College of Education for
Pure Science/ Ibn Al-Haitham
Computer Science Dept.
3rd Class

M.SC. Nadia Mohammed
2022-2021

Artificial Intelligence

FIRST COURSE

3RD CLASS

مدرس المادة

م. نادية محمد

2021-2020

References:

1- Elin Rich, "Artificial Intelligence",1989.
2- William A. Stubblefield & Luger E.George,"Artificial Intelligence and
the Design of Expert Systems", 1998.

week	Theoretical Content	Laboratory Work
1	General introduction to Artificial intelligence.	Introduction to prolog language & characteristics uses.
2	Foundation and History of Artificial Intelligence, Applications of Artificial Intelligence, Architecture of Artificial Intelligence, Language and environment	Prolog language Components (Facts, Rules, Questions).
3	Define the problem as a state space.	Variables of prolog language.
4	Production system	Conjunctions & backtracking.
5	Problem characteristics.	Data type & program structure.
6	Some example of A.I. problem (8-puzzle, monky and banana,……..)	Examples
7	Search technique (Blind search) DFS and BFS.	Read &write function.
8	Intelligent search technique (Hill Climbing, Generate and test).	More examples on read & write function.
9	Best first search.	Arithmetic & logical operation.
10	A - algorithms.	Examples for the previous subjects.
11	A^* - algorithms	Examples for the previous subjects.
12	Min-Max and Alpha-Beta algorithms.	Examples for the previous subjects.
13	Problem reduction and (AND/OR) graph.	Examples for the previous subjects.
14	Forward and backward chaining.	Examples for the previous subjects.
15	Black board approach.	Examples for the previous subjects.
16	First term exam.	First term exam
17	Knowledge representation (Propositional logic & Predicate logic).	Cut & fail & negation.
18	Logical representation.	Cut & fail &negation.

19	(Procedural & network & structured) representations.	Example and exercise on cut & fail & negation
20	Clause form algorithm.	Recursion (Tail and non Tail recursion).
21	Resolution in propositional logic algorithm; propositional resolution.	Recursion (Tail and non Tail recursion).
22	The unification Algorithm.	Example and exercise on (Tail and non Tail recursion).
23	Resolution in predicate logic Algorithm; Resolution.	Repeat in prolog
24	(Continue to)Resolution in predicate logic algorithm; Resolution	List in prolog
25	Expert system (Introduction, Architecture, Characteristic).	Syntax of list & head and tail of list
26	Rule based Application of Expert system.	Program on list
27	Example on Expert system.	Program on list
28	Introduction to neural network	Program on list
29	(Continue to) Introduction to neural network	Example to simple expert system
30	Introduction to genetic algorithm	Example to simple expert system
31	Second term exam	Example to simple expert system
32	Final Exam	Second term exam

problem solving: لحل أي مشكلة يجب تحديد:

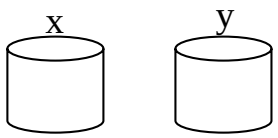
1. initial state and goal state. حالة البدء والحالة النهائية
2. problem state space. وصف مجال المشكلة أي وصف كل الحلول الممكنة لها عن طريق تري
3. find the path between start state and goal state.
4. search about the solution between the start state and the goal state.

Problem state space:

1. knowledge base القواعد العامة التي تستخدم في عملية الوصف المنطقي للمشكلة
   - find all parameter. إيجاد المتغيرات
   - State all the value of parameter. نضع قيم المتغيرات
   - State the initial state and goal state. تحديد نقطة البداية والنهاية
   - Find all the rule of the problem. القواعد الخاصة بالمشكلة
   - Find the tree of the solution. الشجرة الخاصة بالحل

Ex(1):

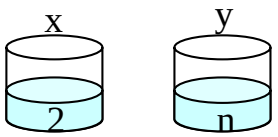
Initial state



4 gallon 3 gallon

You are given two jugs 4 gallon and 3 gallon neither has any measuring markers on it there is pump that can be used to full the jugs with water. How can exactly gate 2 gallon in 4 gallon jug and n gallon in 3 gallon jug.

