- Knowledge Representation

It is a achieved through many ways:

1- logic representation involves:

- (proposition logic)
- (predicate logic)
- (fuzzy logic)

2- production representation

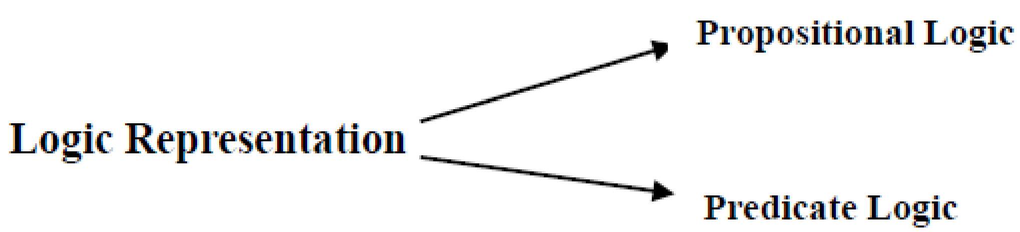
3- network representation

- Semantic network
- Conceptual graph
- Frames & scripts

4- functional representation

LOGIC REPRESENTATION

In order to determine appropriate actions to take to achieve goals, an intelligent system needs to compactly represent information about the world and draw conclusions based on general world knowledge and specific facts.



Propositional Logic

Propositional logic : is a statement about the world, a proposition can assume one of two values true or false.

Propositional symbols are used to represent facts. Each symbol can mean what we want it to be. Each fact can be either true or false. Propositional symbols: P, Q, etc. representing specific facts about the world. For example,

P1 = “Water is a liquid”.

P2= “ Today is Monday”.

P3= “It is hot”

Q1= “The goround is wet”

Q2=”It is raining”

Example: P= water is liquid. T

Q= today is Friday. F

In propositional logic the following symbols are used to generate compound propositions (∧, ∨, ¬ , ⟶ , ≡)

ويتم الاعتماد على جدول الحقيقة (truth table) التابع لهذه الرموز

∧ Conjunction, and, & الوصل

∨ Disconjunction , or الفصل

¬ Negation , not النفي

⟶ Implication الاستلزام

≡ Equivalence, if and only if , xor, التطابق

p	q	¬ p	p ∧ q	p ∨ q	p→ q (¬ p∨q)
T	T	F	T	T	T
T	F	F	F	T	F
F	T	T	F	T	T
F	F	T	F	F	T

Let p & q be two propositions

$\neg$ p	is proposition
p $\wedge$ q	is proposition
p $\vee$ q	is proposition
p$\rightarrow$ q ($\neg$ p$\vee$q)	is proposition

and some identities in propositional logic:

$\neg(\neg p) \Leftrightarrow p$	
$(p \vee q) \Leftrightarrow (q \vee p)$ $(p \wedge q) \Leftrightarrow (q \wedge p)$	قانون التبادل
$(p \wedge q) \vee r \Leftrightarrow (p \vee r) \wedge (q \vee r)$ $(p \vee q) \wedge r \Leftrightarrow (p \wedge r) \vee (q \wedge r)$	قانون التوزيع
$(p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$ $(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$	قانون التجميع
$p \rightarrow q \Leftrightarrow \neg p \vee q$ $p \rightarrow q \Leftrightarrow \neg q \rightarrow \neg p$	قانون التضاد
$\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$ $\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q)$	قانون ديموركان

Definition: expression in propositional calculate is WFF if they are formed of legal symbol through sub sequence of their rules

Ex: $((p \wedge q) \rightarrow r) = \neg p \vee \neg q \vee r$ it is WFF since p, q ,r are propositions

- p, q & r is propositional symbols (WFF)
- $p \wedge q$ is WFF
- $(p \wedge q) \rightarrow r$ is WFF
- $\neg p$, $\neg q$ is WFF
- $\neg p \vee \neg q \vee r$ is WFF

Then $((p \wedge q) \Rightarrow r) = \neg p \vee \neg q \vee r$ is WFF

Ex: it's not raining or it's sunny

يمكن تمثيل هذه الجملة بالصيغة التالية

¬ p   it's not raining
q   it's sunny
¬ p∨q

Ex:

1. if Ali is interested in logic (A), then either he will sign up for logic next semester (B) or he is lazy (C).
2. if Ali has read books about logic on his own (D), then he is interested in logic (A).
3. Ali has read book logic on his own (D).
4. Ali is not last.

Put a proposition for each statement and get the axioms.
Sol:

1- A→ B ∨ C
2- D → A
3- D
4- ¬ C

- Predicate Logic

A predicate names a relationship between zero or more objects. Predicate logic allows us to deal with the component of a sentence. For example,

P= "It rained on Tuesday"

Predicate representation: weather(tuesday, rain)

Q= " Water is liquid"

Predicate representation: property(water, liquid)

Predicate name (property) — Arguments (water, liquid)

For generality, predicate logic representation allows us to use variables, for example,

P1 = “It rained on Tuesday” weather(tuesday, rain)

P2= “It rained on Wednesday” weather(wedensday, rain)

P3= “It rained on Thursady” weather(thursday, rain)

. .

