 1,
    fact(M,Y),
        Result is Y * N.
```

Query the following:

?- fact(0, 1)

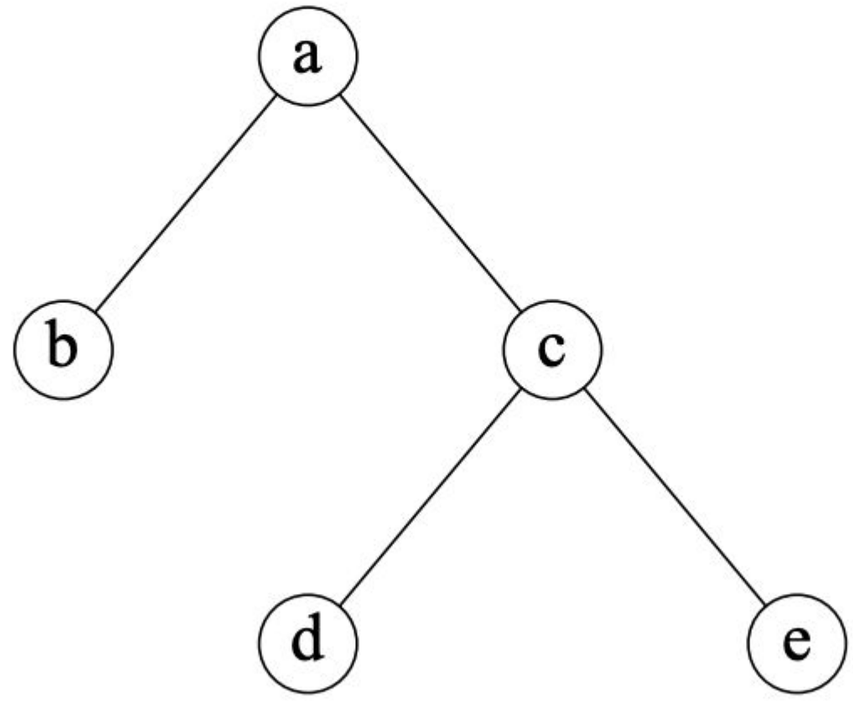
?- fact(5, Value)

?- fact(3, 6)

Execute here : https://swish.swi-prolog.org/example/examples.swinb

A binary tree Example:

```
/*1*/ branch(a,b).
/*2*/ branch(a,c).
/*3*/ branch(c,d).
/*4*/ branch(c,e).
/*5*/ path(X,X).
/*6*/ path(X,Y) :- branch(X,Z), path(Z,Y).
```



Query the following:

?- path(a,a)

?- path(a,d)

Execute here : https://swish.swi-prolog.org/example/examples.swinb

PROLOG interpreter examines rules in the order in which they have been specified in the database.

The database created is called Knowledge Base. It is a list of facts (predicates) and rules about the domain(possible world-situation of interest).

What is Logic :

A formal language

- Syntax – what expressions are legal
- Semantics – what legal expressions mean - It's truth value!
- Proof system – a way of manipulating syntactic expressions to get other syntactic expressions (which will tell us something new)

Automata Formal Languages & Logic

Propositional Logic (Basics)

- We would like to acknowledge efforts of Prof. Channabankapur, Associate Professor, Dept. of CSE in preparation of these slides.

Preet Kanwal

Department of Computer Science & Engineering

Automata Formal Languages and Logic

Unit 5 - Propositional Logic

PES UNIVERSITY ONLINE

Outline:

- What is a Proposition?
- What is Propositional Logic?
- Atomic Sentence
- Complex Sentences using Logical Connectives

(¬, v , ∧, →, ↔)

- Syntax & Semantics of Propositional Logic
- Tautology
- Contradiction
- Logical Equivalence (Laws of Logic)
- Logical Entailment
- Methods to prove Logical Entailment
- Rules of Inference
- Proof by Resolution

Proposition: A proposition is a declarative sentence that is either true or false, but not both.

Propositional Logic (Propositional Calculus):

- The area of logic that deals with propositions.
- It is also a Declarative language.
- Its expressive power is low (limited to simple environments).

Proposition or not?

- Bangalore is the capital city of India.
- New Delhi is the capital city of India.
- Do you like coffee?
- Get me a cup of coffee.
- 1 + 2 = 3
- 2 + 3 = 4
- x + 3 = 10
- x + y = z
- Mahatma Gandhi was born in the year 1869.
- It’s raining today.
- Earth is flat.

Following are Propositions(Atomic Sentences):

- New Delhi is the capital city of India. // true
- Bangalore is the capital city of India. // false
- 1 + 2 = 3 // true
- 2 + 3 = 4 // false
- Mahatma Gandhi was born in 1869. // true
- It's raining today. // false
- Earth is flat. // false

These are NOT propositions:

- Get me a cup of coffee.//not a declarative sentence
- Do you like coffee?//not a declarative sentence
  - Though the response to the question could be yes or no; a binary response.
- x + 1 = 2 // has a variable
  - It's a declarative sentence, but depending on the value of 'x', it can be true or false.
  - It can be turned into a proposition by giving a value for 'x'.
- x + y = z. // has variables

Compound Propositions/ Complex Sentence: are propositions formed from existing propositions using **logical connectives/operators.**

Logical Operators:

- Negation ($\neg$)
- Conjunction ($\wedge$)
- Disjunction ($\vee$)
- Conditional Statement ($\rightarrow$)
- Biconditional Statement ($\leftrightarrow$)

Negation:

- Let p be a proposition. The negation of p, denoted by ¬p, is the statement "It is not the case that p".
- The proposition ¬p is read "not p".
- The truth value of the negation of p is the opposite of the truth value of p.

Eg: Let p be "Today is Friday"
¬p: "It's not the case that today is Friday" or
¬p: "Today is not Friday"

p	¬p
F	T
T	F

Conjunction: Let p and q be propositions.

- The conjunction of p and q, denoted by $p \wedge q$, is the proposition "p and q".
- The conjunction $p \wedge q$ is true when both p and q are true and is false otherwise.

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

Examples for Conjunction:

Example 1:

p: “He brought pen to the class”
q: “He brought notebook to the class”
p∧q: “He brought pen and notebook to the class”

Example 2:

p: “Tea is served in the conference”
q: “Coffee is served in the conference”
p∧q: “Tea and Coffee are served in the conference”

Disjunction: Let p and q be propositions.

- The disjunction of p and q, denoted by $p \vee q$, is the proposition "p or q".
- The disjunction $p \vee q$ is false when both p and q are false and is true otherwise.

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

Example for Disjunction:

p: “You are tall”
q: “You are an adult”
p∨q: “you are tall or you are an adult”

Conditional Statement: Let p and q be propositions. The "conditional statement" $p \rightarrow q$ is false when p is true and q is false, and true otherwise. Conditional statements are also called implications.

In $p \rightarrow q$,
p is called hypothesis,
premise
or antecedent
q is called conclusion
or consequence

Truth table of Conditional Statement:

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

Note :

- Implies($p \rightarrow q$) doesn't mean Suggestion/Causation.
  For example :
  a) Green is a color implies Austin is in Texas
  b) 9 is even implies x is 5.

Both statements are perfectly okay from the logic point of view.

- All Statements must have a truth value (even though it doesn't make sense).

https://www.youtube.com/watch?v=3RFamYOCEHA

Biconditional Statement: Let p and q be propositions. The biconditional statement p↔q is the proposition "p if and only if q". The biconditional statement p↔q is true when p and q have the same truth values, and is false otherwise.

Biconditional statements are also called as bi-implications.

Example:

p→q: "if the switch is on, then the current flows".

p↔q: "current flows if and only if the switch is on".

p	q	p↔q
F	F	**T**
F	T	**F**
T	F	**F**
T	T	**T**

p	q	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p)$	$p \leftrightarrow q$
F	F	T	T	**T**	**T**
F	T	T	F	**F**	**F**
T	F	F	T	**F**	**F**
T	T	T	T	**T**	**T**

$\therefore$ $p \leftrightarrow q$ is equivalent to $(p \rightarrow q) \wedge (q \rightarrow p)$

Syntax & Semantics of Propositional Logic :

Syntax - Specified using Grammar.

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{True} \mid \textit{False} \mid P \mid Q \mid R \mid \ldots \\
\textit{ComplexSentence} &\rightarrow (\textit{Sentence}) \mid [\textit{Sentence}] \\
&\mid \neg \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence}
\end{aligned}
$$

OPERATOR PRECEDENCE : $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

Semantics - Based on Truth value of the sentences. Meaning of the sentence is a function of meaning of its parts.

Let p, q, and r be the propositions
p: You have the flu.
q: You miss the final examination.
r: You pass the course.

Express the propositions as English sentences:

- $p \rightarrow q$
- $\neg q \leftrightarrow r$
- $q \rightarrow \neg r$
- $(p \rightarrow \neg r) \vee (q \rightarrow \neg r)$
- $(p \wedge q) \vee (\neg q \wedge r)$

Solution :

Let p, q, and r be the propositions

p: You have the flu.

q: You miss the final examination.

r: You pass the course.

Express the propositions as English sentences:

- $p \rightarrow q$:If you have the flu then you miss the final examination.
- $\neg q \leftrightarrow r$: You pass the course iff you do not miss the final examination.
- $q \rightarrow \neg r$: If you miss the final examination the you will not pass the course.
- $(p \rightarrow \neg r) \vee (q \rightarrow \neg r)$:If you have the flu then you will not pass the course or if you miss the final examination then you will not pass the course.
- $(p \wedge q) \vee (\neg q \wedge r)$: You have the flu and you miss the final examination or you will not miss the final examination and you pass the course.

Let p and q be the propositions
p: You drive over 60 kmph.
q: You get a speeding ticket.

Write the following propositions using p, q and logical connectives.

a. You do not drive over 60 kmph
b. You drive over 60 kmph, but you do not get a speeding ticket.
c. You will get a speeding ticket if you drive over 60 kmph.
d. If you do not drive over 60 kmph, then you will not get a speeding ticket.
e. You get a speeding ticket, but you do not drive over 60 kmph.

Solution:

p: You drive over 60 kmph. q: You get a speeding ticket.

Write the following propositions using p, q and logical connectives.

a. You do not drive over 60 kmph: $\neg p$.
b. You drive over 60 kmph, but you do not get a speeding ticket : $p \wedge \neg q$.
c. You will get a speeding ticket if you drive over 60 kmph : $p \rightarrow q$
d. If you do not drive over 60 kmph, then you will not get a speeding ticket : $\neg p \rightarrow \neg q$
e. You get a speeding ticket, but you do not drive over 60 kmph : $q \wedge \neg p$.

PES UNIVERSITY ONLINE

Determine the truth values of the following propositions.

a. 2+2=4 if and only if 1+1=2.
b. 1+1=2 if and only if 2+3=4.
c. 0>1 if and only if 2>1.
d. If 1+1=2, then 2+2=5.
e. If 1+1=3, then 2+2=4.
f. If 1+1=3, then 2+2=5.
g. If 2+2=4, then 1+2=3.
h. 1+1=3 if and only if monkeys can fly.
i. If monkeys can fly, then 1+1=3
j. If 1+1=3, then unicorns exist.
k. If 1+1=3, then dogs can fly.
l. If 1+1=2, then dogs can fly.

PES UNIVERSITY ONLINE

Solution :

Determine the truth values of the following propositions.

a. 2+2=4 if and only if 1+1=2. True
b. 1+1=2 if and only if 2+3=4. False
c. 0>1 if and only if 2>1. False
d. If 1+1=2, then 2+2=5. False
e. If 1+1=3, then 2+2=4. True
f. If 1+1=3, then 2+2=5.True
g. If 2+2=4, then 1+2=3. True
h. 1+1=3 if and only if monkeys can fly. True
i. If monkeys can fly, then 1+1=3 True
j. If 1+1=3, then unicorns exist. True
k. If 1+1=3, then dogs can fly. True
l. If 1+1=2, then dogs can fly. False

Tautology: is a compound proposition that is **always true** no matter what the truth values of the propositions that occur in it.

Contradiction: is a compound proposition that is **always false** no matter what the truth values of the propositions that occur in it. .

p	q	$(p \rightarrow q) \wedge (q \rightarrow p) \leftrightarrow (p \leftrightarrow q)$	$\neg(p \rightarrow q) \wedge \neg(p \wedge \neg q)$
F	F	T	F
F	T	T	F
T	F	T	F
T	T	T	F

Logical Equivalence: The compound propositions p and q are called logically equivalent if $p \leftrightarrow q$ is a tautology. The notation $p \equiv q$ denotes that p and q are logically equivalent. (Their truth value is the same)

In other words, compound propositions that have the same truth values in all possible cases are called logically equivalent. $(p \rightarrow q) \wedge (q \rightarrow p) \equiv (p \leftrightarrow q)$

Note: The symbol ≡ is not a logical connective.
Hence "p ≡ q" is not a compound proposition.
"p ≡ q" means $p \leftrightarrow q$ is a tautology.

PES UNIVERSITY ONLINE

Using truth table, show that

1. $(p \leftrightarrow q) \equiv (p \rightarrow q) \wedge (q \rightarrow p)$.
2. $p \rightarrow q \equiv \neg p \vee q$

Example 1: Using truth table, show that $(p \leftrightarrow q) \equiv (p \rightarrow q) \wedge (q \rightarrow p)$.

p	q	$p \leftrightarrow q$	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p)$	$(p \leftrightarrow q) \leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$
F	F					
F	T					
T	F					
T	T					

Example 1: Using truth table, show that $(p \leftrightarrow q) \equiv (p \rightarrow q) \wedge (q \rightarrow p)$.

p	q	$p \leftrightarrow q$	$(p \rightarrow q) \wedge (q \rightarrow p)$	$(p \leftrightarrow q) \leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$
F	F	T	T	T
F	T	F	F	T
T	F	F	F	T
T	T	T	T	T

Truth table demonstrates,

$(p \leftrightarrow q) \leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$ **is a tautology.**

Therefore, $(p \leftrightarrow q) \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

Example 2: Using truth tables, show that $p \rightarrow q$ and $\neg p \vee q$ are logically equivalent.

p	q	$\neg p$	$p \rightarrow q$	$\neg p \vee q$	$(p \rightarrow q) \leftrightarrow (\neg p \vee q)$
F	F				
F	T				
T	F				
T	T				

Example 2: Using truth tables, show that $p \rightarrow q$ and $\neg p \vee q$ are logically equivalent.