Goal state

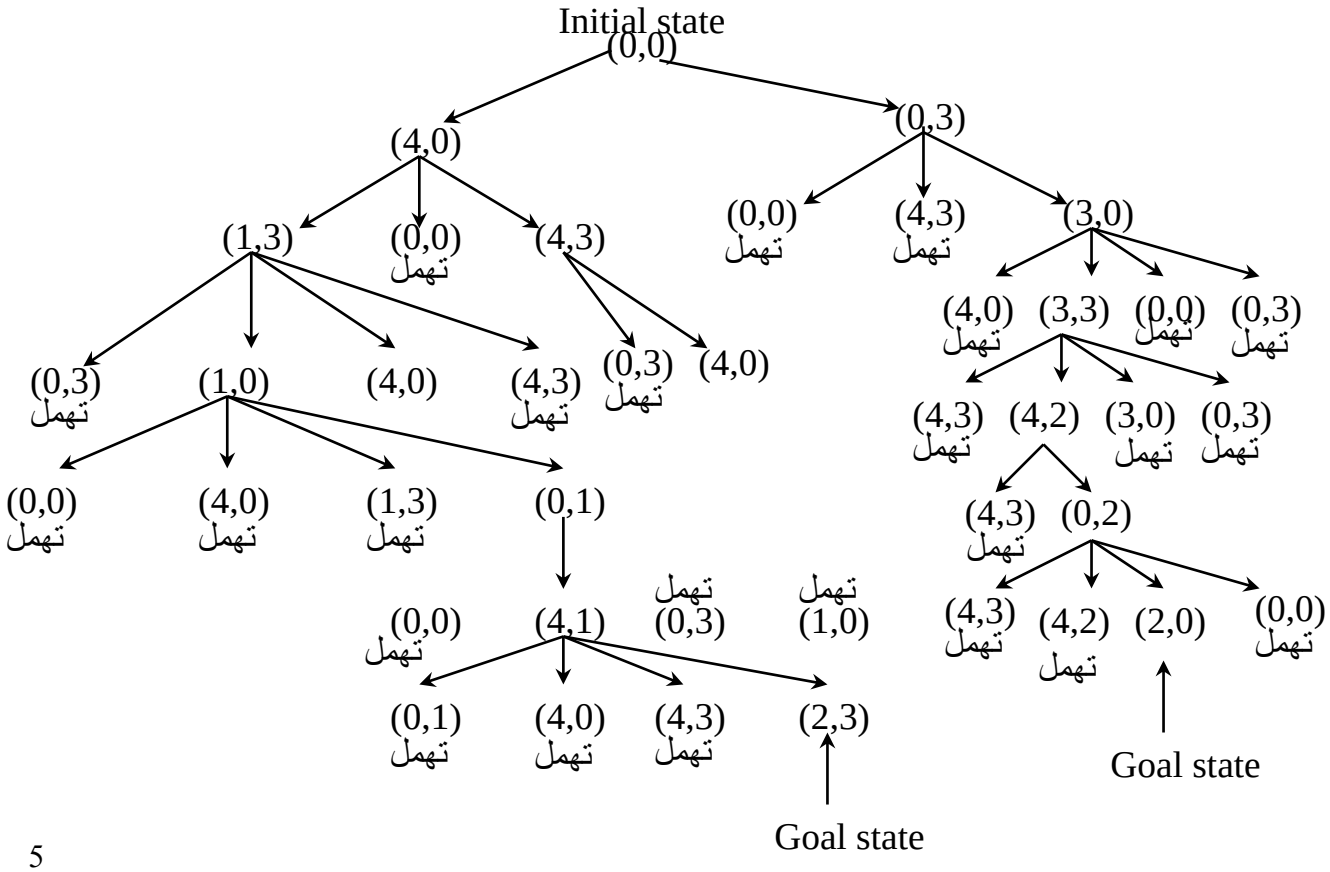


4 gallon 3 gallon

Sol:

1. x, y. حددنا المتغيرات
2. x=1,2,3,4 y=1,2,3. تحديد القيم
3. initial state (0,0) goal state (2,n). نقطة البداية
4. the rule of the problem:
   a. if x < 4 then (4,y) if y < 3 then (x,3). ملئ
   b. if x > 0 then (0,y) if y > 0 then (x,0). تفريغ
   c. if x+y <=4 and y>0 then (x+y,0)
      if x+y <=3 and x>0 then (0,x+y)
   d. if x+y >4 and y>0 then (4,x+y-4)
      if x+y >3 and x>0 then (x+y-3,3)
5. the tree of the problem:

ملاحظة: عند عملية الوصف للمشكلة داخل الـknowledge base تكون هناك انتقالية واحدة فقط في كل مرة والتي تقوم بإيجاد حالة جديدة واحدة فقط أما الحالات التي تكون مشابهة للحالات السابقة أو حالة الـinitial فسوف تهمل.



ex (2):

Initial state goal state

Sol:

1. parameter from x1,x2,x3,…,x7.
2. the value of the parameter x1,x2,x3,…,x7=1 or x=0.
3. initial state = AAA-BBB goal state= BBB-AAA.
4. the rules of problem:
   a. A _ ⇒ _ A
   b. _ B ⇒ B _
   c. A B _ ⇒ _ B A
   d. _ A B ⇒ B A _
5. the solution space is:

initial state AAA__BBB

a	AA__ABBB
d	AABA__BB
b	AABAB__B
c	AAB__BAB
c	A__BABAB
a	__ABABAB
d	BA__ABAB
d	BABA__AB
d	BABABA__
a	BABAB__A
c	BAB__BAA
c	B__BABAA
b	BB__ABAA
d	BBBA__AA

a **Goal state BBB__AAA**

Ex(4):-The tower of (Hanoi) problem:-

Initial $\begin{pmatrix} C1 \\ B1 \\ A1 \end{pmatrix}$ goal $\begin{pmatrix} C3 \\ B3 \\ A3 \end{pmatrix}$

المسموح للمشكلة :-

1- توجد حركة واحدة في كل وقت.

2- لايمكن للحلقة الكبيرة تصبح فوق الصغيرة.

المطلوب:-

1- state space

2-احتمال اكثر من حل.

Legal move

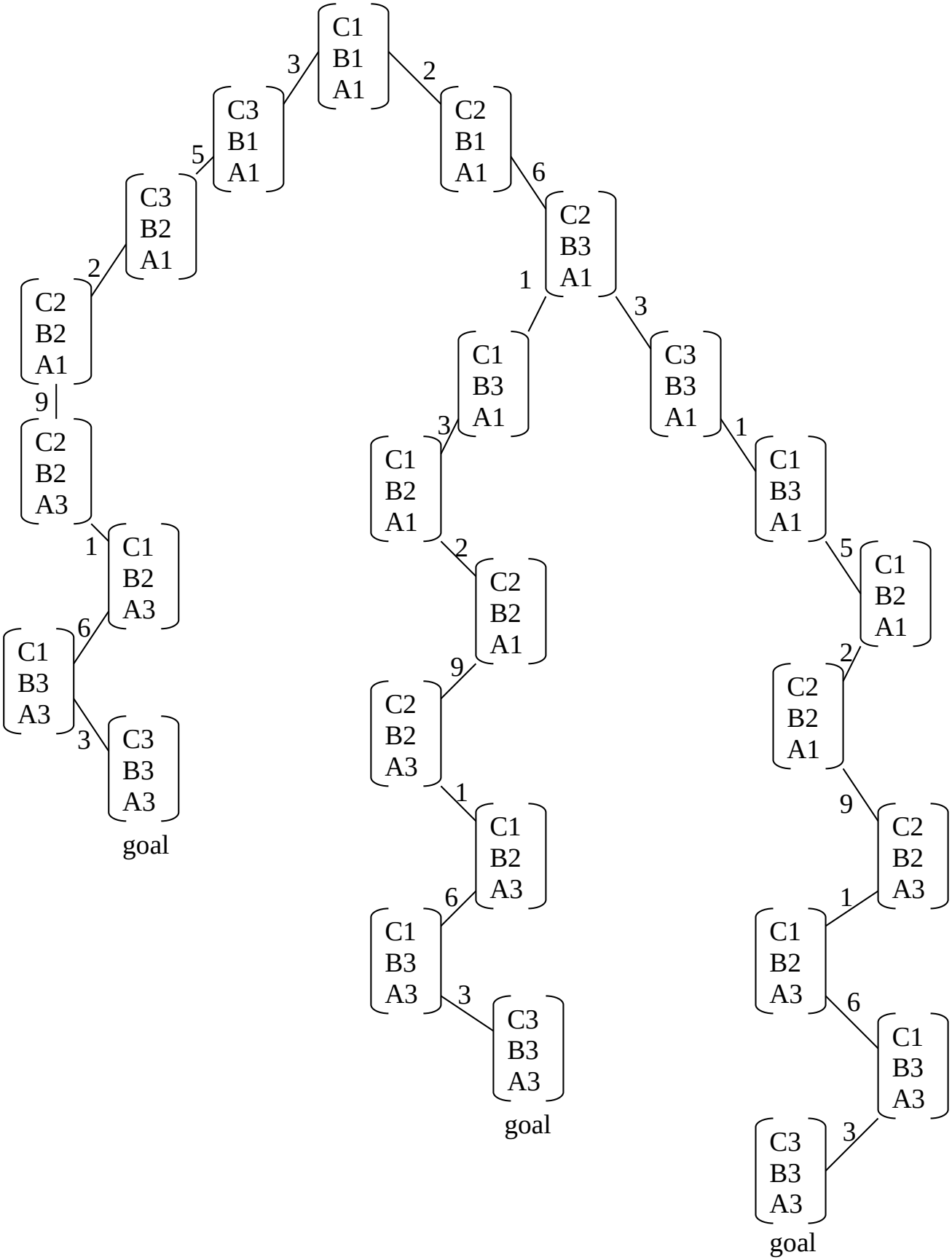
1-move(C,I). 4-move(B,I) 7-move(A,I)