. .

. .

Pn= “It rained on Monday” weather(monday, rain)

It is more efficient to use variables in the representation format of the predicate.

weather(X, rain)

where X { Sunday, Monday, … , Saturday}

Constant: A constant refers to a specific object. A constant starts with a lower case letter. Such as (car, blue, man,….)

Variable: A variable is used to refer to a general classes of objects. A variable starts with an upper case letter. (X,Y, Man, Person, City,….)

Clauses: A clause is one or more predicates combined using the connectives above. A clause with one predicate is called a unite clause.

Horn Clause: A horn clause has the following form:

b1( ) b2 ( ) … bn ( ) →a( )

where b1( ), …, bn( ) and a( ) are all positive predicates. a( ) is called the head of the horn clause. b1( ), …, bn( ) is called the body of the horn clause. There are three cases of the horn clause:

1. the horn clause has no body (i.e. ⟶ a) in this case the clause is called a **fact** and is written as a.
2. the horn clause has the standard form : b1^b2^….^bn⟶ a in this case the clause is called a **rule**.
3. the horn clause has no head b1^b2^….^bn⟶ in this case the expression b1^b2^….^bn represent a set of sub goals which has to be prove.

Quantification there are two quantifications that are used to describe the possible values the variable can assume:

∀ for all

∃ there exist

Ex:

∀X weather (X, rain)	X ∈ the days of week
∃X weather (X, rain)	true for some values of X
∀X like(X, football)	X ∈ { male students} or X ∈ { set of all people}

∀ use when the predicate is true for all values of X

∃ use when the predicate is true for some values of X

Ex: All basketball players are tall

∀X basketball- players(X) ⟶ tall(X) X ∈{ for all people} Or

∀X{[player(X) ^ play(X, basketball)] ⟶ tall(X)}

∀X∀Y{[player(X) ^ play(X,Y) ^ game(Y, basketball)] ⟶ tall(X)}

X∈ {set of people}

Y ∈ {set of games}

Ex: no one like taxes

¬∀X like(X, taxes)

Ex: ∃X bird(X)^ ¬flies(X)

there exist some bird that doesn't fly هناك بعض الطيور لاتستطيع الطيران

- Unification: the process of making two argument look like.
ex: توحيد الرموز هو جعل اثنين من المعاملات متساويين

Like(ali, X) ⟶ like(ali, computer)
Unification set= {(computer, X)}

- Function Expression: a function name starts with a lower case letter. It has an associated number of arguments. Each argument can be a constant, a variable or another function.

Ex: product(A, B, C)

product(2, 3, P)

- Function Expression: مثل father(ali) المتكونة من اسم الدالة يتبعه عدد من الـ(arguments) والتي تكون عبارة عن (terms) متكونة من (ثابت، متغير، دالة).

Term: a term can either be a constant, variable or function.

Some examples of knowledge representation:

Ex: find the predicate logic statement which equivalent to following:

1) mosul is city of iraq   is-city-of(mosul, iraq)

2) 2+3=5   equal(plus(2,3),5)

3) 2+3≠6   ¬ equal(plus(2,3),6) or plus(2,3) ⟶¬ equal(6)

4) if jen cross or jump, then he'll pass: cross(jen) ∨jump(jen) →pass(jen)

5) all peoples which are not poor and intelligent they will be happy.

∀X people(X) ^ ¬ poor(X) ^ intelligent(X) → happy(X)

6) If it does not rain tomorrow, Zeki will go to the lake.

¬ weather(tomorrow, rain) → go (zeki, lake)

Ex: convert the following to predicate Logic

1- Marcusاسم شخص was a man.
2- Marcus was a Pompeianاسم منطقة.
3- All pompeians were Romans.
4- Caesar قيصرwas a rulerحاكم.
5- All Romans were either loyalولاء to Caesar or hatedيكره him.
6- Everyone is loyal to someone.
7- People only try to assassinateيغتال rules they are not loyal to.
8- Marcus tried to assassinate Caesar.

Sol: the predicate Logic

1- man(Marcus)
2- Pompeian(Marcus)
3- $\forall x$ Pompeian(x) → Roman(x)
4- Ruler(Caesar)
5- $\forall x$ Roman(x) → loyalto (x, Caesar) ∨ hate(x,Caesar)
6- $\forall x \exists y$ loyalto(x,y)
7- $\forall x \forall y$ people(x)∧ruler(y)∧try assassinate(x,y)→ ¬ loyalto(x,y)
8- try assassinate(Marcus,Caesar)

- Reasoning with knowledge :-

In logic two common methods are used:

1- Modus ponens
2- Resolution

1- Modus ponens

Given p → q and p is true
There we can conclude q is true
Ex: "if it rain then the ground is wet" . true
" it rain" . true
" the ground is wet " q is true الاستنتاج
p → q
p
∴ q is true

2- Resolution:

It is a process of taking two clauses in normal form such that one must contain the negation of a literal in the second clause. The outcome is a new clause with all the literals of the two clauses except the concerned literal& its negation.