$(p \rightarrow q) \leftrightarrow (\neg p \vee q)$ is a tautology.

Therefore, $(p \rightarrow q) \equiv (\neg p \vee q)$

p	q	$\neg p$	$p \rightarrow q$	$\neg p \vee q$	$(p \rightarrow q) \leftrightarrow (\neg p \vee q)$
F	F	T	T	T	T
F	T	T	T	T	T
T	F	F	F	F	T
T	T	F	T	T	T

PES UNIVERSITY ONLINE

Laws of Logic

Equivalence	Name of Identity
$p \wedge T \equiv p$ $p \vee F \equiv p$	Identity Laws
$p \wedge F \equiv F$ $p \vee T \equiv T$	Domination Laws
$p \wedge p \equiv p$ $p \vee p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \wedge q \equiv q \wedge p$ $p \vee q \equiv q \vee p$	Commutative Laws
$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ $(p \vee q) \vee r \equiv p \vee (q \vee r)$	Associative Laws
$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	Ditributive Laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's Laws
$p \wedge (p \vee q) \equiv p$ $p \vee (p \wedge q) \equiv p$	Absorption Laws
$p \wedge \neg p \equiv F$ $p \vee \neg p \equiv T$	Negation Laws

Q: Prove the following logical equivalences using the laws of logic (without using the truth table).

1. $\neg(p \rightarrow q) \equiv p \wedge \neg q$
2. $p \wedge (p \rightarrow q)) \equiv p \wedge q$

Homework

3. $\neg(p \vee (\neg p \wedge q)) \equiv \neg(p \vee q)$

Q: Prove the following logical equivalences using the laws of logic (without using the truth table).

1. $\neg(p \rightarrow q) \equiv p \wedge \neg q$

   Soln: LHS $\equiv \neg(p \rightarrow q)$

Q: Prove the following logical equivalences using the laws of logic (without using the truth table).

1. $\neg(p \rightarrow q) \equiv p \wedge \neg q$

Soln: LHS $\equiv \neg(p \rightarrow q)$

$\equiv \neg(\neg p \vee q)$ $\because (p \rightarrow q) \equiv \neg p \vee q$

$\equiv \neg(\neg p) \wedge \neg q$ De Morgan's law

$\equiv p \wedge \neg q$

$\therefore \neg(p \rightarrow q) \equiv p \wedge \neg q$

2. $p \wedge (p \rightarrow q)) \equiv p \wedge q$

Soln: LHS $\equiv p \wedge (p \rightarrow q))$

$\equiv$

2. $p \wedge (p \rightarrow q)) \equiv p \wedge q$

Soln: LHS $\equiv p \wedge (p \rightarrow q))$

$\equiv p \wedge (\neg p \vee q))$ $\because (p \rightarrow q) \equiv (\neg p \vee q)$

$\equiv (p \wedge \neg p) \vee (p \wedge q)$ Distribution law

$\equiv F \vee (p \wedge q)$

$\equiv p \wedge q$

$\therefore p \wedge (p \rightarrow q)) \equiv p \wedge q$

Homework

3. $\neg(p \vee (\neg p \wedge q)) \equiv \neg(p \vee q)$

Soln: $LHS \equiv \neg(p \vee (\neg p \wedge q))$

$\equiv$

3. $\neg(p \vee (\neg p \wedge q)) \equiv \neg(p \vee q)$

Soln: $LHS \equiv \neg(p \vee (\neg p \wedge q))$

$\equiv \neg p \wedge \neg(\neg p \wedge q))$ De Morgan's law

$\equiv \neg p \wedge (\neg(\neg p) \vee (\neg q))$ De Morgan's law

$\equiv \neg p \wedge (p \vee \neg q)$

$\equiv (\neg p \wedge p) \vee (\neg p \wedge \neg q)$ Distribution law

$\equiv F \vee (\neg p \wedge \neg q)$

$\equiv \neg p \wedge \neg q$

$\equiv \neg(p \vee q)$ De Morgan's law

$\therefore \neg(p \vee (\neg p \wedge q)) \equiv \neg(p \vee q)$

PES UNIVERSITY ONLINE

Q: Prove the following expressions are tautologies using the laws of logic (without using the truth table)

1. $(p \wedge q) \rightarrow (p \vee q)$
2. $(p \wedge q) \rightarrow (p \rightarrow q)$

Q: Prove the following expressions are tautologies using the laws of logic (without using the truth table)

1. $(p \land q) \rightarrow (p \lor q)$

   Soln: $(p \land q) \rightarrow (p \lor q)$

PES UNIVERSITY ONLINE

Q: Prove the following expressions are tautologies using laws of logic (without using truth table)

1. $(p \wedge q) \rightarrow (p \vee q)$

Soln: $(p \wedge q) \rightarrow (p \vee q)$

$\equiv \neg(p \wedge q) \vee (p \vee q)$ $\quad \because (p \rightarrow q) \equiv (\neg p \vee q)$

$\equiv (\neg p \vee \neg q) \vee (p \vee q)$ De Morgan law

$\equiv \neg p \vee \neg q \vee p \vee q$ Associative law

$\equiv \neg p \vee p \vee \neg q \vee q$ Commutative law

$\equiv T \vee T$

$\equiv T$

$\therefore (p \wedge q) \rightarrow (p \vee q)$ is a tautology

2. $(p \wedge q) \rightarrow (p \rightarrow q)$

Soln: $(p \wedge q) \rightarrow (p \rightarrow q)$

$\equiv$

2. $(p \wedge q) \rightarrow (p \rightarrow q)$

Soln: $(p \wedge q) \rightarrow (p \rightarrow q)$

$\equiv \neg(p \wedge q) \vee (\neg p \vee q)$ $\because (p \rightarrow q) \equiv (\neg p \vee q)$

$\equiv (\neg p \vee \neg q) \vee (\neg p \vee q)$ De Morgan's law

$\equiv \neg p \vee \neg q \vee \neg p \vee q$ Associative law

$\equiv \neg p \vee \neg p \vee \neg q \vee q$ Commutative law

$\equiv \neg p \vee \neg q \vee q$

$\equiv \neg p \vee T$

$\equiv T$ Domination law

$\therefore (p \wedge q) \rightarrow (p \rightarrow q)$ is a tautology

Logical entailment (α ⊨ β):

Logical entailment describes the relationship between statements that hold true when one statement logically follows from one or more statements.

A rule of the form $(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \rightarrow q$ is called a logical entailment(or a valid argument). It must be a Tautology.

p_1
p_2
...
p_n

∴ q

The sentence q logically follows from $(p_1 \wedge p_2 \wedge \ldots \wedge p_n)$ which is another sentence.
$(p_1 \wedge p_2 \wedge \ldots \wedge p_n)$ are called premises and q is called the conclusion .We can say conclusion is entailed by the premises

Mathematically, we can specify logical entailment as: α ⊨ β
We can say, α ≡ β iif α ⊨ β and β ⊨ α

Logical Entailment can be done via the following methods:

1. Model Checking (by constructing the truth table)
2. Rules of Reference (Theorem Proving)
   - Using Resolution rule alone we can construct the required proof. (one rule is all you need to prove things)

Rules of Inference:

- We can use a truth table to show that an argument form is valid.

- But when we have more propositional variables involved in the argument, solving by truth table is too laborious (having just 30 variables will require over a billion rows in the truth table).

- That's when we need well-known argument forms to simplify a complex argument form at hand.

- These well-known valid argument forms are called Rules of Inference.

Rule of Inference	Tautology	Name
$\begin{array}{l} p \\ p \rightarrow q \\ \hline \therefore q \end{array}$	$(p \wedge (p \rightarrow q)) \rightarrow q$	Modus Ponens
$\begin{array}{l} \neg q \\ p \rightarrow q \\ \hline \therefore \neg p \end{array}$	$(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$	Modus Tollens
$\begin{array}{l} p \rightarrow q \\ q \rightarrow r \\ \hline \therefore p \rightarrow r \end{array}$	$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$	Hypothetical syllogism
$\begin{array}{l} \neg p \\ p \vee q \\ \hline \therefore q \end{array}$	$(\neg p \wedge (p \vee q)) \rightarrow q$	Disjunctive Syllogism
$\begin{array}{l} p \\ \hline \therefore (p \vee q) \end{array}$	$p \rightarrow (p \vee q)$	Addition
$\begin{array}{l} (p \wedge q) \rightarrow r \\ \hline \therefore p \rightarrow (q \rightarrow r) \end{array}$	$((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$	Exportation
$\begin{array}{l} p \vee q \\ \neg p \vee r \\ \hline \therefore q \vee r \end{array}$	$((p \vee q) \wedge (\neg p \vee r)) \rightarrow q \vee r$	Resolution

PES UNIVERSITY ONLINE

Proof by Resolution :

- Pretty much every modern automated theorem-prover is implements Resolution technique.
- It's apparently the best way for computers to think about proving things.

Resolution rule (it is an ∧ of ∨'s) :
Each premise has to in the form a disjunction!

$$\frac{p \vee q \qquad \neg p \vee r}{\therefore q \vee r} \quad \text{or} \quad \frac{p \qquad \neg p \vee r}{r}$$

Proof Strategy is called **Resolution Refutation**(act of proving), It contains 3 steps:

1. Convert all of your sentences to **Conjunctive Normal Form**(CNF)
2. **Negate the desired conclusion** (then convert to CNF) - We are essentially doing a proof by Contradiction. (You assert that the thing that you're trying to prove is false, and then you try to derive a contradiction.)
3. Add each of these clauses as a premise of your proof.
4. **Apply resolution rule until** either –
   - If we derive false (contradiction), then the conclusion follows from the axioms
   - If we can't apply any more, then the conclusion cannot be proved from the axioms.

Resolution refutation is sound and complete :

It's guaranteed that you'll always either prove false, or run out of possible steps.

- **It's complete, because it always generates an answer.**
- **The process is sound: the answer is always correct.**

Conjunctive Normal Form :

To convert a formula into a CNF.

- Open up the implications($p \rightarrow q$) to get ORs.
- Get rid of double negations.
- Convert :

$A \vee (B \wedge C)$ to $(A \vee B) \wedge (A \vee C)$ //distributivity

Example:

Convert $A \rightarrow (B \wedge C)$ to CNF.

$A \rightarrow (B \wedge C) \equiv \neg A \vee (B \wedge C)$

$\equiv (\neg A \vee B) \wedge (\neg A \vee C)$

Note:

- $(y \vee \neg z) \wedge (\neg y) \wedge (y \vee z)$ is a CNF
- $(x \vee \neg y \vee z)$ is a CNF. So is $(x \wedge \neg y \wedge z)$.
- $(x \vee \neg y \wedge z)$ is not a CNF

PES UNIVERSITY ONLINE

Answer which of the following is/are in Conjunctive Normal Form :

1. $(x \vee \neg z \vee y) \wedge (\neg x \vee \neg y) \wedge (\neg y)$
2. $(x \wedge z) \vee (\neg y \wedge z \wedge x) \vee (\neg x \wedge z)$
3. $(x \wedge \neg(y \vee z) \vee u)$

PES UNIVERSITY ONLINE

Modus Ponens and Modus Tollens (as a special case of the Resolution rule)

Prove the following :

1) Modus Ponens

$p \rightarrow q$

p

$\therefore q$

2) Modus Tollens

$p \rightarrow q$

$\neg q$

$\therefore \neg p$

Modus Ponens and Modus Tollens (as a special case of the Resolution rule)

Prove the following :

1) Modus Ponens

$p \rightarrow q$

p

$\therefore q$

Solution 1):

$\neg p \vee q$

p

$\therefore q$

2) Modus Tollens

$p \rightarrow q$

$\neg q$

$\therefore \neg p$

Solution 2):

$\neg p \vee q$

$\neg q$

$\therefore \neg p$

For each of the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Q	Premises	Conclusion	In terms of implication
1	P v Q P → R Q → R	R	((P v Q) ∧ (P → R) ∧ (Q → R)) → R
2	P → Q R → S	(P v R) → (Q v S)	((P → Q) ∧ (R → S)) → ((P v R) → (Q v S))

PES UNIVERSITY ONLINE

For each of the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Q	Premises	Conclusion	Solution
1	P v Q P → R Q → R	R	Step 1 : Translate to CNF: i) P v Q ii) ¬P v R iii) ¬Q v R Step 2 : Resolve away the variable P using i) and ii) we get iv) Q v R Step 3: Resolve away variable Q using iii) and iv) we get R, hence proved

For each of the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Q	Premises	Conclusion
2	P → Q R → S	(P v R) → (Q v S)

Step 1 : Translate to CNF:

i) ¬P v Q

ii) ¬R v S

Step 2 : Negate the conclusion

= ¬(¬(P v R) v (Q v S))

= (P v R) ∧ (¬Q ∧ ¬S)

Therefore, we have the following premises :

i) ¬P v Q

ii) ¬R v S

iii) P v R

iv) ¬Q

iv) ¬S

For each of the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Step 3:
Resolve away Q using i) and iv)
i) ¬P v Q
ii) ¬R v S
iii) P v R
iv) ¬Q
v) ¬S
we get :
ii) ¬R v S
iii) P v R
iv) ¬S
vi) ¬P

Step 4:
Resolve away S using ii) and v)
ii) ¬R v S
iii) P v R
v) ¬S
vi) ¬P
we get :
iii) P v R
vi) ¬P
vii) ¬R

Step 5:
Resolve away P using iii) and vi)
iii) P v R
vi) ¬P
vii) ¬R
we get :
vii) ¬R
viii) R

Step 6: **Resolving R we get False.**

For each of the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Q	Premises	Conclusion	In terms of implication
3	P ¬P	Z	(P ∧ ¬P) → Z
	Example that shows that logical systems are brittle(fragile/breakable)		
4	A V ¬B ¬C V B ¬A C V B V D	D	((A V ¬B) ∧ (¬C V B) ∧ (¬A) ∧ (C V B V D)) → D

For each of the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Q	Premises	Conclusion	Solution