2-move(C,II) 5-move(B,II) 8-move(A,II)

3-move(C,III) 6-move(B,III) 9-move(A,III)

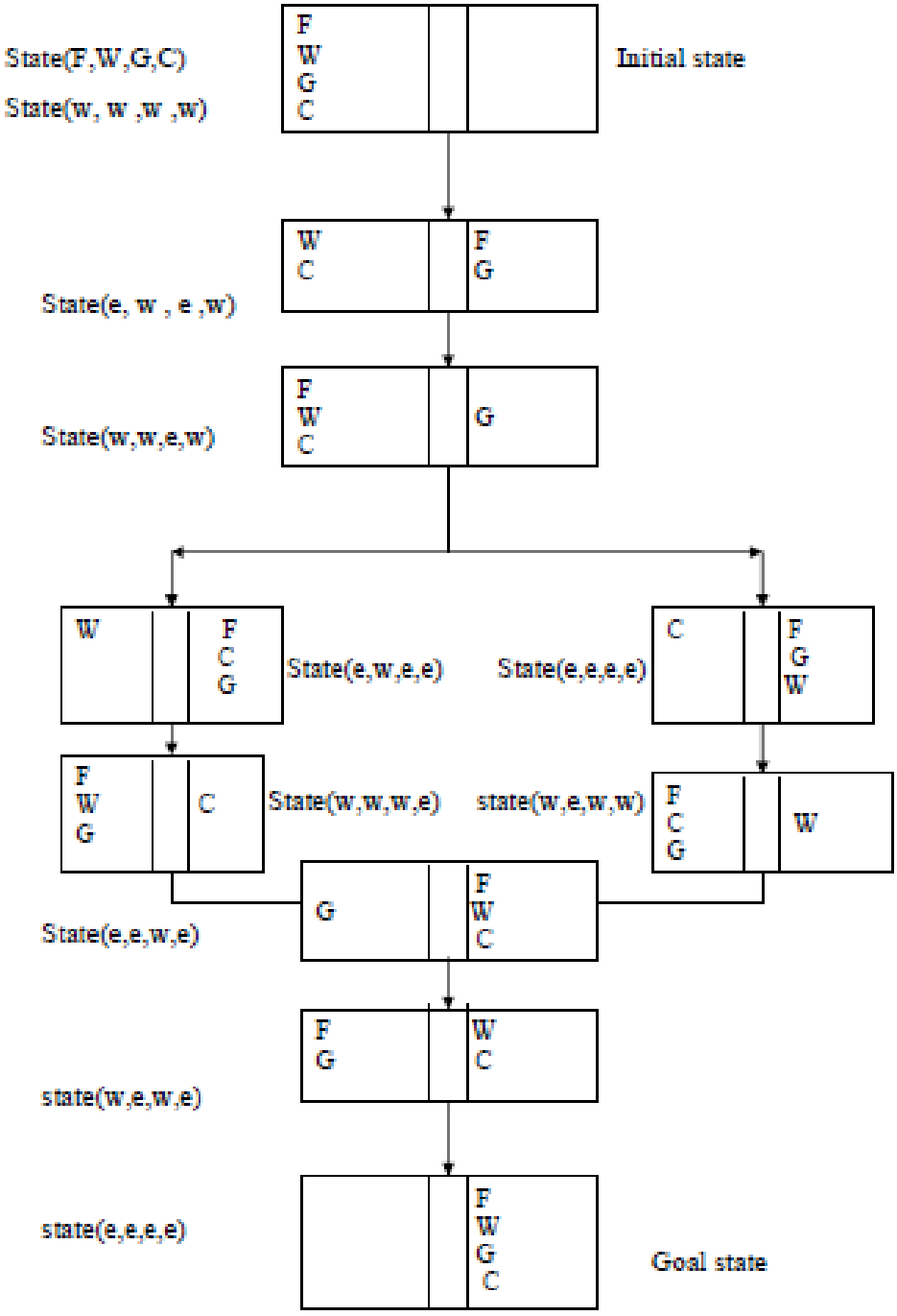
Tower of Hanoi problem

There are three tower (I,II,III) and 3 disk or ring of different size(A,B,C).The disk have hold in theft center so that they can stacked on the towers .Initially the three disk in are one tower I, the largest disk A is on the bottom, and the middle size disk B in the middle and the smallest disk C is on the top .It is desired to transfer the three disk to tower III by moving disk one at a time. Only the top disk on the tower can be moved ,but it can never be placed on top of smaller disk .The initial and goal state configuration are shown above


C1 B1 A1
C3 B1 A1
C2 B1 A1
C3 B2 A1
C2 B3 A1
C2 B2 A1
C1 B3 A1
C3 B3 A1
C2 B2 A3
C1 B2 A1
C1 B2 A3
C1 B3 A3
C3 B3 A3
goal
C2 B2 A1
C2 B2 A3
C1 B2 A3
C1 B3 A3
C3 B3 A3
goal
C1 B3 A1
C2 B2 A1
C1 B2 A1
C2 B2 A3
C1 B2 A3
C1 B3 A3
C3 B3 A3
goal

Ex(4): ***The Farmer , Wolf, Goat and Cabbage Problem:-***

A farmer wants to move himself , a wolf , a goat and some cabbage across a river. Unfortunately his boat is soting , the farmer can take only one of his possession across any trip
worse yet, an attended wolf will eat a goat, and and attended gaot will eat cabbage , so the farmer not leave the wolf alone with goat or the goat alone with the cabbage. What he is to do?


