Knowledge representation

To create programs that have "intelligent" qualities, it is necessary to develop techniques for representing knowledge. Unlike to people, computers do not have the ability to acquire knowledge on their own.

Al programs use structures called knowledge structures to represent objects, facts, rules, relationships, and procedures. The main function of the knowledge structure is to provide the needed expertise and information so that a program can operate in an intelligent manner. Knowledge structures are usually composed of both traditional data structures and other complex structures such as Logical, frames, scripts, semantic networks, conceptual graph, and ATN(augment transition network).

Knowledge representation schemes

In Al, there are four basic categories of representational schemes: logical, procedural, network and structured representation schemes.

1. **Logical representation**

It uses expressions in formal logic to represent its knowledge base. Predicate Calculus is the most widely used representation scheme.

2. **Procedural representation**

It represents knowledge as a set of instructions for solving a problem. These are usually if-then rules we use in rule-based systems.

3. **Network representation**

It captures knowledge as a graph in which the nodes represent objects or concepts in the problem domain and the arcs -represent relations or associations between them.

4. **Structured representation**

It extends network representation schemes by allowing each node to have complex data structures named slots with attached values.

In this course, we will focus on logic representation schemes Insha'Allah.

Logical representation scheme

The prepositional calculus and the predicate calculus are first of all languages. Using their words, phrases, and sentences, we can represent and reason about properties and relationships in the world. The first step in describing a language is to introduce the pieces that make it up a set of symbols.

1. Prepositional Calculus (PPC)

This type of representation can be used some of conceptions such as:

- Axiom: it always known as truth.
- Proposition: its known as a Boolean sentence, the truth symbols maybe true or false,
- Theorem: it's a Boolean sentence; it can conclude form the axioms.

The PPC is content form the following three parts:

- A set of the concepts, axioms and proposition that can be represented by **Well-Formed Formula (WFF)**. It symbols denote *propositions of* statements about the world that may be either true or raise, such as "the car is red" or "water is wet." WFFs are denoted by uppercase letters near the end of the English alphabet (i.e. P, Q …etc.).
- A set of connections that can connect two or more WFF sentences:

NOT	⌐ ┐	Negation Connection
AND	$\wedge$	Conjunction Connection
OR	$\vee$	Disjunction Connection
IF	$\rightarrow$	Implication Connection
IFF	$\leftrightarrow$	Equivalence Connection

- A set of Inference Rules that can be used to conclude new sentences form the old sentences. We can explain two types of these inference rules:

  a) Modus Ponens (MP)

  If the sentences P and (P→Q) are known to be True, Then this rule infers Q is True.

  b) Modus Tolen (MT)

  If the sentence (P→Q) is known to be True and Q is known to be False, Then this rule infers ¬P is False.

2. Predicate Calculus (PC)

In prepositional calculus, each atomic symbol (P, Q, etc.) denotes a proposition of some complexity. There is no way to access the components of an individual assertion. Predicate calculus provides this ability. For example, instead of letting a single prepositional symbol, P, denote: The entire sentence "it rained on Tuesday," we can create a predicate weather that describes a relationship between a date and the weather, such as: weather (Tuesday, rain) through inference rules we can manipulate predicate calculus expression accessing their individual components and inferring new sentences.

It can represent the predicates by:

name-predicate (parameters).

➢ **Examples of English sentences represented in predicate calculus:**

1- If it doesn't rain tomorrow, Tom will go to the mountains.

¬ weather (rain, tomorrow) → go(tom, mountains).

2- All basketball players are tall.

$\forall X$ (basketball _ player(X) → tall (X))

3- Some people like borrowing.
$\exists X$ (person(X) ∧ likes(X, borrowing))

4- Nobody likes taxes
¬ $\exists X$ likes(X, taxes).

Automatic Theorem Proving

It's also called Resolution technique for theorem proving in propositional and predicate calculus which attempts to show that the negation of the statement produces a contradiction with the known statements. This technique depends on the Refutation that will happen in the Knowledge Base (KB).

Algorithm Resolution technique proofs involve following steps:

1. **Assume that ┐P is True.**
2. **Show that the basic axioms together with ┐P lead to contradiction.**
3. **Conclude that, since the axioms are correct, ┐P must be False.**
4. **Since ┐P is False, P must be True.**

Before done this algorithm, it must be convert all the sentences form WFF to Clause form. Therefore, it can use the following algorithm to done this convert:

Algorithm to convert a WFF to Clause Form :

1. Change P→Q to ┐P∨Q

   P↔Q to (┐P∨Q) ∧ (┐Q∨P)

2. Reduce the range of negative; for example covert

   $\neg(\neg a) \equiv a$

   $\neg(\exists X)a(X) \equiv (\forall X)\neg a(X)$

   $\neg(\forall X)b(X) \equiv (\exists X)\neg b(X)$

   $\neg(a \wedge b) \equiv \neg a \vee \neg b$

3. Relocate the universal quantifier ∀ to front of the clauses. Example:

$\forall X\ \forall Y$ (P(X) ∨ ┐Q(Y)) to $\forall X$ P(X) ∨ $\forall Y$ ┐Q(Y)

4. Rewrite the sentence in conjunction normal form (i.e. the AND would be distributed with respect to the OR). For example:

   (A ∧ B) ∨ (B ∧ C) would be redistribute (A∨ B) ∧ (B∨ C) ∧ B ∧(A∨ C).