3	P ¬P	Z	(P ∧ ¬P) = False False → Z is True.

PES UNIVERSITY ONLINE

For each of the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Q4 Solution
i) A V ¬B
ii) ¬C V B
iii) ¬A
iv) C V B V D
v) ¬D

→

Resolve C
i) A V ¬B
ii) ¬C V B
iii) ¬A
iv) C V B V D
v) ¬D

→

Resolve B
i) A V ¬B
iii) ¬A
iv) B V D
v) ¬D

→

Resolve A
iii) ¬A
iv) A V D
v) ¬D

→

Resolve A
iv) D
v) ¬D

→ **false**

Homework :

For the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Premises	Conclusion
$(P \rightarrow Q) \rightarrow Q$ $(P \rightarrow P) \rightarrow R$ $(R \rightarrow S) \rightarrow \neg(S \rightarrow Q)$	R

Homework :

For the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Given Premises Prove R	Solving i) (P → Q) → Q
i) (P → Q) → Q ii) (P → P) → R iii) (R → S) → ¬(S → Q)	(¬P v Q) → Q ¬(¬P v Q) v Q (P ∧ ¬Q) v Q (P v Q) ∧ (¬Q v Q) (P v Q) ∧ T P v Q

Homework :

For the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Given Premises Prove R	Solving ii) (P → P) → R
i) (P → Q) → Q ii) (P → P) → R iii) (R → S) → ¬(S → Q)	(¬P v P) → R ¬(¬P v P) v R ¬T v R F v R R

PES UNIVERSITY ONLINE

Homework :

For the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Given Premises Prove R	Solving iii) $(R \rightarrow S) \rightarrow \neg(S \rightarrow Q)$
i) $(P \rightarrow Q) \rightarrow Q$ ii) $(P \rightarrow P) \rightarrow R$ iii) $(R \rightarrow S) \rightarrow \neg(S \rightarrow Q)$	$(\neg R \vee S) \rightarrow \neg(\neg S \vee Q)$ $(\neg R \vee S) \rightarrow (S \wedge \neg Q)$ $\neg(\neg R \vee S) \vee (S \wedge \neg Q)$ $(R \wedge \neg S) \vee (S \wedge \neg Q)$ $(R \vee S) \wedge (R \vee \neg Q) \wedge (\neg S \vee S) \wedge (\neg S \vee \neg Q)$ $(R \vee S) \wedge (R \vee \neg Q) \wedge T \wedge (\neg S \vee \neg Q)$ $(R \vee S) \wedge (R \vee \neg Q) \wedge (\neg S \vee \neg Q)$

Homework :

For the given question use Resolution Refutation to prove that the conclusion logically entails from the premises :

Given Premises Prove R	We have the following premises
i) $(P \rightarrow Q) \rightarrow Q$ ii) $(P \rightarrow P) \rightarrow R$ iii) $(R \rightarrow S) \rightarrow \neg(S \rightarrow Q)$	i) P v Q ii) R iii) (R v S) iv) (R v ¬Q) v) (¬S v ¬Q) Append the negated conclusion we get vi) ¬R We get false by resolving R using ii) and vi) Hence we can say the conclusion follows logically from the premises.

Homework :

What are the results of applying Propositional Resolution to the following pairs of clauses.

(a) {p V q V ¬r} and {r V s}

(b) {p V q V r} and {r V ¬s V ¬t}

(c) {¬p V q V r} and {p V ¬q V ¬r}

Homework :

What are the results of applying Propositional Resolution to the following pairs of clauses.

Q	Given clauses	Result of applying Propositional Resolution
1	p V q V ¬r r V s	p V q V s (resolve away r)
2	p V q V r r V ¬s V ¬t	p V q V r V¬s V ¬t (nothing can be resolved)
3	¬p V q V r p V ¬q V ¬r	False (resolve away p, q and r)

Automata Formal Languages & Logic

Wumpus World

- An application of Propositional logic in the area of Artificial Intelligence

Preet Kanwal

Department of Computer Science & Engineering

Automata Formal Languages and Logic

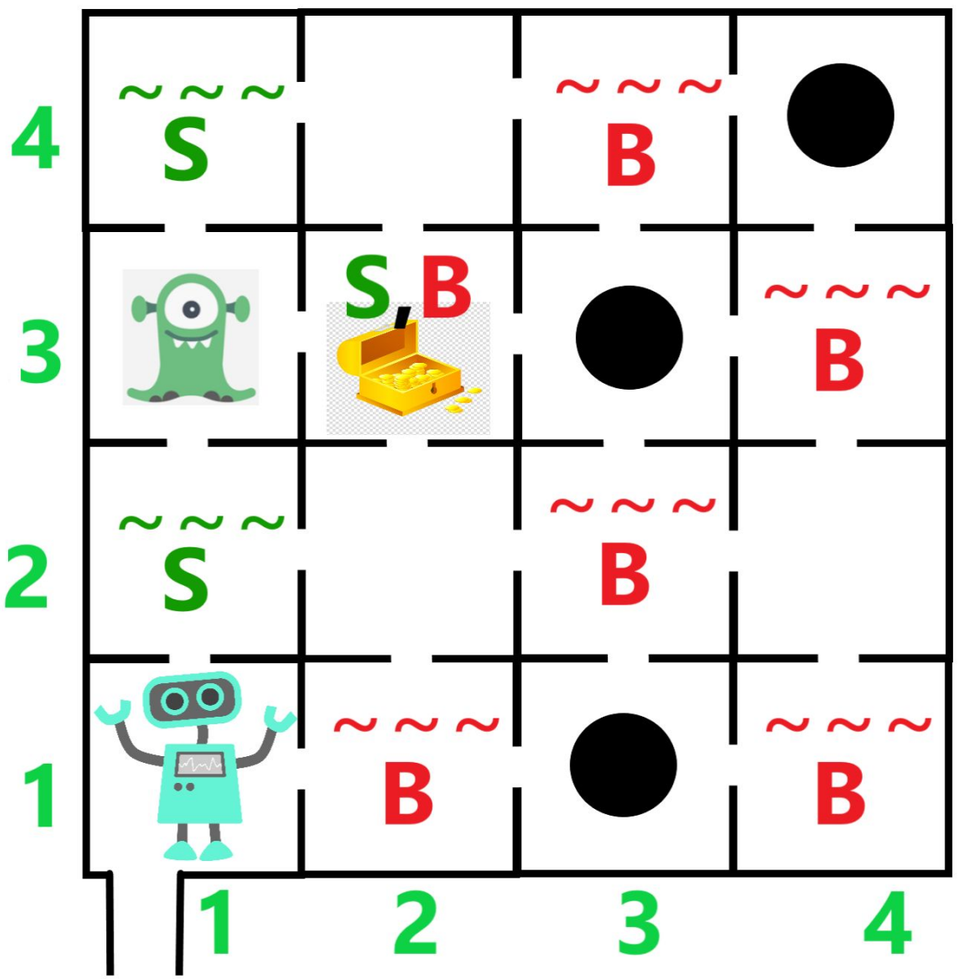
Unit 5 - The Wumpus World

Let us see What the game is about :

[wumpus world simulator]

AI is about

"Making computational models of human behavior that behave intelligently"



Knowledge-based Agent :

It is a Software that gathers information about an environment and takes actions based on that(Basically a **Player** in a game), It could be :

- a robot
- a web shopping program
- a traffic control system…

Knowledge-based agents are composed of two main parts:

- Knowledge-base and
- Inference system.

Knowledge-base:

Knowledge-base is a central component of a knowledge-based agent, it is also known as KB. It is a collection of sentences (Stores facts about the world)

These sentences are expressed in a knowledge representation language/ Declaration language for example : Propositional Logic or First-Order Predicate Logic.

Knowledge-base is required for updating knowledge for an agent to learn with experiences and take action as per the knowledge.

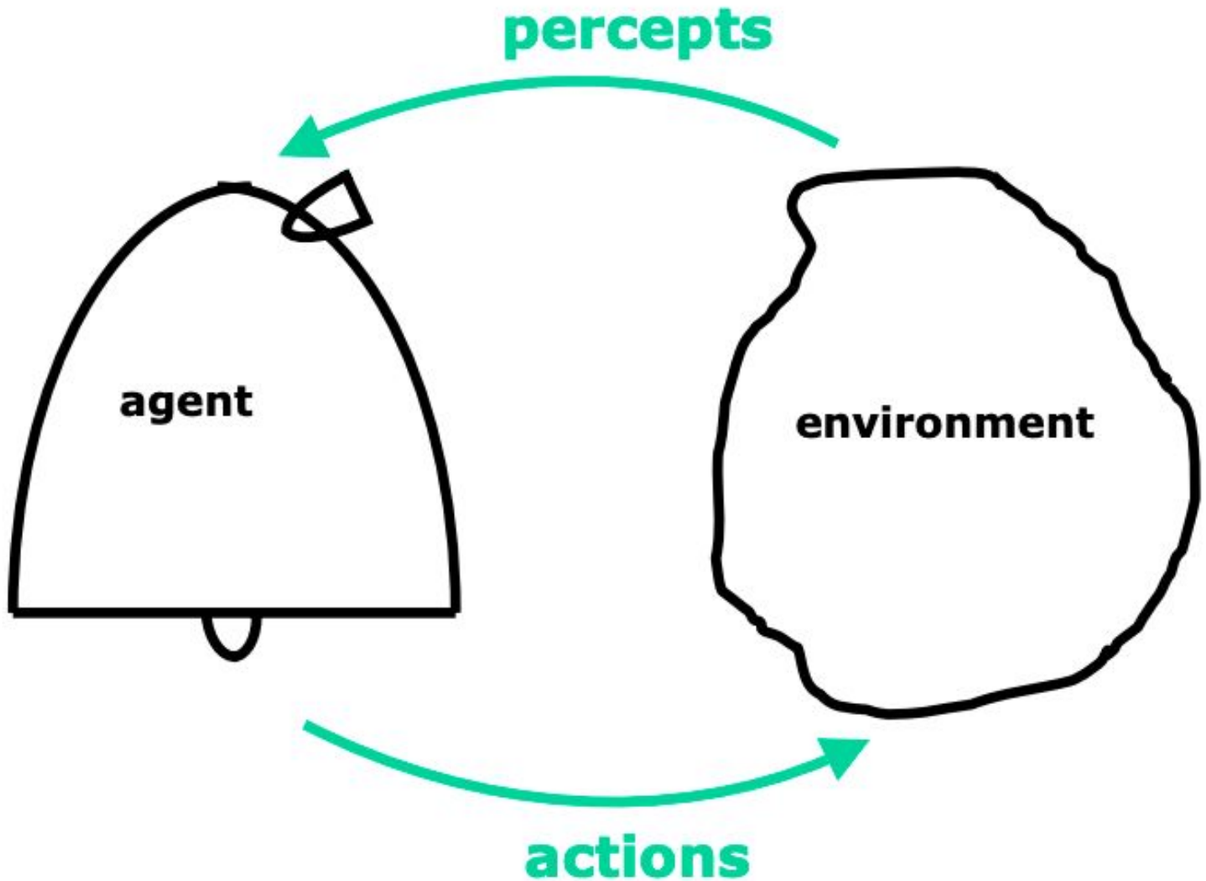
Inference System:

- Inference means deriving new sentences from old.
- Inference system allows us to add a new sentence to the knowledge base.
- A sentence is a proposition about the world.
- Inference system applies logical rules to the KB to deduce new information (logical entailment).
- Inference system generates new facts so that an agent can update the KB.

Our Objective :

- We will see how a knowledge-based agent can represent the world in which it operates (KB) and deduce what actions it should take.
- We will use Propositional Logic as representational language.

The Agent and the Environment



In Wumpus World :

Agent - Player

Environment - Wumpus World Model

Actions - Moving left, right, up or down

Percepts - Knowledge about the environment received via the sensors for example: [Stench, Breeze, Glitter, Bump, Scream]

- The image specifies an agent(player) and the world(environment) it lives in.
- The agent is going to take actions that affect the state of the environment
- The agent receive percepts that provide information about what's happening in the environment. Agent updates its knowledge after observations
- The agent changes the state of the environment when it perform an action, it then perceives some new information about the state of the environment. This happens in a loop until the game is over!

Three Operations performed by the Knowledge-based Agent:

- TELLs the knowledge base what it perceives from the environment. (Input)
- ASKs the knowledge base what action it should perform.
- PERFORMs the selected action. (Output)

The agent maintains the knowledge base (KB), and it initially has some background knowledge of the real world.

How to define the Environment in which the agent lives: (World Model)

- Action Space (Actuators)
- Percept Space (Sensors)
- Environment

How to define the Environment in which the agent lives: (World Model)

- **Action Space (Actuators) : Actions that the agent can take for example:**
  - **Move Forward**
  - **Turn Left or Right by 90 degrees**
  - **Grab (to pick the gold)**
  - **Shoot (Shoot in Straight line)**

How to define the Environment in which the agent lives: (World Model)

- **Percept Space :** provides information about the environment with the help of the sensors:
  - If a cell contains the Wumpus, Agent perceives **Stench** in adjacent cells (not diagonally)
  - If a cell contains a Pit, Agent perceives **Breeze** in adjacent cells (not diagonally)
  - If a cell contains Gold, Agent perceives **Glitter** in the same cell.
  - When Agent walks into the wall, it perceives a **Bump**.
  - When the Agent kills the Wumpus, a woeful **Scream** can be perceived anywhere in the cave.

How to define the Environment in which the agent lives: (World Model)

- **Environment :**
  - 4 X 4 Grid of rooms
  - Agent always starts in the [1,1] cell facing to the right
  - Locations of Gold and Wumpus are chosen randomly with a uniform distribution (except [1,1]).
  - Any cell can be Pit (except [1,1]) with a Probability 0.2

The game ends either when the agent dies or when the agent exits the game successfully.

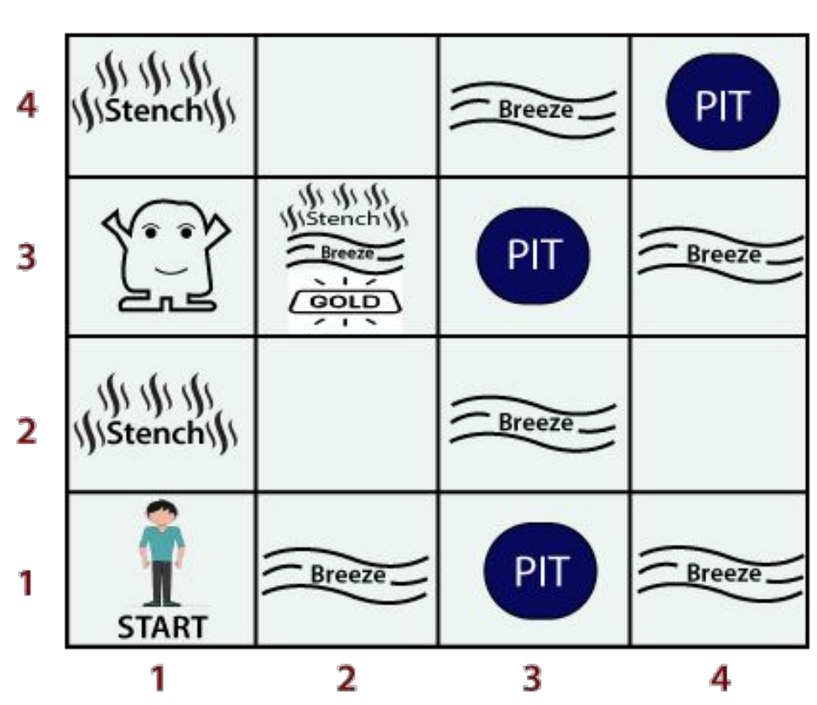
Points are awarded each time the gold is obtained.