State(F,W,G,C)
State(w, w ,w ,w)
F W G C
Initial state
W C
F G
State(e, w , e ,w)
F W C
G
State(w,w,e,w)
W
F C G
State(e,w,e,e)
State(e,e,e,e)
C
F G W
F W G
C
State(w,w,w,e)
state(w,e,w,w)
F C G
W
G
F W C
State(e,e,w,e)
F G
W C
state(w,e,w,e)
F W G C
state(e,e,e,e)
Goal state

Control or search strategy:

هي الأساليب المتبعة للوصول إلى الهدف والتي تستخدم في طرق البحث، وتكون على ثلاثة أنواع:

1. **backward chaining (goal driven)**

goal → data (يبدأ من الهدف وينتهي بالبيانات)

2. **forward chaining (data driven)**

data → goal

3. **hybrid** (1&2) تجمع بين الطريقتين السابقتين ولكي تتحقق القاعدة يجب توفر جميع الشروط في وقت واحد

* (,),(A)⇒(and)

* (;),(Λ)⇒(or)

* في شجرة لفرع معين (مسار معين) لا يجوز تكرار اكثر من عقدة اكثر من مرة واحدة

Ex(1):

suppose we have the following rules & facts, prove a:

لحل أي سؤال يجب إعطاء ثلاث عناصر وهي القواعد والحقيقة والهدف.

rules a:- b, c fact: u

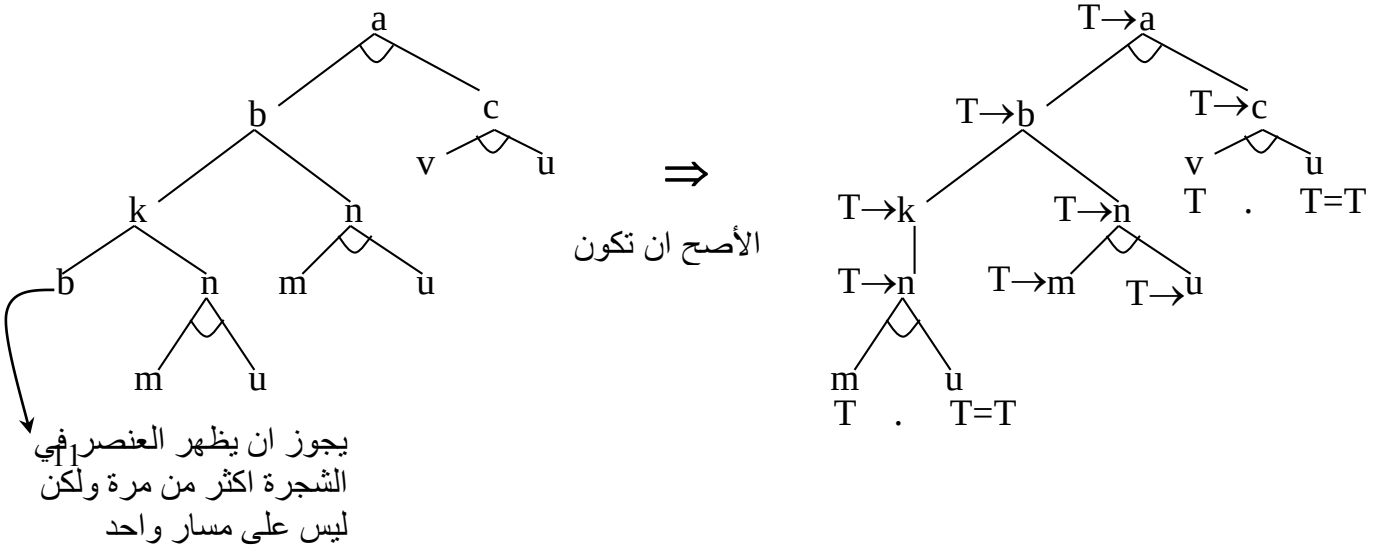
b:- k; n v } always true

c:- v, u m

n:- m, u

k:- b; n

1) backward chaining:



2) forward chaining: (نفس المثال السابق)

fact: u, v, m

step1: use u	step2: use v	fact: u
a:- b, c	a:- b, c	v
b:- k; n	b:- k; n	m
c:- v	n:- m	c
n:- m	k:- b; n	
k:- b; n		

step3: use m	fact: u	step4: use c
a:- b, c	v	a:- b
b:- k; n	m	b:- k; n
k:- b; n	c	k:- b; n
	n	

step5: use n	fact: u	step6: use b
a:- b	v	[ ]
	m	
	c	
	n	
	b	
	k	

∴ a is true

3) hybrid: (نفس المثال السابق)

fact: u, v, m

step1: use u,v,m	step2: use u,v,m,c,n	step3: use u,v,m,c,n,b,k
a:- b, c	a:- b, c	[ ]

b:- k; n

∴ a is true

k:- b; n

fact: u, v, m, c, n

Ex(2):

suppose you have the following rules & facts, prove goal:

facts: fact1, fact2

rules:

goal:- goal1; goal2

goal1:- a, b

goal2:- c(x)