The new clause must be in the form such that the substation required to unify the literals are applied to the new clause.

C1: p ∨ q ∨¬ r
C2: p∨ t∨ s∨ r } The parent of the resolution

The resultant p ∨ q ∨ t ∨ s

Ex:

C1: $\forall x$(man(x) →mortal(x))
C2: man(ali)

Convert to normal form

C1: ¬ man(x) ∨ mortal(x)
C2: man(ali)
(ali / x) → Mortal(ali)

- To proof by Refutation :

C1: p
C2: ¬ p
→ □ Empty clause (contradiction)

To proof a goal by refutation do the following

- convert all clauses into Normal Form
- add negation of the goal to the set of knowledge (convert to Normal Form)
- resolve clauses

If any empty clause (contradiction) is obtained then the goal is true. Else if the system can not generate more new clauses then the goal is false.

Ex: given c1: ∀x(man(x) → mortal(x))
c2: man(ali)
goal: mortal(ali)?

C1: ¬ man(x) ∨ mortal(x)
C2: man(ali)
G: ¬ mortal(ali)

¬ man(x) ∨ mortal(x)
¬ mortal(ali)
ali/x → ¬ man(ali)

¬ man(ali)
man(ali)
→ □

Ex: goal: mortal(y)

C1: ¬ man(x) ∨ mortal(x)
C2: man(ali)
G: ¬ mortal(y)

¬ man(x) ∨ mortal(x)
¬ mortal(y)
x/y → ¬ man(x)

¬ man(x)
man(ali)
ali/x → □

x=y=ali

Ex: All people that are not poor and are smart are happy those people that read are not stupid. Ali can read and is wealthy. Happy people have exciting lives.

Question: can any one be found with an exciting life? And how is?

Resolve:

C1: ∀x[¬ poor(x) ^ smart(x) → happy(x)]

C2: ∀x[read(x) → smart(x)]

C3: read(ali) ^ ¬ poor(ali)

C4: ∀x[happy(x) → exciting (x)]

Goal: ∃z exciting(z)

Convert to clause form:

C1: poor(x1) ∨ ¬ smart(x1) ∨ happy(x1)
C2: ¬ read(x2) ∨ smart(x2)
C3: read(ali)
C4: ¬ poor(ali)
C5: ¬ happy(x3) ∨ exciting(x3)
G: ¬ exciting(z)

Resolution:

¬ exciting(z)
¬ happy(x3) ∨ exciting(x3)
→ (x3/z) C6: ¬ happy(x3)

C6: ¬ happy(x3)
C1: poor(x1) ∨ ¬ smart(x1) ∨ happy(x1)
→ (x1,x3) C7: poor(x1) ∨ ¬ smart(x1)

C7: poor(x1) ∨ ¬ smart(x1)
C2: ¬ read(x2) ∨ smart(x2)
→ (x1,x2) C8: ¬ read(x1) ∨ poor(x1)

C8: ¬ read(x1) ∨ poor(x1)
C3: read(ali)
→ (ali,x1) C9: poor(ali)

C9: poor(ali)
C4: ¬ poor(ali)
→ □ ∴ exciting(ali)

∴ goal is proved with z=x3=x1=ali

Ex: we wish to prove that spike will die from the following statement"

Statement	Predicate form	Clause form
1- all dogs are animals	∀x dog(x)→ animal (x)	¬dog(x) ∨animal(x)
2- spike is a dog	dog(spike)	dog(spike)
3- all animals will die	∀x animal(x)→ die(x)	¬animal(y)∨die(y)
4- conclusion: ¬die(spike)		

¬dog(x) ∨animal(x)   ¬animal(y)∨die(y)

¬dog(x) ∨die(y)   dog(spike)

die(spike)   ¬die(spike)

□

Ex: consider the following information:

C1: p ∨¬(q ^ ¬ r)

C2: (¬ s ∨ t) ^ (¬ t ∨ p)

C3: ¬ s ∨ w

C4: s ^ ¬ r

Goal: w ^ ¬ q

1- information

C1: p ∨¬q ∨ r

C2: ¬s ∨ t

C3: ¬t ∨ p

C4: ¬ s ∨ w

C5: s

C6: ¬ r

G: ¬ w ∨ q

2- resolution

g: ¬ w ∨ q ; c4: ¬ s ∨ w → q ∨¬ s

q ∨¬ s ; c5:s → q

q ; c1: p ∨¬q ∨ r → p ∨ r

p ∨ r ; c6: ¬ r → p

g: $\neg w \lor q$	$\neg w \lor p \lor r$	$p \lor r \lor \neg s$	$p \lor \neg s$	p
c1: $p \lor \neg q \lor r$	c4: $\neg s \lor w$	c6: $\neg r$	c5: s	