You could also introduce points for other actions.

The Wumpus World Knowledge Base :

Given the model, we define a set of propositions for each cell [x,y]

1. P_{xy} is true if there is a pit in [x,y]. There are 16 propositions in total since it is a 4 X 4 grid.
2. W_{xy} is true if there is a Wumpus in [x,y]. There are 16 propositions in total since it is a 4 X 4 grid.
3. B_{xy} is true if agent perceives a Breeze in [x,y]. There are 16 propositions in total since it is a 4 X 4 grid.
4. S_{xy} is trueif agent perceives a Stench in [x,y]. There are 16 propositions in total since it is a 4 X 4 grid.

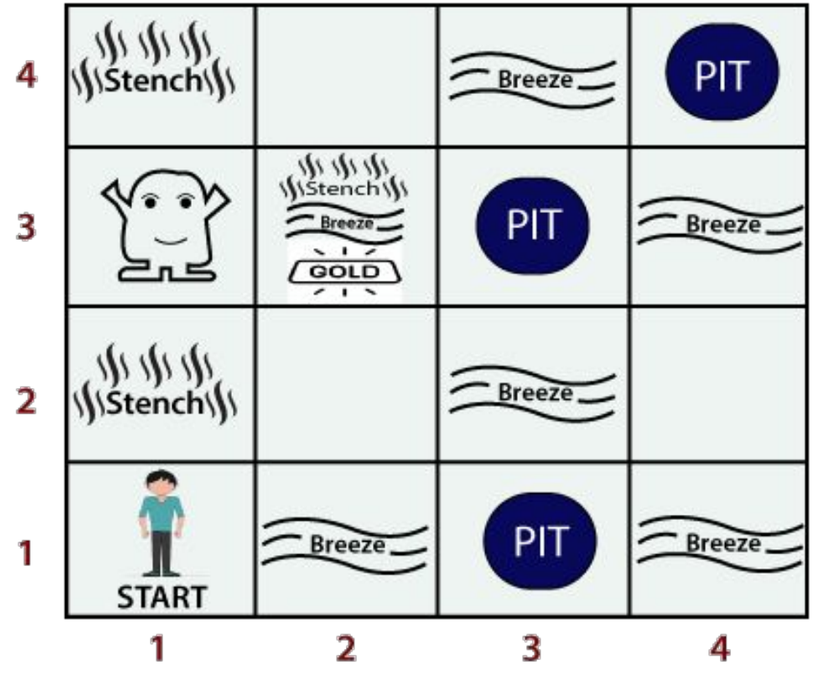


The Wumpus World Knowledge Base :

Given the model, we define a set of propositions for each cell [x,y]

1. P_{xy} is true if there is a pit in [x,y]. There are 16 propositions in total since it is a 4 X 4 grid. That is,

Proposition	Truth Value
P_{11} - There is a pit in [1,1]	False
P_{12} - There is a pit in [1,2]	False
…	…
…	…
P_{44} - There is a pit in [4,4]	True

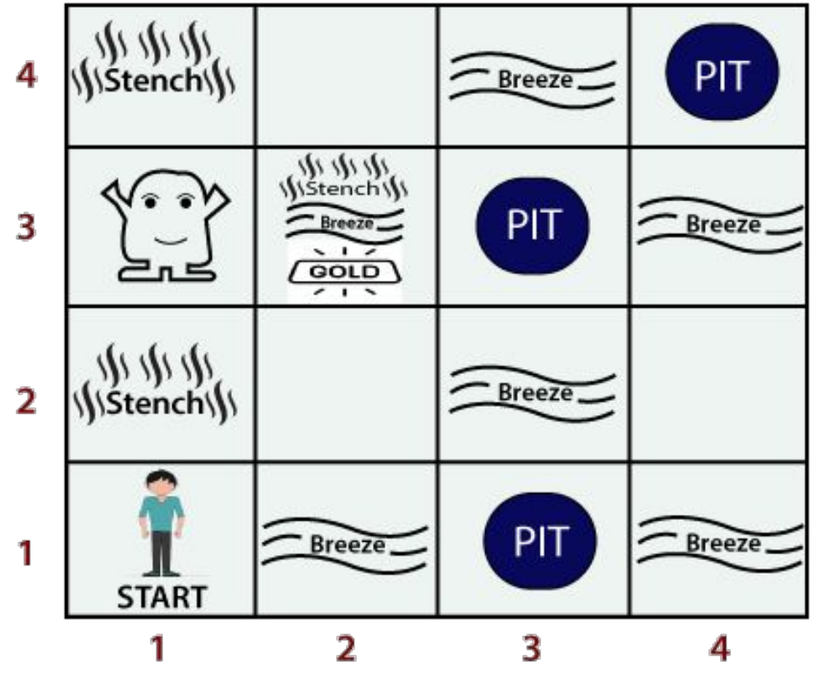


The Wumpus World Knowledge Base :

Given the model, we define a set of propositions for each cell [x,y]

2. W_{xy} is true if there is a Wumpus in [x,y]. There are 16 propositions in total since it is a 4 X 4 grid. That is,

Proposition	Truth Value
W_{11} - There is a Wumpus in [1,1]	False
W_{12} - There is a Wumpus in [1,2]	False
W_{13} - There is a Wumpus in [1,3]	True
...	...
W_{44} - There is a Wumpus in [4,4]	False

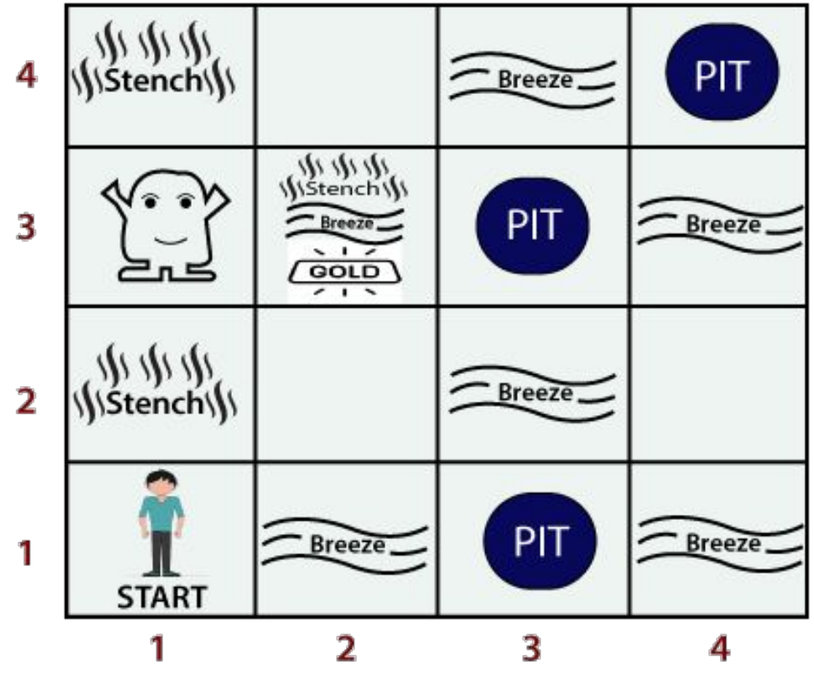


The Wumpus World Knowledge Base :

Given the model, we define a set of propositions for each cell [x,y]

3. B_{xy} is true if agent perceives a Breeze in [x,y]. There are 16 propositions in total since it is a 4 X 4 grid. That is,

Proposition	Truth Value
B_{11} - There is a Breeze in [1,1]	False
B_{12} - There is a Breeze in [1,2]	False
B_{21} - There is a Breeze in [2,1]	True
...	...
B_{44} - There is a Breeze in [4,4]	False

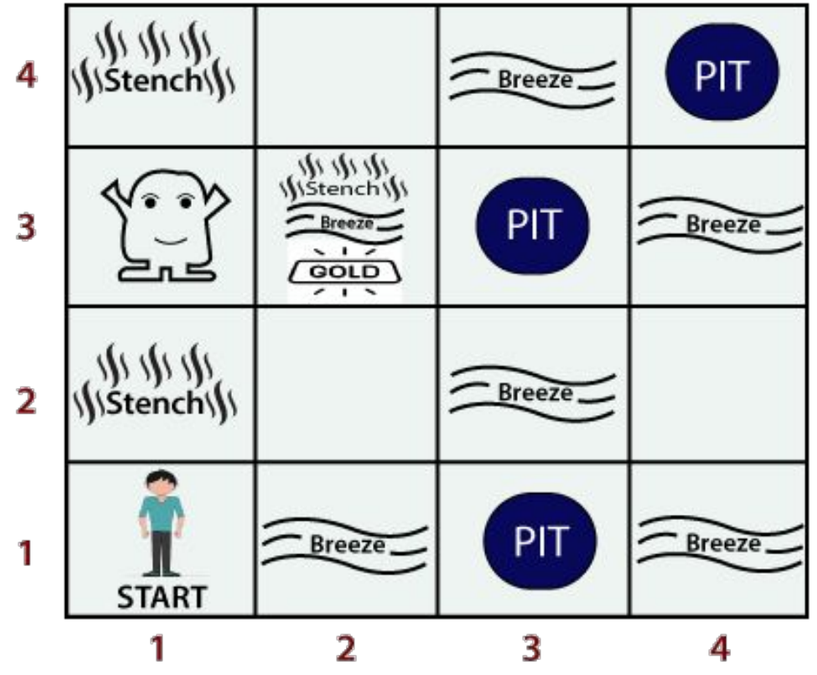


The Wumpus World Knowledge Base :

Given the model, we define a set of propositions for each cell [x,y]

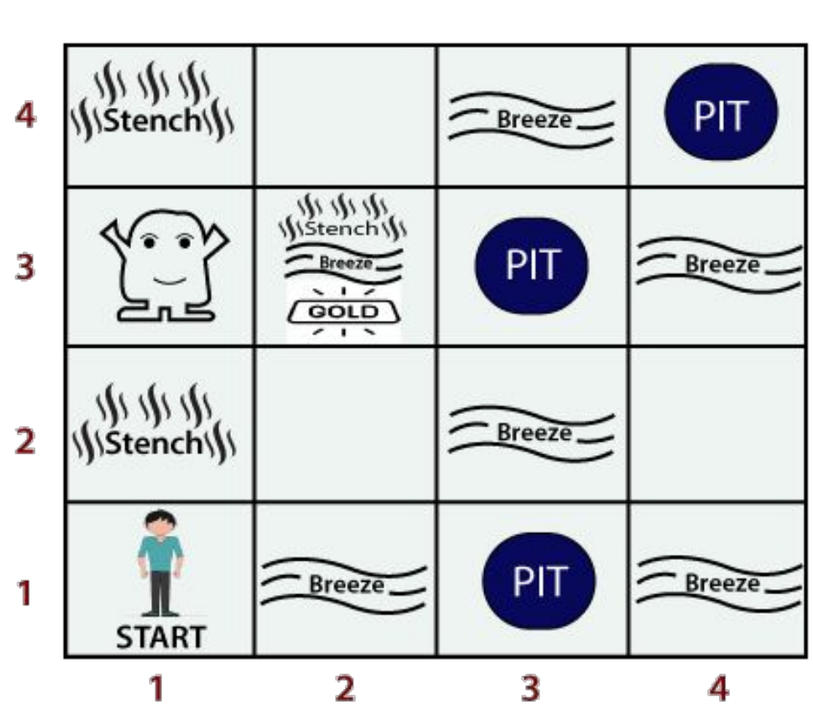
4. S_{xy} is true if agent perceives a Stench in [x,y]. There are 16 propositions in total since it is a 4 X 4 grid. That is,

Proposition	Truth Value
S_{11} - There is a Stench in [1,1]	False
S_{12} - There is a Stench in [1,2]	True
S_{21} - There is a Stench in [2,1]	False
...	...
S_{44} - There is a Stench in [4,4]	False



Some Aspects of The Wumpus World

Proposition	Meaning
General Aspects - Immutable (Applies to all models) - facts known	
$R_1 : \neg P_{11}$	There is no pit in [1,1]
$R_2 : B_{11} \leftrightarrow (P_{12} V P_{21})$	There is a breeze in [1,1] iff there is a pit in adjoining cell i.e. [1,2] or [2,1]
$R_3 : B_{21} \leftrightarrow (P_{11} V P_{31} V P_{22})$	There is a breeze in [1,1] iff there is a pit in adjoining cells i.e. [1,1] or [3,1] or [2,2]
Aspects related to a Specific Model like the one shown in the figure - facts gained	
$R_4 : \neg B_{11}$	There is no breeze in [1,1]
$R_5 : B_{21}$	There is breeze in [2,1]



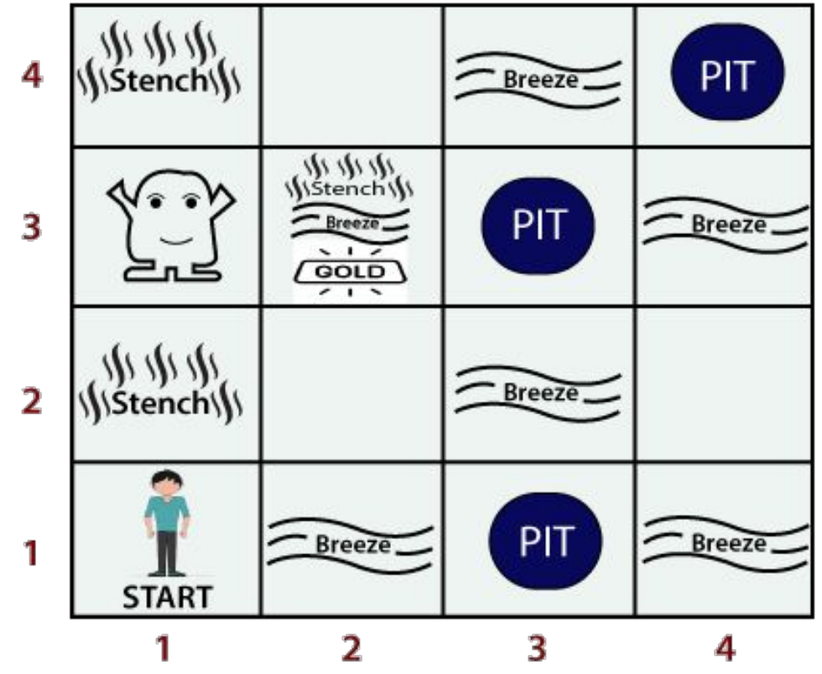
The Wumpus World Knowledge Base Example:

Assume the Knowledge base (KB) is made up of the rules R_1 to R_5

The propositions involves in these rules are :
B_{11}, B_{21}, P_{11},P_{12},P_{21},P_{22}, P_{31}

Can we entail any logical result from the KB?? i.e.