a:- ¬ d

b:- e

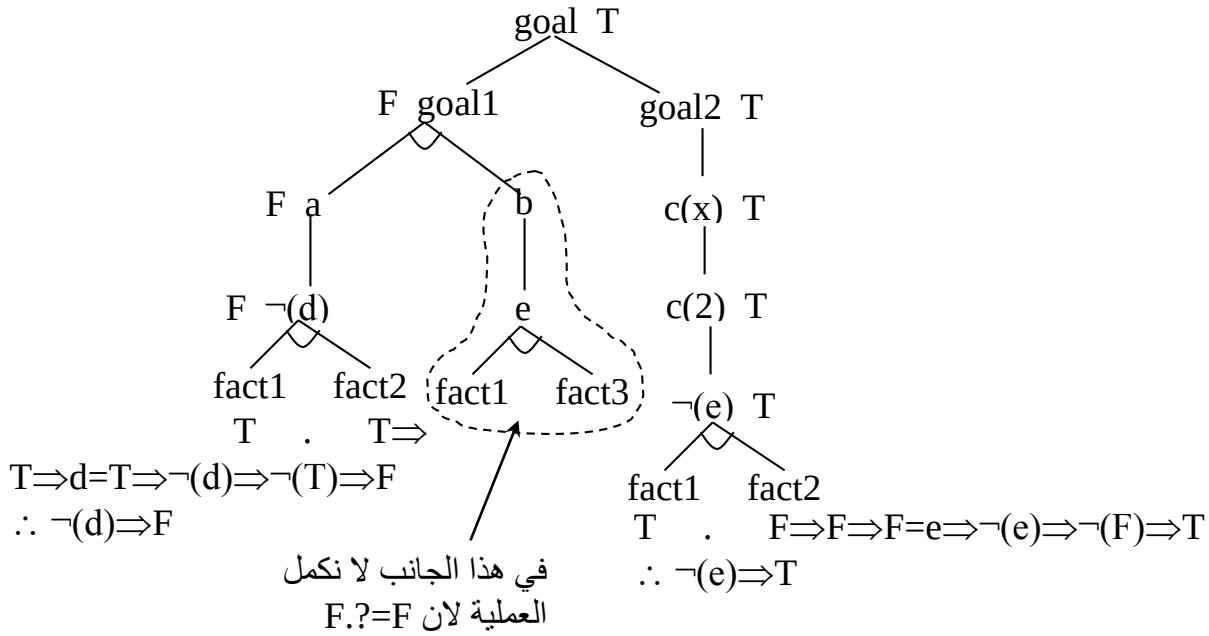
c(2):- ¬ e

d:- fact1, fact2

e:- fact1, fact3

sol:

1) backward chaining:



∴ Goal is true

2) forward chaining:

fact: fact1, fact2

ملاحظة: x متغير

step1: use fact1	step2: use fact2	step3: use d
goal:- goal1; goal2	goal:- goal1; goal2	goal:- goal1; goal2
goal1:- a, b	goal1:- a, b	goal1:- a, b
goal2:- c(x)	goal2:- c(x)	goal2:- c(x)
a:- ¬ d	a:- ¬ d	a:- ¬ d أي خطوة فيها نفي نتركها
b:- e	b:- e	b:- e
c(2):- ¬ e	c(2):- ¬ e	c(2):- ¬ e
d:- fact2 فقط نحذف fact1	e:- fact3	e:- fact3
e:- fact3	fact: fact1, fact2, d	

* في الخطوة الثالثة لم تنتج أي شئ لذا نقوم بخطوة أخرى تعتمد على النفي

Step4: use negation	step5: use c(2)	step6: use goal2
goal:- goal1; goal2	goal:- goal1; goal2	goal1:- a, b
goal1:- a, b	goal1:- a, b	goal2:- c(x)
goal2:- c(x)	goal2:- c(x)	a:- ¬ d
a:- ¬ d	a:- ¬ d	b:- e
b:- e	b:- e	e:- fact3
e:- fact3	e:- fact3	∴ Goal is true
fact: fact1, fact2, d, c(2)	fact: fact1,fact2,d,c(2),goal2	

* في الخطوة الرابعة F= ¬d إذن لا تحذف، ¬e -:c(2) =T إذن تحذف وتصبح حقيقة.

* في الخطوة الخامسة goal2:- c(x) لا تحذف هذه القاعدة لأنها تحوي على متغير بالرغم من تحققها (T) ولكن تأخذ كحقيقة.

3) hybrid

fact: fact1, fact2

step1: use fact1,fact2

goal:- goal1; goal2

goal1:- a, b

goal2:- c(x)

a:- ¬ d

b:- e

c(2):- ¬ e

e:- fact1, fact3

step2: use fact1,fact2,d

goal:- goal1; goal2

goal1:- a, b

goal2:- c(x)

a:- ¬ d

b:- e

c(2):- ¬ e

e:- fact1, fact3

لم تنتج هذه الخطوة أي شئ

* في الخطوة الأولى نحذف الـd لان T.T=T ولا نحذف الـe لان T.F=F

step3: use negation

goal:- goal1; goal2

goal1:- a, b

goal2:- c(x)

a:- ¬ d

b:- e

e:- fact1, fact3

Step4: use fact1,fact2,d,c(2)

goal:- goal1; goal2

goal1:- a, b

goal2:- c(x) تحققت ولكن لا تحذف

a:- ¬ d

b:- e

e:- fact1, fact3

fact: fact1, fact2, d, c(2)

step5: use fact1,fact2,d,c(2),goal2

goal1:- a, b

goal2:- c(x)

a:- ¬ d

b:- e

e:- fact1, fact3

∴ Goal is true

search in AI:

search: study the methods (ways) of searching with the knowledge base and comparison between them how to reach the goal with less time and cost.

دراسة لطرق البحث مع قاعدة المعرفة بالمقارنة بينهم ايهم تصل للهدف باقل وقت واقل كلفة.

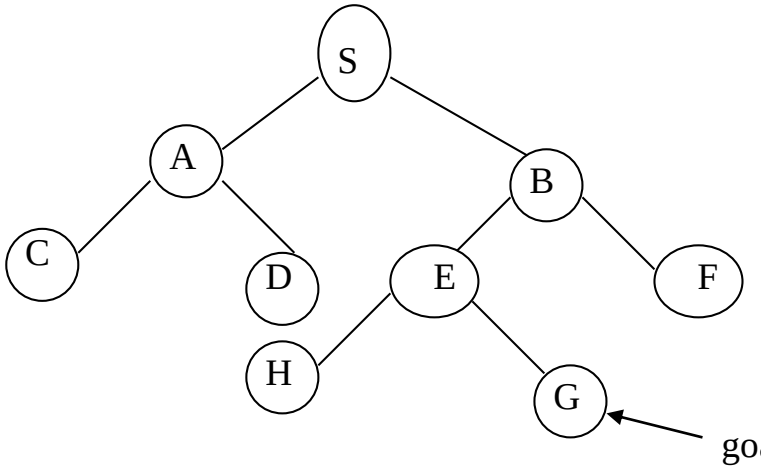
1)Blind search: البحث الأعمى

1. **breadth first search**:

```
procedure breadth first search;
begin
  open := [start];
  closed :=[];
  while open = [] do
     begin
       remove leftmost state from open, call it X;
         if x is a goal then return (success)
           else begin
                    generate children  of X ;
                    put X on close;
                     eliminate children of X on open or close;
                      put remaining children on right end of open
                end;
        return(failure)
       end.
```

Ex(1):-see the tree and find the goal by the breadth first search?

أيجاد الهدف او طريقة البحث الـ breadth أي طريقة البحث الافقي.



ملاحظة:- نقوم يوضح جدولين الأول open والثاني close ونفحص أول عقده إذا كانت الهدف آم لا. إذا كانت الهدف أولا نحذف العقدة من قائمة الـ open ونظيف أبناء العقد الجديدة إلى قائمة الـ open وتكون عملية الإضافة في نهاية قائمة الـ open .