Ex1: You have the following axioms:

1. feather (rook).
2. $\forall X$(feather(X)⟶ bird(X))

By Automatic Theorem Proving prove that **bird (rook)**

Firstly, change WFF to Clause form

1. feather (rook).
2. ┐feather(X) ∨ bird(X))

Add the negative theorem that need to prove

3. ¬ bird (rook)

(1),(2) 4. bird (rook) according to MP

(3),(4) empty

So, the theorem ¬ bird (rook) must be False, therefore the theorem **bird(rook)** must be True.

Ex2: You have the following axioms:

1. father (ali, ahmed).
2. has (ali, money)
3. (father (Z, X) ∧ has (Z, Y)) ⟶ has (X, Y)

By Automatic Theorem Proving prove that **has (ahmed, money)**

Firstly, change WFF to Clause form

1. feather (ali, ahmed).
2. has (ali, money).
3. ┐father(Z,X) ∨ ┐has(Z, Y) ∨ has(X, Y)

Add the negative theorem that need to prove

4. ¬ has (ahmed, money)

(2),(3) 5. ┐father(ali, X) ∨ has(X, money)

(4),(5) 6. ┐ father(ali, ahmed)
(1),(6) empty

So, the theorem ¬ has (ahmed, money) must be False, therefore the theorem **has (ahmed, money)** must be True.

Ex3: You have the following axioms:

1. Fido is a dog.
2. All dogs are animals.
3. All animals will die.

By Automatic Theorem Proving prove that **Fido will die**

Firstly, change all sentences to WFF

1. dog (fido).
2. $\forall X$(dog(X)⟶ animal(X))
3. $\forall Y$(animal(Y)⟶ die(Y)).
   Add the negative theorem that need to prove
4. ¬ die (fido)

After that, change WFF to Clause form

1. dog (fido).
2. ┐ dog(X) ∨ animal(X))
3. ┐ animal(Y) ∨ die(Y)).
4. ¬ die (fido)

(1),(2) 5. animal (fido) according to MP
(3),(5) 6. die(fido) according to MP
(4),(6) empty

So, the theorem ¬ die (fido) must be False, therefore the theorem **Fido will die** must be True.

Ex4: You have the following axioms by WFF:

1. gt (2017, 79).
2. man (ali)
3. man (X2) → dead (X2, 79)
4. Now= 2017
5. alive (X4, T3) → ┐ dead (X4, T3)
6. [dead (X4, T3) ∧ gt (T4, T3)] → dead (X4, T4)

By Automatic Theorem Proving prove that **: ┐ alive (ali, Now).**

Firstly, change WFF to Clause form

1. gt (2017, 79).
2. man (ali)
3. ┐ man (X2) ∨ dead (X2, 79)
4. Now= 2017
5. ┐ alive (X4, T3) ∨ ┐ dead (X4, T3)
6. ┐ dead (X4, T3) ∨ ┐ gt (T4, T3) ∨ dead (X4, T4)

Add the negative theorem that need to prove

7. alive (ali, Now)

(4),(7) 8. alive (ali, 2017)
(2),(3) 9. dead (ali, 79) according to MP
(6),(9) 10. ┐ gt (T4, 79) ∨ dead (ali, T4) according to MP
(1),(10) 11. dead (ali, 2017) according to MP
(5),(11) 12. ┐ alive (ali, 2017) according to MT
(8),(12) empty

So, the theorem **alive (ali, 2017)** must be False, therefore the theorem ┐ **alive (ali, 2017)** must be True.

Ex5: You have the following paragraph:

All people that are not poor and smart are happy. Those people that read are smart. John can read. Also, he is not poor. Happy people have exciting lives. Can anyone be found with an exciting life?

Firstly, change all sentences to WFF

1) All people that are not poor and smart are happy.
$\forall X$ [┐ poor(X) Λ smart(X)] ⟶ happy(X).
2) Those people that read are smart.
$\forall Y$ (read (Y) ⟶ smart(Y)).
3) John can read.
read (john)
4) John is not poor
┐ poor (john)
5) Happy people have exciting lives.
$\forall Z$ (happy (Z) ⟶ exciting (Z)).
6) The negation of the conclusion is:
┐ exciting (W).

After that, change WFF to Clause form

1. poor(X) ∨ ┐ smart(X) ∨ happy(X).
2. ┐ read (Y) ∨ smart(Y)
3. read (john)
4. ┐ poor (john)
5. ┐ happy (Z)) ∨ exciting (Z)

Add the negative theorem that need to prove

6. ┐ exciting (W)

(1),(4) 7. ┐ smart(john) ∨ happy(john)
(2),(3) 8. smart(john)
(7),(8) 9. happy(john)
(5),(9) 10. exciting(john)
(6),(10) empty

The theorem ┐ **exciting (john)** must be False, therefore the theorem **exciting (john)** must be True. So, we can find one person has an exciting life.