What is α??
KB ⊨ α

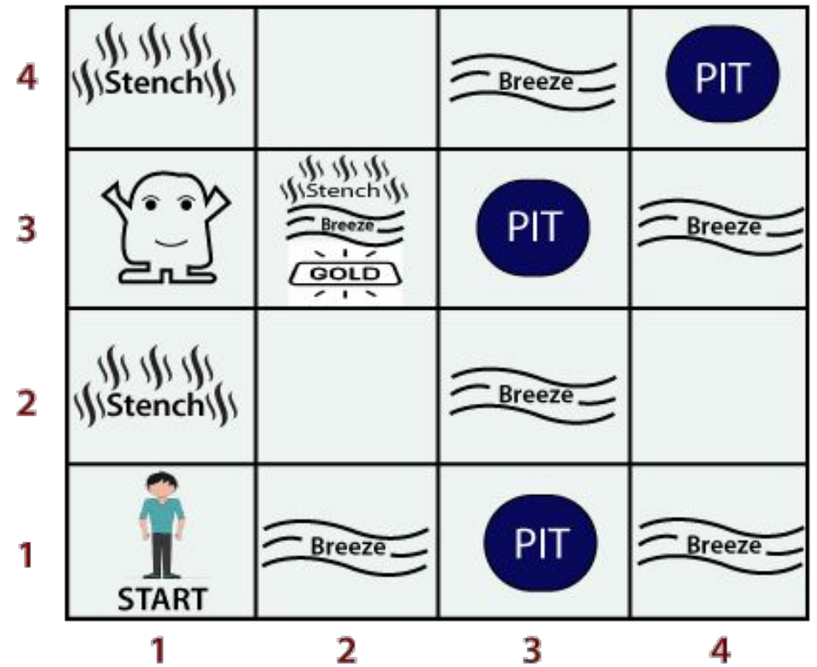
α must be true in every model in which KB is true



PES UNIVERSITY ONLINE

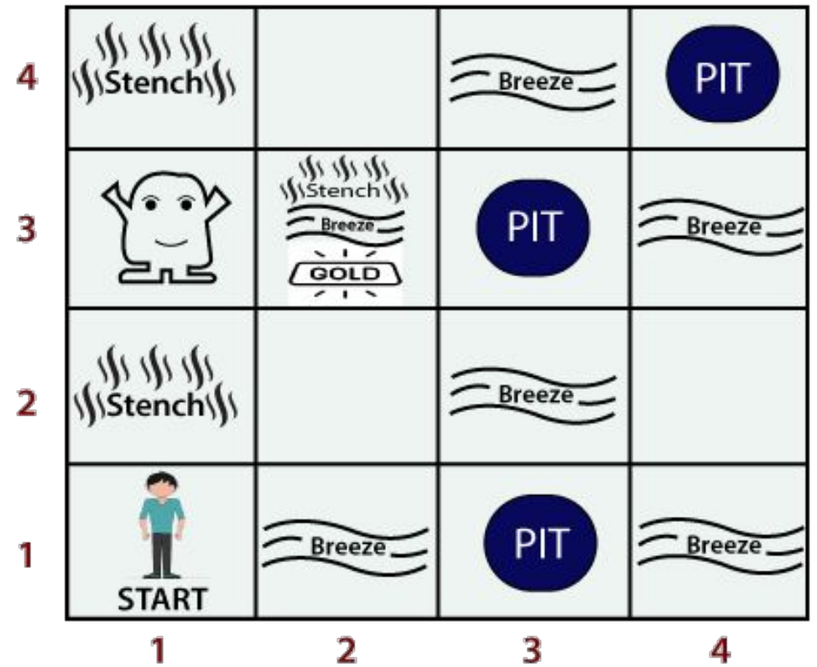
We can create a Truth table for the KB involving these 7 propositions. We will have 2^7 entries in the truth table.

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*



Each row represents a Model.
Only in the highlighted models(rows), Our KB is true.
Whenever our KB is true, we can infer other logical conclusions.

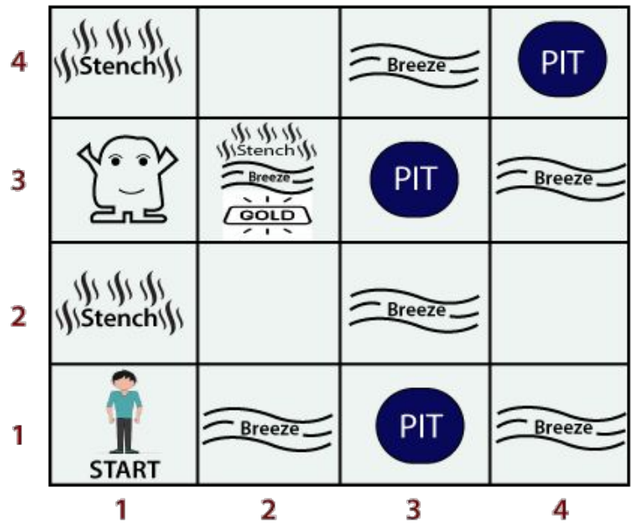
$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*



Here, we can entail the following from the KB that :

- There is no Breeze in [1,1]
- There is a Breeze in [2, 1]
- There is no Pit in [1,1], [1,2], [2,1]

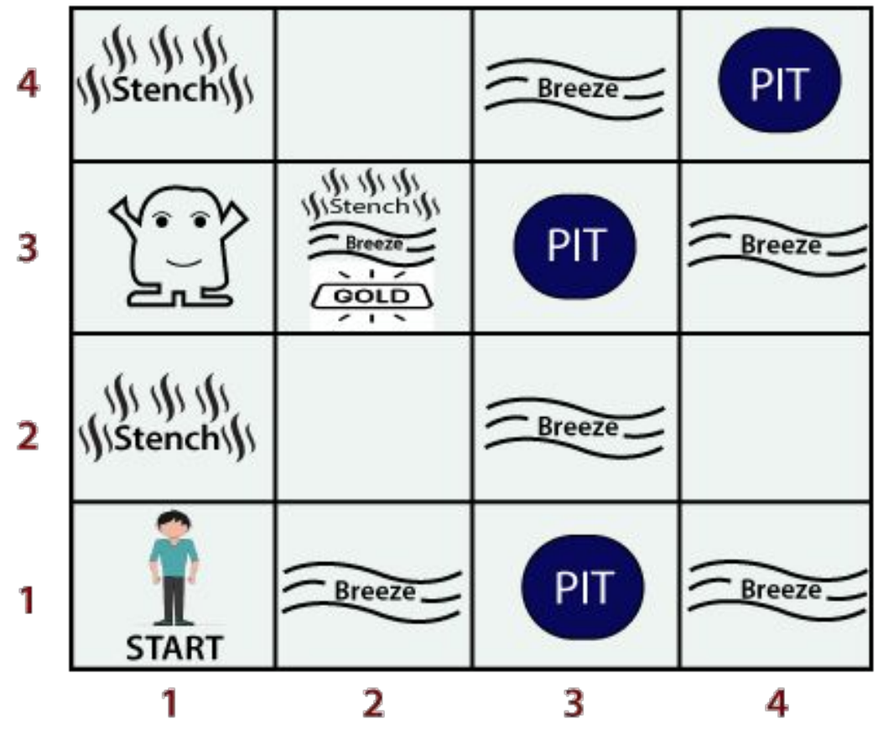
$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*



Here, we cannot infer the following from the KB that :

- Whether or not there is Pit in [2,2] , [3,1]

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*



Hence we can entail the following from the knowledge base (KB)

$KB \models \neg B_{11}$

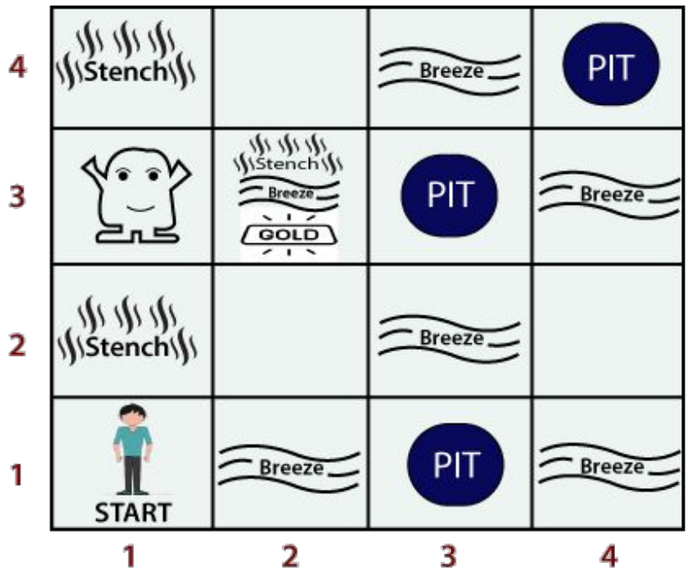
$KB \models B_{21}$

$KB \models \neg P_{11}$

$KB \models \neg P_{12}$

$KB \models \neg P_{21}$

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*



PES UNIVERSITY ONLINE

Proof by Resolution that the KB $\models \neg P_{12}$

Premises	Conclusion
1. $\neg P_{11}$ 2. $B_{11} \leftrightarrow (P_{12} V P_{21})$ 3. $B_{21} \leftrightarrow (P_{11} V P_{31} V P_{22})$ 4. $\neg B_{11}$ 5. B_{21}	$\neg P_{12}$

Proof by Resolution that the KB $\models \neg P_{12}$

Premises	Conclusion
1. $\neg P_{11}$ -ignore (not related) 2. $B_{11} \leftrightarrow (P_{12} V P_{21})$ 3. $B_{21} \leftrightarrow (P_{11} V P_{31} V P_{22})$ - ignore (no P_{12} term) 4. $\neg B_{11}$ 5. B_{21}- ignore (not related)	$\neg P_{12}$

Proof by Resolution that the KB $\models \neg P_{12}$

Premises	Conclusion	Solution
1. $B_{11} \leftrightarrow (P_{12} \vee P_{21})$ 2. $\neg B_{11}$ Therefore we have: i) $\neg B_{11} \vee P_{12} \vee P_{21}$ ii) $\neg P_{12} \vee B_{11}$ iii) $\neg P_{21} \vee B_{11}$ iv) $\neg B_{11}$ v) P_{12}	$\neg P_{12}$	Converting $(B_{11} \rightarrow (P_{12} \vee P_{21})) \wedge ((P_{12} \vee P_{21}) \rightarrow B_{11})$ to CNF $= (\neg B_{11} \vee (P_{12} \vee P_{21})) \wedge (\neg (P_{12} \vee P_{21}) \vee B_{11})$ $= (\neg B_{11} \vee P_{12} \vee P_{21}) \wedge ((\neg P_{12} \wedge \neg P_{21}) \vee B_{11})$ $= (\neg B_{11} \vee P_{12} \vee P_{21}) \wedge ((\neg P_{12} \vee B_{11}) \wedge (\neg P_{21} \vee B_{11}))$ $= (\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg P_{12} \vee B_{11}) \wedge (\neg P_{21} \vee B_{11})$ **Solution:** Resolve away B_{11} using ii) and iv) result is P_{12}. Using the above result (P_{12})and v) we get **FALSE**. Hence Conclusion logically follows from the premises.

Satisfiability and validity:

- In mathematical logic, satisfiability and validity are elementary concepts of semantics.
- A sentence is satisfiable if it is possible to find an interpretation (model) that makes the formula true.
- In our example, KB is satisfiable because 3 models(rows) make the KB true. Hence, KB is satisfiable.
- A sentence is valid if all interpretations make the formula true. That is KB $\models \alpha$ is valid iff KB $\rightarrow \alpha$ is a tautology.

https://en.wikipedia.org/wiki/Satisfiability#:~:text=In%20mathematical%20logic%2C%20satisfiability%20and,that%20makes%20the%20formula%20true.&text=The%20question%20whether%20a%20sentence,satisfiable%20is%20a%20decidable%20problem.

Given the following Knowledge Base, how many models can you frame??

R_1: $\neg P_{1,1}$

R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R_4: $\neg B_{1,1}$

R_5: $B_{2,1}$

R_6: $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

R_7: $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

R_8: $(\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$

R_9: $\neg(P_{1,2} \vee P_{2,1})$

R_{10}: $\neg P_{1,2} \wedge \neg P_{2,1}$

R_{11}: $\neg B_{1,2}$

R_{12}: $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$

R_{13}: $\neg P_{2,2}$

R_{14}: $\neg P_{1,3}$

R_{15}: $P_{1,1} \vee P_{2,2} \vee P_{3,1}$

R_{16}: $P_{1,1} \vee P_{3,1}$

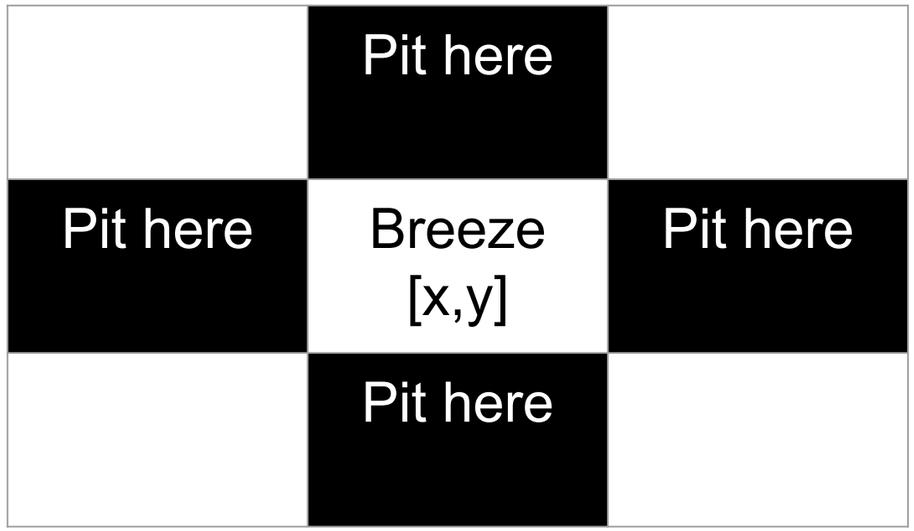
R_{17}: $P_{3,1}$

Limitations of Propositional Logic:

- Can be used to model simple environments/models
- Does not capture Pattern

For example : We will have to write down a separate Rule for every cell about Breezes and Pits, although all the rules would convey the same information which is :

The Breeze exists in cell [x,y] iff there is a Pit in adjoining cells i.e. it could in either of the four locations:
[x, y+1] (up) or
[x, y-1] (down) or
[x-1, y] (left) or
[x+1,y] (right)



We overcome this limitation with the help of First-Order Predicate Logic (FOPL).