Path === S ⟶ A ⟶ B ⟶ C ⟶ D ⟶ E ⟶ F ⟶ H ⟶ G

الباث يؤخذ من قائمة الـ CLOSE

Open	Close
S	--------
A B	S
B C D	A
C D E F	B
D E F	C
E F	D
F H G	E
H G	F
G	H

(G) IS THE GOAL

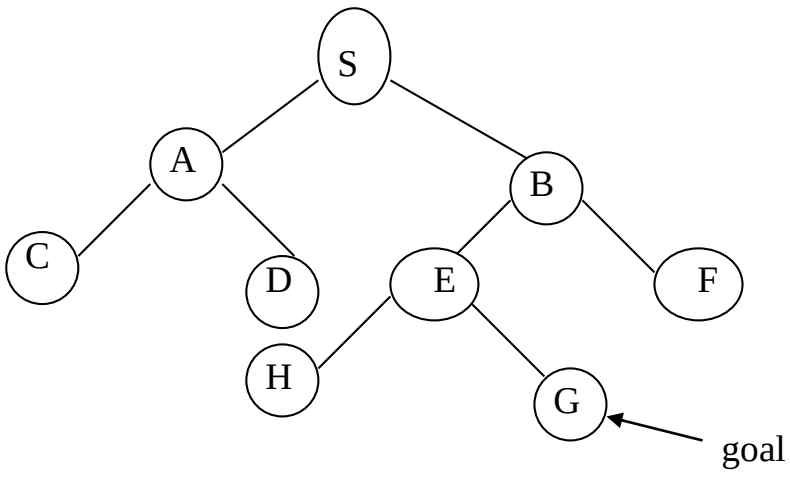
ملاحظة :- في هذه الخوارزمية لا يوجد الـ backtracking لان عملية البحث تكون level by level.

2-**Depth first search**:

```
procedure depth first search;
begin
  open := [start];
  closed :=[];
  while open = [] do
     begin
       remove leftmost state from open, call it X;
         if X is a goal then return (success)
           else begin
                  generate children  of X ;
                  put X on close;
                   eliminate children of X on open or close;
                    put remaining children on left end of open
                end;
        return(failure)
       end.
```

Ex(1):-see the tree and find the goal by the depth first search?

أيجاد الهدف او طريقة البحث الـ depth أي طريقة البحث العمودي.



ملاحظة:- نقوم يوضح جدولين الأول open والثاني close ونفحص أول عقده إذا كانت الهدف آم لا. إذا كانت الهدف أولا نحذف العقدة من قائمة الـ open ونظيف أبناء العقد الجديدة إلى قائمة الـ open وتكون عملية الإضافة في بداية قائمة الـ open .

Path === S ⟶ A ⟶ C ⟶ D ⟶ B ⟶ E ⟶ H ⟶ G

الباث يؤخذ من قائمة الـ CLOSE

Open	Close
S	--------
A B	S
C D B	A
D B	C
B	D
E F	B
H G F	E
G F	H

(G) IS THE GOAL

ملاحظة :- اختيار أي طريقة يعتمد على موقع الهدف إذا كان الهدف قريب من حالة البداية التي هي initial state نختار البحث الأفقي breadth first search وإذا كان الهدف بعيد عن نقطة البداية نختار البحث العمودي depth first search وتعتمد على الخبرة.

The advantages of depth first search:-

1. D.F.S. requires less memory since only the nodes on the current path are stored. This contrasts with B.F.S. where all of the tree that has so far been generated must be stored.
2. D.F.S. may find a solution without examining much of the search space at all. This contrasts with B.F.S. in which all parts of the tree must be examined to level N before any nodes on level(n+1) can be examined this particularly significant if many acceptable solution exist, D.F.S. can stop when one of them is found.

ميزات طريقة البحث العمودي:-

1. d.f.s. يحتاج إلى ذاكره اقل لأنه يخزن العقد التي في طريقه الحالي وعند مقارنته مع b.f.s. فأنه يقوم بخزن الشجرة المولدة كأمله.
2. d.f.s. من الممكن أن يجد الحل بدون اختبار الكثير من فضاء البحث. بالمقارنة مع b.f.s. الذي يجب أن تكون كل أجزاء الشجرة ممتحنة ابتداءا" من العقدة n إلى العقده n+1 أي ان d.f.s. يتوقف أينما يجد الهدف.

The advantages of B.F.S:-

1. B.F.S. will not get trapped exploring a blind only .this contrasts with D.F.S. ,which may follow a single, unfruitful path for every long time. this is a particular problem in D.F.S. if there are loops unless special care is expanded to test for such a situation.
2. if there is a solution , then B.F.S. is guaranteed to find it, further more if there are multiple solutions, then a minimal solution (minimum number steps) will be found ,this is guarantied by the fact that longer path are never explored until shorter ones have already been examined.

فوائد البحث الافقي B.F.S. :-

1. b.f.s. لا تقع ضمن مكيدة أو مصيدة البحث الأعمى بالمقارنة مع d.f.s. يتبع طريق وحيد وغير مثمر ولفترة طويلة وإذا كانت هناك حلقات ما لم تأخذ بعناية وخصوصية تؤدي إلى توسع في الاختبار للوصول إلى الحلول.
2. اذا كان هناك حل اذن B.F.S. سوف يضمن لنا ان نجد الحل ، واكثر من ذلك اذا كان هناك حلول متعددة اذن اقل الحلول (اقل عدد من الخطوات) سوف نجدها. هذا سوف يضمن بالحقيقة بان مسارات أطول لم تكتشف طالما كل المسارات القصيرة كانت مختبرة مسبقا.