Limitations of Propositional Logic:

How can we say that adding 1 to an even number produces an odd number?

- We need infinite symbols and infinite rules.
- – A symbol O1 for “1 is odd”, a symbol E2 for “2 is even”, …

What do these limitations buy us?

– Simple syntax: just symbols and connectives.

– Inference algorithms (like TT-Entails) that are horribly slow (exponential time), but at least terminate in finite time.

Automata Formal Languages & Logic

Unit 5 - First-Order Predicate Logic

Preet Kanwal

Department of Computer Science & Engineering

Propositional Logic (PL)	First Order Predicate Logic (FOPL)
Declarative Language	Declarative Language; An extension to PL
Used to represent knowledge about simple environments	Used to represent knowledge about complex environments
Lacks Expressive power to concisely describe an environment Example : We are forced to write a separate rule about breezes and pits for each square	Can represent rules generally. Example : “Squares neighboring the wumpus are smelly”
Assumes the world contains facts	Assumes the world contains Objects, Relations and Functions.
facts either hold or do not hold	relations between objects hold or do not hold

Basic Elements which describe a Model in FOPL consists of:

- **Set of Objects (called Domain of the Model).**
- **Relations (specify how objects are related, whether that relationship holds or not)**
    - **Input : Object(s) or Function(s)**
    - **Output : True or False**
- **Functions (A special case of a relation, where given object related to exactly one object). Must be a Total Function (i.e. It must be defined/provide output for all inputs of the right type)**
    - **Input : Object(s)**
    - **Output : Object**

Basic Elements which describe a Model in FOPL consists of:

- **Set of Objects (called Domain of the Model).**
  Example : A, B, people, numbers, colors, wars, theories, squares, pits, wumpus, John, Mary, house, backpack, Arlington, Texas…

Basic Elements which describe a Model in FOPL consists of:

- **Relations** (specify how objects are related, whether that relationship holds or not)
  - **Input** : Object(s) or Function(s)
  - **Output** : True or False

Example : Siblings(John, Mary), >(100, 5), Red(laptop551),
Team(John, Mary, Sue, Jim)

Basic Elements which describe a Model in FOPL consists of:

- **Functions (A special case of a relation, where given object related to exactly one object). Must be a Total Function (i.e. It must be defined/provide output for all inputs of the right type)**
    - **Input : Object(s)**
    - **Output : Object**

Example : Father(Raj),
Capital(Texas),
Mother(John),
Sum(25, 12),
sqrt(20),..

Read the functions as : Father of Raj, Capital of Texas, Mother of John, Sum of 25 and 12, Sqrt of of

Note : Both Relations and Functions can have any arity(i.e. any no. of arguments)

Should "sibling" be a relation or a function?

– Relation: siblings(John, Mary)

– Function: sibling(John) returns Mary.

- "Sibling" should be a relation, because someone can have many (or no) siblings. A function must and can only return one value.

Note : Both Relations and Functions can have any arity(i.e. any no. of arguments)

Should "mother" be a relation or a function?

– Relation: mother(Liza, John)

– Function: mother(John) returns Liza.

"Mother" can be either a relation or a function.

– A person (or animal) has only one mother.

First-order Predicate Logic as a Declarative language has:

- A Syntax
- Semantics

Basic Elements of First-Order logic Syntax:

Constant Symbols (Objects)	1, 2, A John, Mumbai, Person, Chair
Predicate Symbols (Relations)	Brother, Father, >, ….
Function Symbols	sqrt, multiply
Variables	x,y,z,a,b...
Term	Constant Symbol or Function Symbol or Variable
Equality	=
Connectives	¬ , ∨ , ∧, →, ↔
Quantifier	∀, E

https://www.javatpoint.com/first-order-logic-in-artificial-intelligence

Syntax of First-Order logic:

- Model in First-Order Logic must be specified as per the Syntax.
- The syntax of FOL determines which collection of symbols is a logical expression in first-order logic.

Basic Syntactic elements of first-order logic are symbols, which are:

- Constant Symbols (Objects)
- Predicate Symbols (Relations)
- Function Symbols (Functions)

Model must provide the Interpretation, that is :

- Which objects are referred by the Constant Symbols (as a constant symbol can map to any object).
- Which relations and functions are referred by Predicate and Function Symbols respectively.

The model must provide all the information required to determine the truth value of any given sentence.

Syntax of First-Order logic (Continued …) :

Quantifiers in First-order logic:

What makes first-order logic powerful is that it allows us to make general assertions using quantifiers. 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:

- **Universal Quantifier, (for all, everyone, everything) : If x is a variable, then ∀x is read as:**
  - **For all x**
  - **For each x**
  - **For every x.**
- **Existential quantifier, (for some, at least one). If x is a variable, then ∃x is read as:**
  - **There exists a 'x.'**
  - **For some 'x.'**
  - **For at least one 'x.'**

Syntax of First-Order logic (Continued …) :

Points to remember:

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

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 $\forall y \exists x$.

Let the domain be all the animals.

D(x): x is a dog

M(x): x is a mammal

C(x): x is cute

"All dogs are mammals" is equivalent to

$\forall x\,(D(x) \rightarrow M(x))$ or $\forall x\,(D(x) \wedge M(x))$?

"Some dogs are cute" is equivalent to

$\exists x\,(D(x) \rightarrow C(x))$ or $\exists x\,(D(x) \wedge C(x))$?

Explanation for: “All dogs are mammals”

What does $\forall x\,(D(x) \rightarrow M(x))$ mean?

$\forall x\,(D(x) \rightarrow M(x)) \equiv \forall x\,(\neg D(x) \vee M(x))$

Each animal is a non-dog or a mammal.

It means all dogs are mammals.

What does $\forall x\,(D(x) \wedge M(x))$ mean?

$\forall x\,(D(x) \wedge M(x)) \equiv (\forall x\,D(x)) \wedge (\forall x\,M(x))$

All animals are dogs and all animals are mammals.

It is not same as “All dogs are mammals”.

Explanation for: “Some dogs are Cute”

What does $\exists x (D(x) \rightarrow C(x))$ mean?

There exists an animal, if that animal is a Dog, then it is cute.

It means all dogs are cute.

It is not same as “Some dogs are Cute”

What does $\exists x (D(x) \wedge C(x))$ mean?

There exists some animal, which is a dog and is cute.

Please note the question says only few dogs are cute and not all.

Hence, there could exist a dog who is not cute.

Syntax of First-Order logic (Continued …) :

Free and Bound Variables:

The quantifiers interact with variables which appear in a suitable way. There are two types of variables in First-order logic which are given below:

Free Variable: A variable is said to be a free variable in a formula if it occurs outside the scope of the quantifier.

Example: ∀x ∃(y)[P (x, y, z)], where z is a free variable.

Bound Variable: A variable is said to be a bound variable in a formula if it occurs within the scope of the quantifier.

Example: ∀x [A (x) B(y)], here x and y are the bound variables.

Syntax of First-Order logic (Continued …) :

Variable :

- Can take values in a domain. (Domain is a set of objects)
- Can only be used together with quantifiers.

Example (Here x and y are variables) :

∀ x, y brothers(x, y) => siblings(x, y)

Meaning: For every x and y, if x is a brother of y then x is a sibling of y

∃ x 2*5 + x = 18

Meaning: There exists x such that result of 2*5 + x is equal to 18

Syntax of First-Order logic (Continued …) :

Term can be:

- Constant Symbols (Objects)
- Function Symbols (Functions which can take Term as argument)
- Variable

Predicate symbols can take Term as argument :

- Which objects are referred by the Constant Symbols (as a constant symbol can map to any object).
- Which relations and functions are referred by Predicate and Function Symbols respectively.

Term = Term:

- Equality symbol (=) is used to signify that two terms refer to the same object.

Syntax of First-Order logic (Continued …) :

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).**

In a Prolog program, this refers to a fact

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

elephant is bigger than horse => bigger(elephant, horse).

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

Note: Functions do not state facts and form no sentence.

Syntax of First-Order logic (Continued …) :

Complex Sentences:

Complex sentences are made by combining atomic sentences using connectives($\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$).

Example:

find(X, Z):- (bigger(X,Y), bigger(Y,Z)) ; bigger(X,Z).

[In prolog use must write a rule to specify complex sentence]

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} &\rightarrow (\textit{Sentence}) \mid [\textit{Sentence}] \\
&\mid \neg \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
&\mid \textit{Quantifier Variable}, \ldots \textit{Sentence} \\
\textit{Term} &\rightarrow \textit{Function}(\textit{Term}, \ldots) \\
&\mid \textit{Constant} \\
&\mid \textit{Variable} \\
\textit{Quantifier} &\rightarrow \forall \mid \exists \\
\textit{Constant} &\rightarrow A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} &\rightarrow a \mid x \mid s \mid \cdots \\
\textit{Predicate} &\rightarrow \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} &\rightarrow \textit{Mother} \mid \textit{LeftLeg} \mid \cdots \\
\text{OPERATOR PRECEDENCE} &: \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

Number of Possible Worlds/Models :

In propositional logic, suppose that you have 100 symbols. How many possible worlds do you have?

– 2^{100}

– One for each row of the truth table.

In first-order logic, how can we even count the number of possible worlds?

Number of possible worlds is counted in terms of:

- The number of objects
- Mapping of Constant Symbols to Objects (No. of predicates and its arity)

Number of possible models are unbounded. Checking entailment via Model Checking is not feasible as the number of combinations can become very large.

http://vlm1.uta.edu/~athitsos/courses/cse4308_fall2016/lectures/03b_first_order_logic.pdf

Number of Possible worlds in FOPL:

Suppose that we have:
– Five constants.
– No functions.
– One predicate, that takes one argument.

- **How many possible worlds can we have?**

- **For each possible argument of the predicate, we must specify if the predicate returns true or false.**

- **We have five possible arguments.**

- **In total, 2^5 possible worlds.**

Number of Possible worlds in FOPL:

Suppose that we have:
– Let the Five constants be V, W, X, Y, Z
– No functions.
– Let the predicate be P, that takes one argument(either of V, W, X, Y, Z) .

There will be 2^5 possible worlds/models (each row of the truth table is a model).

Model	P(V)	P(W)	P(X)	P(Y)	P(Z)
1	*false*	*false*	*false*	*false*	*false*
2	*true*	*false*	*false*	*false*	*false*
...	...	...	...	...	...
...	...	...	...	...	...
2^5 =32	*true*	*true*	*true*	*true*	*true*

Number of Possible worlds in FOPL:

Suppose that we have:
– Five constants.
– No functions.
– One predicate, that takes two arguments.

- **How many possible worlds can we have?**

- **For each possible combination of arguments of the predicate, we must specify if the predicate returns true or false.**

- **We have 25 possible combinations of arguments.**

- **In total, 2^{25} possible worlds.**

Number of Possible worlds in FOPL:

Suppose that we have:
– Let the Five constants be V, W, X, Y, Z
– No functions.
– Let the predicate be P, that takes two arguments(either of V, W, X, Y, Z).

- **We have 2^{25} possible worlds/models**

Model	P(V,V)	P(V,W)	...	...	P(Z,Z)
1	*false*	*false*	*false*	*false*	*false*
2	*true*	*false*	*false*	*false*	*false*
...	...	...	...	...	...
...	...	...	...	...	...
2^{25}	*true*	*true*	*true*	*true*	*true*

Number of Possible worlds in FOPL:

Suppose that we have:

– Five constants.

– No functions.

– Three predicates, that take two arguments.

– One predicate that takes one argument.

- **How many possible worlds can we have?**
- **For each possible combination of arguments of each predicate, we must specify if the predicate returns true or false.**
- **For two arguments, we have 25 combinations of arguments. We have three such predicates, so we must specify 75 values.**
- **For one argument, we have 5 combinations of arguments. We have one such predicate, so we must specify 5 values.**
- **In total, $2^{5+75} = 2^{80}$ possible worlds.**

Example:

For the wumpus world, to say that "pits cause breezes in adjacent squares" using propositional logic, we need 16 rules like:

$B_{12} \rightarrow (P_{11} \vee P_{22} \vee P_{13})$

In first-order logic, how can we say the same thing?

Objects : Pits, Squares

Relations : Breeze, adjacent

$\forall\ x_1, y_1$ **Breeze**$(x_1, y_1) \leftrightarrow (\ \exists\ x_2, y_2$ **Pit**(x_2, y_2) **AND Adjacent**$(x_1, y_1, x_2, y_2)\ )$

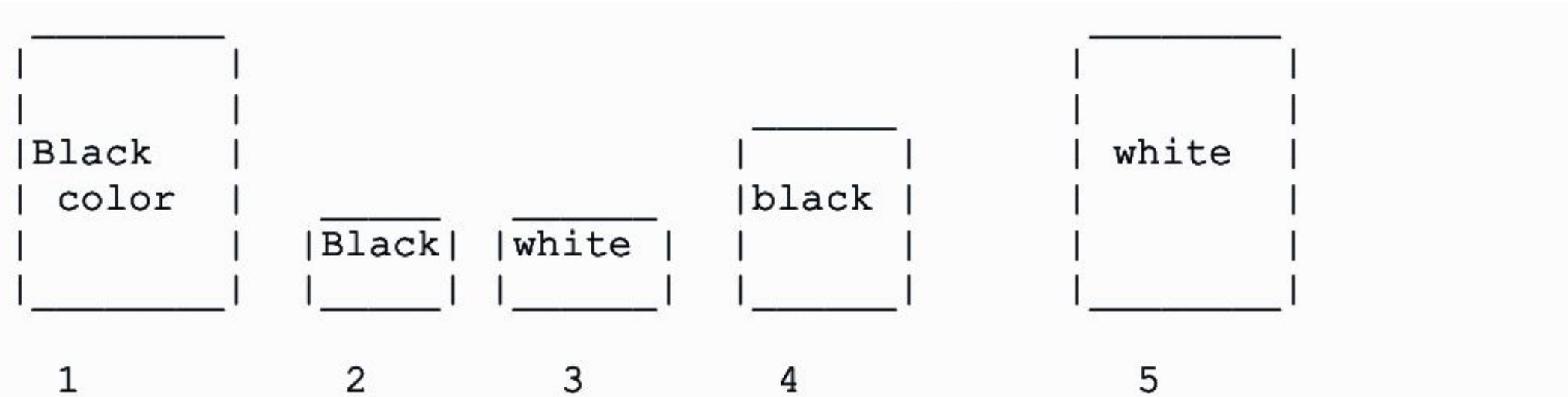
Knowledge Representation using FOPL

Knowledge Representation using FOPL :

We will look at examples from the following Domain:

- A puzzle - Box Solver
- Kinship (Family Relationships)
- Mathematical Sets

Example 1 : Let us look at an example called Box Solver. There are 5 boxes



Ann, Bill, Charlie, Don, Eric own one box each but we don't know which box.
Try to find each boxes' owner if you know that:
- Ann and Bill have boxes with the same color.
- Don and Eric have boxes with the same color.
- Charlie and Don have boxes with the same size.
- Eric's box is smaller than Bill's.

In the puzzle - Box Solver we have:

- Objects : Box
- A Unary Predicate : box
- A Ternary Predicate : box_details
- A Function : Owners

```
Ann, Bill, Charlie, Don, Eric own one box each but we don't know
which box.
Try to find each boxes' owner if you know that:
Ann and Bill have boxes with the same color.
Don and Eric have boxes with the same color.
Charlie and Don have boxes with the same size.
Eric's box is smaller than Bill's.
```

In the puzzle - FOPL representation:

$\exists A,B,C,D,E,c1,c2,c3,s1,s2,s3$

$(A \neq B \neq C \neq D \neq E) \wedge$

box_details(A,c1,s1) $\wedge$ box_details(B,c1,s2) $\wedge$ % Ann and Bill's box have same color

box_details(D,c2,s3) $\wedge$ box_details(E,c2,s4) $\wedge$ % Don and Eric's box have same color

box_details(C,c3,s3)$\wedge$ box_details(D,c2,s3) $\wedge$ % Charlie and Don's box have same size

box_details(E,c2,s4) $\wedge$ box_details(B,c1,s2) $\wedge$ % Eric's box is smaller than Bill's

$s4 < s2$

In order to solve the problem, you can:

1. Write a Prolog database with the characteristics of the boxes, associating a number to each box, for instance:
box_details(2, black,1) represents that the second box is of color black and size 1;

2. Write a predicate to compute the solution of the problem: owner(A, B, C, D, E) must be true if A is the box owned by Ann, B the one owned by Bill, and so on. . .

```
% First define numbers for each box.
box(1).
box(2).                                          %Unary Predicate
box(3).
box(4).
box(5).
% Box number, color, size.
box_details(1,black,3).
box_details(2,black,1).
box_details(3,white,1).                          %Ternary Predicate
box_details(4,black,2).
box_details(5,white,3).

owners(A,B,C,D,E):-                              %Function implementation as a rule
 box(A), box(B), box(C), box(D), box(E),
 A\=B,A\=C,A\=D,A\=E,
 B\=C,B\=D,B\=E,                                 %Distinct values
 C\=D,C\=E,
 D\=E,
 box_details(A,ColorA,_), box_details(B,ColorA,_),   % Ann and Bill have same color
 box_details(D,ColorD,_), box_details(E,ColorD,_),   % Don and Eric have same color
 box_details(C,_,SizeC), box_details(D,_,SizeC),     % Charlie and Don have same size
 box_details(E,_,SizeE), box_details(B,_,SizeB),
 SizeE < SizeB.                                  % Eric's box is smaller than Bill's
```

Example 2 : Kinship Domain

It consists of

- Object – People
- Unary Predicates - Male and Female
- Binary Predicates - Parent,Brother,Sister
- Functions – Father, Mother
- Relations – Brotherhood, sisterhood.

Examples

The kinship domain:

- Brothers are siblings

  $\forall x,y\ Brother(x,y) \Rightarrow Sibling(x,y)$

- Male and female are disjoint categories

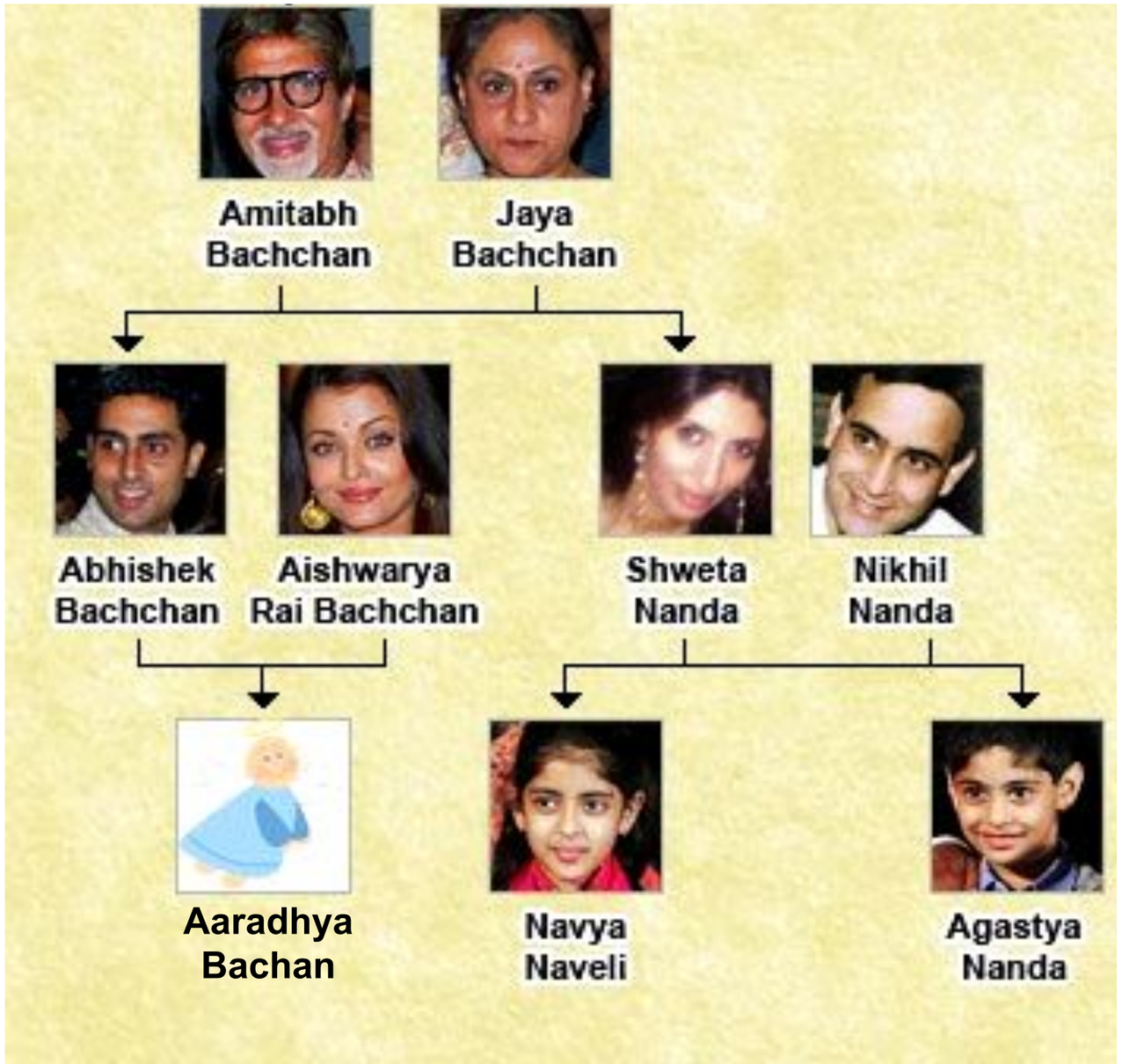
  $\forall x,\ Male(x) \Leftrightarrow \neg Female(x)$

- Parent and child are inverse relations

  $\forall p,c\ Parent(p,c) \Leftrightarrow Child(c,p)$

Example 2 :

Kinship



```
/* Kinship Program */
/* facts */
male(amitabh).
male(abhishek).
male(agastya).
male(nikhil).
female(jaya).
female(aishwarya).
female(aaradhya).
female(shweta).
female(navya).
parent(amitabh, abhishek).
parent(jaya, abhishek).
parent(amitabh, shweta).
parent(abhishek, aaradhya).
parent(aishwarya, aaradhya).
parent(shweta, navya).
parent(shweta, agastya).
parent(nikhil, navya).
/* rules */
mother(M,X):- parent(M,X),female(M).
father(F,X):- parent(F,X), male(F).
grandparent(G, X):- parent(G,P), parent(P,X).
sister(S,X):- parent(Z,X), parent(Z,S), female(S), S\=X.
brother(B,X):- parent(P, B), parent(P, X),male(B), not(B=X).
aunt(X,Y) :- female(X), sister(X,Z), parent(Z,Y).
```

Example 4 : Mathematical set representation

- Constant – Empty set (s = {})
- Predicate – Member and subset ($s_1 \subseteq s_2$)
- Functions – Intersection($\cap$) and union ($\cup$)

- Example: Two sets are equal if and only if each is a subset of the other.

$\forall s_1, s_2\ (s_1 = s_2) \Leftrightarrow (\text{subset}(s_1, s_2) \wedge \text{subset}(s_2, s_1))$

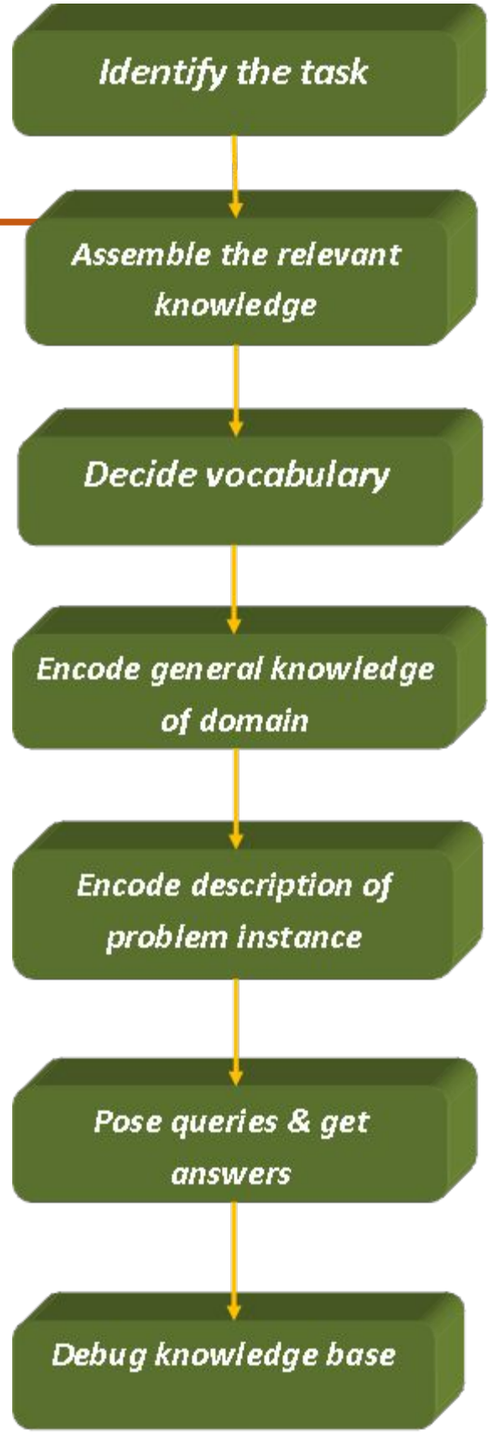
Other eg:

$\forall x, s_1, s_2\ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$

$\forall x, s_1, s_2\ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

Knowledge Engineering in FOPL:

The process of constructing a knowledge-base in first-order logic is called as knowledge-engineering.



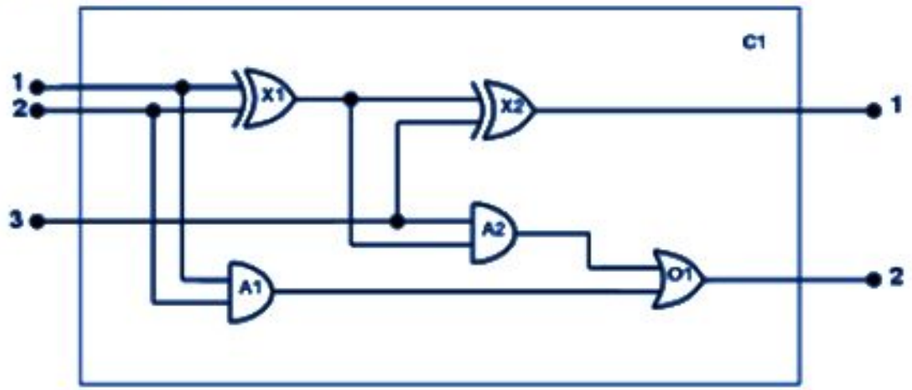
Electronic circuit domain

https://www.javatpoint.com/ai-knowledge-engineering-in-first-order-logic

https://slideplayer.com/slide/260102/

The electronic circuits domain

One-bit full adder



Possible queries:

- does the circuit function properly?
- what gates are connected to the first input terminal?
- what would happen if one of the gates is broken?

and so on

The electronic circuits domain

1. Identify the task
   - Does the circuit actually add properly?

2. Assemble the relevant knowledge
   - Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   - Two input terminals and one output terminal

3. Decide on a vocabulary **of Constant, Predicate and Function Symbols**
   - Alternatives:

Type(X_1) = XOR (function)

Type(X_1, XOR) (binary predicate)

XOR(X_1) (unary predicate)

It can be represented by either binary predicate or individual type.

We use the following Vocabulary to describe an electronic circuit :

Objects : Gates,
Signal Values -1,0 ,
Terminals,
wires,
circuit

Constant Symbols : C_1(maps to object Circuit)
X_1(maps to object XOR gate 1),
X_2 (maps to object XOR Gate 2),
A_1(maps to object AND gate 1),
A_2 (maps to object AND gate 2),
O_1(maps to object OR gate 1),
1(maps to Terminal 1),
2(maps to Terminal 2),
3(maps to Terminal 3)

We use the following Vocabulary to describe an electronic circuit :

Predicate (s)

: Gate(X)	%returns true if X is a gate
: Circuit(X)	%returns true if X is a circuit
: Terminal(X)	%returns true if X is a Terminal
: Arity(X, i, j)	%returns true if X(gate or circuit) has i input terminals and j output terminals
: Connected(t1, t2)	%returns true if terminal t1 is connected to terminal t2

Functions :

Type(X)	%returns the type of gate that X (a constant symbol) maps to.
: In(n, Gate)	%returns nth Input Terminal of a Gate
: Out(n, Gate)	%returns nth Output Terminal of a Gate
: Signal(t)	%returns value of signal at Terminal t which could be either 0 or 1.

4. Encode general knowledge of the domain

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

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

2.The signal at every terminal is either 1 or 0 (but not both)

$\forall t$ Signal$(t) = 1 \vee$ Signal$(t) = 0$

$1 \neq 0$

3. Connected is a commutative predicate.

$\forall t_1,t_2$ Connected$(t_1, t_2) \Rightarrow$ Connected(t_2, t_1)

4. An OR gate's output is 1 if and only if any of its input is 1.

$\forall g\ \text{Type}(g) = \text{OR} \Rightarrow$
$\text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \exists n\ \text{Signal}(\text{In}(n,g)) = 1$

5. An AND gate's output is 0 if and only if any of its input is 0.

$\forall g\ \text{Type}(g) = \text{AND} \Rightarrow$
$\text{Signal}(\text{Out}(1,g)) = 0 \Leftrightarrow \exists n\ \text{Signal}(\text{In}(n,g)) = 0$

6. An XOR gate's output is 1 if and only if any of its inputs are different:

$\forall g\ \text{Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow$
$\text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g))$

5. Encode the specific problem instance

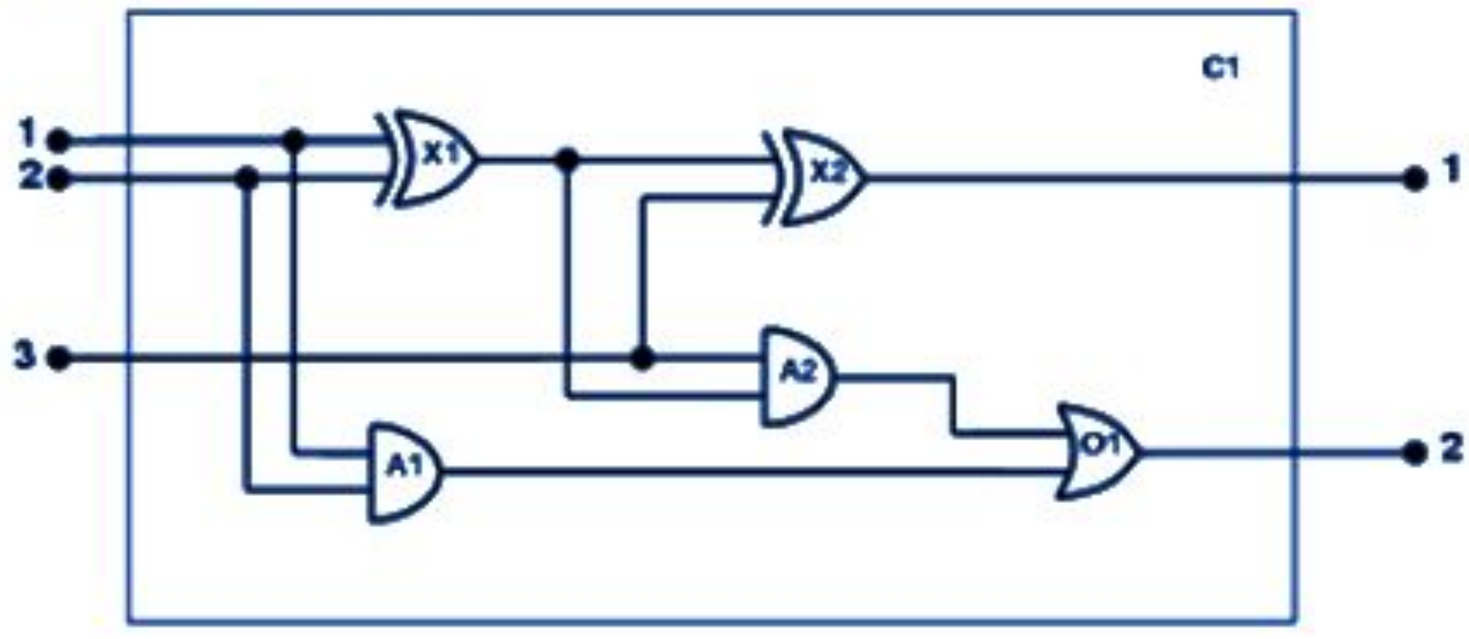
- First we categorize the gates:

$Type(X_1) = XOR$
$Type(A_1) = AND$
$Type(O_1) = OR$

$Type(X_2) = XOR$
$Type(A_2) = AND$

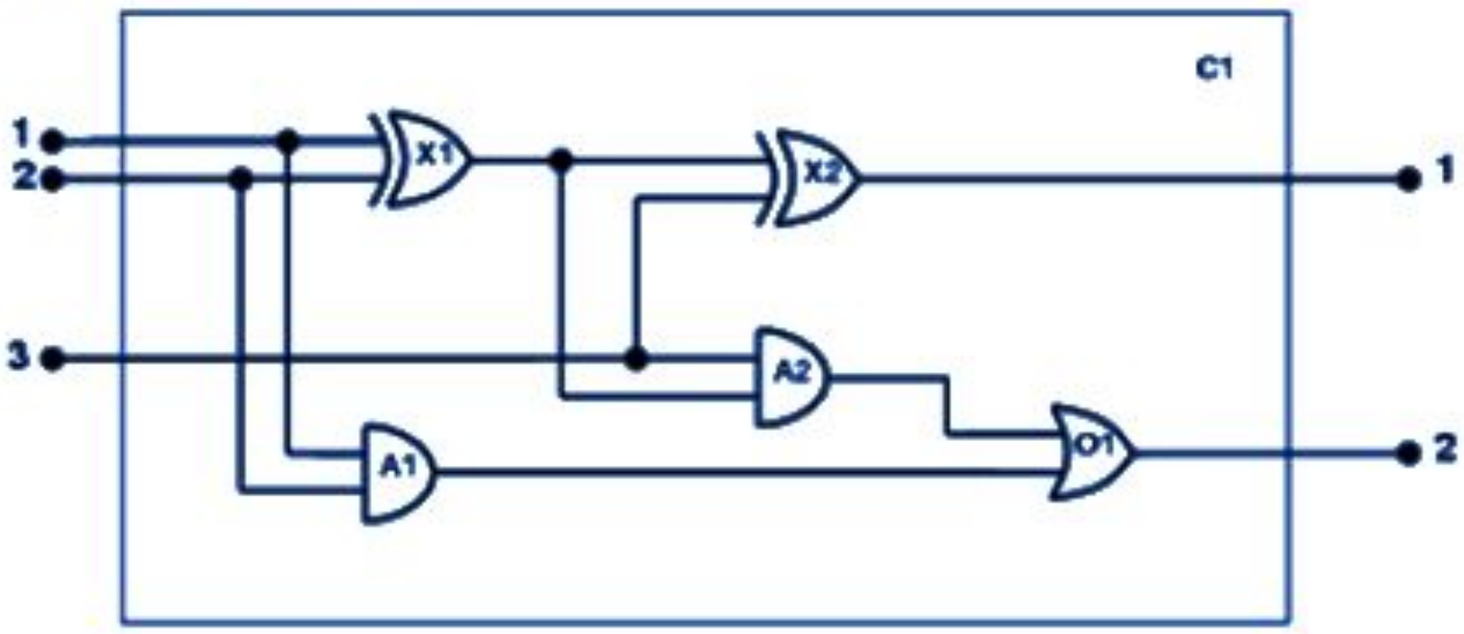
- **We specify the Arity of the Circuit and each gate :**

Arity(C1, 3, 2)
Arity(X1, 2, 1)
Arity(X2, 2, 1)
Arity(A1, 2, 1)
Arity(A2, 2, 1)
Arity(O1, 2, 1)



5. Encode the specific problem instance

– Then show the connections between them:



Connected(In(1,C1), In(1, X1))
Connected(In(2,C1), In(2, X1))

Connected(In(1,C1), In(1, A1))
Connected(In(2,C1), In(2, A1))

Connected(Out(1,X1), In(1, X2))
Connected(In(3,C1), In(2, X2))

Connected(In(3,C1), In(1, A2))
Connected(Out(1,X1), In(2, A2))

Connected(Out(1,A2), In(1, O1))
Connected(Out(1,A1), In(2, O1))

7. Query and get answers

Query represented in FOPL:

∃ i1, i2, i3, o1, o2
signal(in(1,c1)) = i1 ∧
signal(in(2,c1)) = i2 ∧
signal(in(3,c1)) = i3 ∧
signal(out(1,c1)) = o1 ∧
signal(out(2,c1)) = o2

In Prolog we can write a rule find(i1, i2, i3, o1, o2) to do the same:

-? find(i1, i2, i3, o1, o2)

7. Query and get answers

We can examine output of each gate:

∃ i1, i2, o
signal(in(1,c1)) = i1 ∧
signal(in(2,c1)) = i2 ∧
signal(out(1,x1)) = o

Practice Problems

Problem 1

Using the following notation, express the given statements in FOPL :

Q(x) : x is a rational number

R(x) : x is a real number

LESS(x, y) : x is less than y

Statements:

i) There exists a number that is rational.

ii) Every rational number is a real number

iii) For every number x, there exists a number y, which is greater than x.

iv) Some real numbers are rational numbers.

Using the following notation, express the given statements in FOPL :

Q(x) : x is a rational number

R(x) : x is a real number

LESS(x, y) : x is less than y

Statements:

i) There exists a number that is rational.

ii) Every rational number is a real number

iii) For every number x, there exists a number y, which is greater than x.

iv) Some real numbers are rational numbers.

Solution:

(i) $(\exists x)\ Q(x)$

(ii) $(\forall x)\ (Q(x) \rightarrow R(x))$

(iii) $(\forall x)\ (\exists y)$ LESS(x, y)

(iv) $(\exists x)\ (Q(x) \wedge R(x))$

Each of the expressions (i), (ii), and (iii) is called a formula or a well-formed formula or wff.

Problem 2

Problem Description:

- The domain of interest is the natural numbers, ℕ.
- There are objects, 0, 1, 2, 3, ….
- There are functions, addition and multiplication, as well as the square function, on this domain.
- There are predicates on this domain, “even,” “odd,” and “prime.”
- There are relations between elements of this domain, “equal,” “less than”, and “divides.”
- For our logical language, we will choose symbols 1, 2, 3, add, mul, square, even, odd, prime, div, and so on, to denote these things.

Consider some ordinary statements about the natural numbers, express the same in FOPL:

1. Every natural number is even or odd, but not both.
2. A natural number is even if and only if it is divisible by two.
3. If some natural number, x, is even, then so is x^2.
4. A natural number x is even if and only if x+1 is odd.
5. Any prime number that is greater than 2 is odd.
6. For any three natural numbers x, y, and z, if x divides y and y divides z, then x divides z.
7. There exists an odd composite number.

Solution

1. $\forall x((\text{even}(x) \vee \text{odd}(x)) \wedge \neg(\text{even}(x) \wedge \text{odd}(x)))$
2. $\forall x(\text{even}(x) \leftrightarrow \text{div}(x,2))$
3. $\forall x(\text{even}(x) \rightarrow \text{even}(x^2))$
4. $\forall x(\text{even}(x) \leftrightarrow \text{odd}(x+1))$
5. $\forall x(\text{prime}(x) \wedge x>2 \rightarrow \text{odd}(x))$
6. $\forall x \forall y \forall z(\text{div}(x,y) \wedge \text{div}(y,z) \rightarrow \text{div}(x,z))$
7. $\exists x(\text{odd}(x) \wedge \text{composite}(x))$

Problem 3

Using a language with variables ranging over people, and predicates trusts(x,y), politician(x), crazy(x), knows(x,y), related-to(x,y), and rich(x), write down first-order sentences asserting the following:

1. **Nobody trusts a politician.**
2. **Anyone who trusts a politician is crazy.**
3. **Everyone knows someone who is related to a politician.**
4. **Everyone who is rich is either a politician or knows a politician.**

Using a language with variables ranging over people, and predicates trusts(x,y), politician(x), crazy(x), knows(x,y), related-to(x,y), and rich(x), write down first-order sentences asserting the following:

Solution

1	Nobody trusts a politician.	$\forall x \forall y$ politician(x) $\rightarrow \neg$ trusts(y,x)
2	Anyone who trusts a politician is crazy.	$\forall x \forall y$ (trust(x,y) $\wedge$ politician(y)) $\rightarrow$ crazy(x)
3	Everyone knows someone who is related to a politician.	$\forall x \exists$ y $\exists$ z (knows(x,y) $\wedge$ related(y,z) $\wedge$ politician(z))
4	Everyone who is rich is either a politician or knows a politician.	$\forall x \exists$ y rich(x) $\rightarrow$ (politician(x) v (knows(x,y) $\wedge$ politician(y))

Problem 4

Let C(x) mean “x is a used-car dealer,” and H(x) mean “x is honest.” Translate each of the following formulas into English:

(i) $(\exists x)C(x)$

(ii) $(\exists x)\, H(x)$

(iii) $(\forall x)C(x) \rightarrow \sim H(x))$

(iv) $(\exists x)\,(C(x) \wedge H(x))$

Let C(x) mean “x is a used-car dealer,” and H(x) mean “x is honest.” Translate each of the following formulas into English:

(i) There is (at least) one (person) who is a used-car dealer.

(ii) There is (at least) one (person) who is honest.

(iii) All used-car dealers are dishonest.

(iv) (At least) one used-car dealer is honest.

THANK YOU

Preet Kanwal

Department of Computer Science & Engineering

preetkanwal@pes.edu

+91 80 6666 3333 Extn 724