2. **Heuristic searching**: البحث حسب الأفضلية (البحث التنقيبي)

a) **Hill-climbing**:

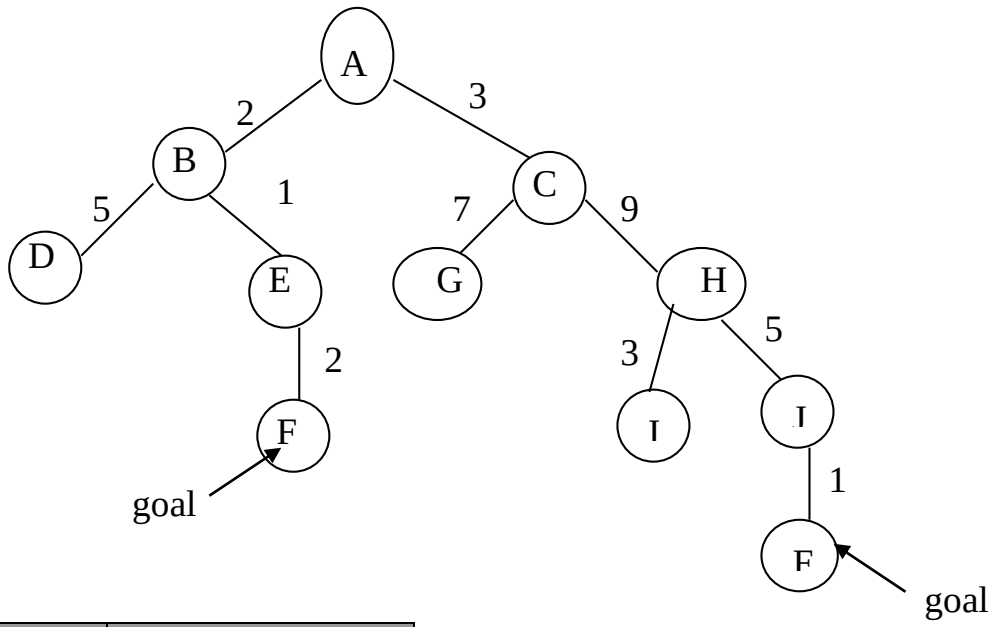
وهذه الطريقة مشابهة تماما لطريقة الـdepth الا انها تختلف عنها فقط بان الاختيار لا يكون من الجهة اليسرى او اليمنى وانما حسب المعيار الاعلى قيمة وعند الدخول في فرع معين من الشجرة فاننا نبقى في ذلك الفرع لحين الانتهاء من جميع العقد التي يحتويها لذلك هو مشابه للـdepth.

* تعتبر من اقوى الخوارزميات في مجال البحث التنقيبي واخذت الفكرة من متسلق الجبل الاعمى الذي يملك مجموعة من المعلومات التي تساعده في اختيار طريقة الوصوا الى القمة وتعتمد الخوارزمية على دالة التنقيب (heuristic function)

```
Hill Climbing Algorithm
Begin
Cs=start state;
Open=[start];
Stop=false;
Path=[start];
While (not stop) do
{
if (cs=goal) then
return (path);
generate all children of cs and put it into open
if (open=[]) then
stop=true
else
{
x:= cs;
for each state in open do
{
compute the heuristic value of y (h(y));
if y is better than x then
x=y
}
if x is better than cs then
cs=x
else
stop =true;
}
}
return failure;
```

}

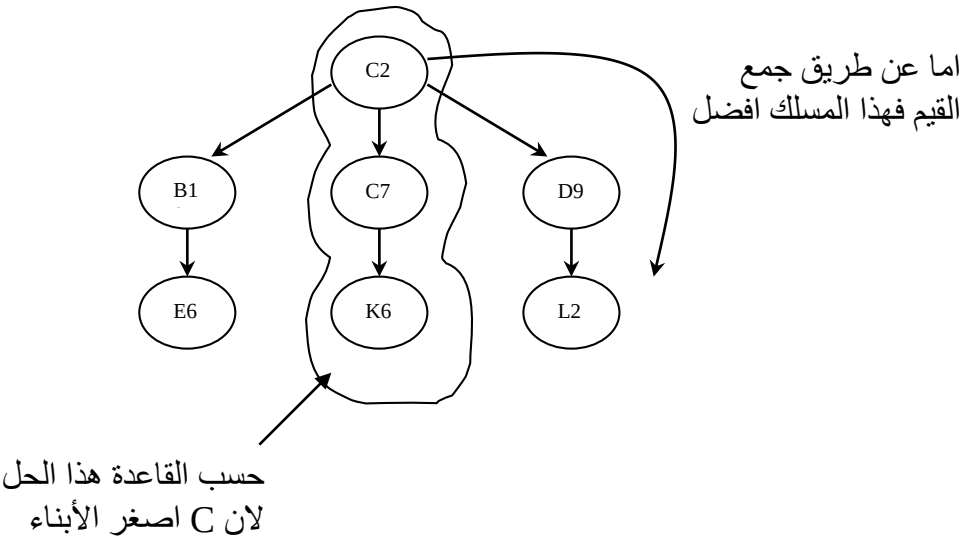
مثال :



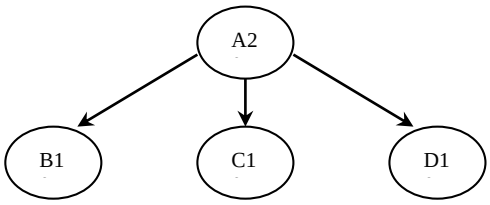
state	Open	close
A	[A]	[ ]
C	[CB]	[A]
H	[HGB]	[CA]
J	[JIGB]	[HCA]
F	[FIGB]	[JHCA]
I	[IGB]	[***F***JHCA]
G	[GB]	[IFJHCA]
B	[B]	[GIFJHCA]
D	[DE]	[BGIFJHCA]
E	[E]	[DBGIFJHCA]
F	[F]	[EDBGIFJHCA]
stop	[ ]	[***F***EDBGIFJHCA]

Disadvantage of hill climbing:

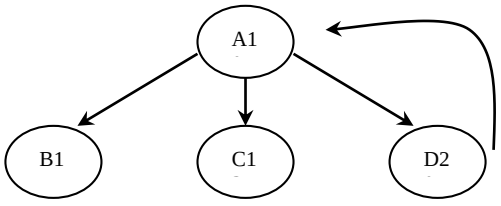
1. local maximum: is a state that is better that all it's neighbors but is not better that some of her status further way. صحيح انه احسن من البقية لكنه في العمق سيكون اكثر من الباقين أي معناه اذا اتبعنا القاعدة سيكون الناتج اكبر من عدم اتباع القاعدة



2. flate area : is a flate area of search space in which a whole set of neighboring states have the same value. أي معناه في هذه الطريقة عند النزول الى المستوى الثاني تكون الارقام متساوية ففي مثل هذه الحالة نعمل قفزة لمستوى واحد أي الى الابناء ونرى ما هو الاقل قيمة فيكون هو الناتج



3. ridage: is a special kind of local maximum it's an area of the search space that is higher than surrounding area. معناها الارتطام بصخرة وهي حالة خاصة (الابناء اكبر قيمة من الابناء)



there are some ways of dealing with these problem:

1. backtracking to earlier node and try going in a different direction. نرجع الى الاب ونبدأ بالبحث من جديد للوصول الى الحل الامثل كما في الحالة الاولى
2. make a big jump in some direction to try to get a new direction of the search. اذا كان الابناء متساوين في القيمة جميعهم فنعمل قفزة للوصول الى الحل
3. apply tow or more rules before going to test. نضيف قاعدة جديدة الى الشجرة السابقة لانها لاتحل باي طريقة سابقة

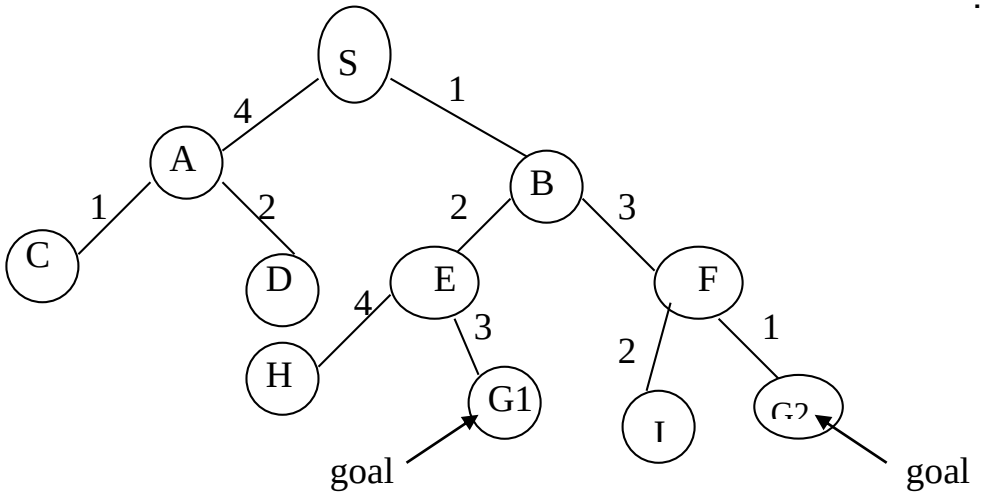
b) **Least cost**:

طريقة عكس الـhill-climbing.

ملاحظة:

* يتم في هذه الطرقة حساب الكلفة بتجميع كلفة المسار موصولا الى الهدف ويتم ترتيب هذه العقد او القيم في مصفوفة الـopen تصاعديا وذلك باختيار اول عقدة وفحصها اذا كانت الهدف ام لا وعند عدم ايجاد الهدف تحسب كلفة الاولاد ثم ترتب عناصر المصفوفة مع الاولاد الجدد تصاعديا لحين ايجاد الهدف.

مثال :



Open	
S(0)	
B(1) A(4)	كتبنا الـB اولا لان كلفتها اقل وهي 1
E(3) A(4) F(4)	
A(4) F(4) G1(6) H(7)	جاءت قيمة G1(6) من جمع ارقام المسار من نقطة البداية وصولا للنقطة الطلوبة
F(4) C(5) G1(6) D(6) H(7)	يتم الترتيب حسب الارقام من الاصغر الى الاكبر
C(5) G2(5) G1(6) D(6) I(6) H(7)	حذفت الـC(5) لان ليس لديها ابناء
G2(5) G1(6) D(6) I(6) H(7)	في هذه المرحلة يتم التوقف لانه وصلنا الى الهدف وهو G2 واصبح في البداية في الخطوات السابقة لم يتم التوقف بالرغم من احتوائها على الهدف لانه ليس بالبداية

c) **Best –first search**:

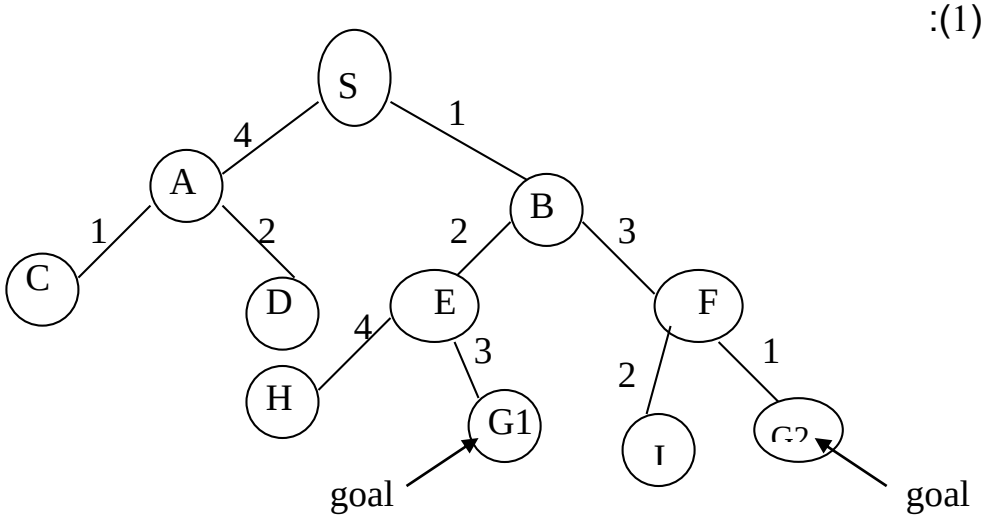
هذه الطريقة هي لحل مساوئ الـdepth & breadth ولكن وفق معيار معين وغالبا ما يكون الاقل قيمة.

ملاحظة: عندما ندخل الابناء الى الـopen ندخلهم بالترتيب حسب المعيار المذكور مثلا حسب الاقل قيمة وبعد الإدخال بذلك الترتيب نقوم بترتيب الابناء الذين ادخلوا مع ما موجود في الـopen حسب الاقل قيمة.

```
Function best-first search
Begin
  Open=[start];
  closed :=[];
     while open <> [] do
        begin
          remove leftmost state from open, call it X;
            if X = goal then return the path from start to X
              else begin
                      generate children  of X ;
                      for each chilled of X do
```

```
                    case
                      the chilled is not open or close
                       begin
                          assign the chilled a heuristic value
                          add the chilled to open
                       end;
                    the chilled is already on open
                    if chilled was reached by shorter path then
                    given the state on open the shorter path
                    the chilled is already on close
                    if the chilled was reached by the shorter path then
                    begin
                      remove the state from close;
                      add the chilled to open
                     end;
                  end case;
              put X on close;
              re order on state open by heuristic value
          end;
           return(failure)
    end;
```

مثال (1):



Open	
S(0)	
B(1) A(4)	
E(2) F(3) A(4)	
G1(3) F(3) A(4) H(4)	First goal
F(3) A(4) H(4)	
G2(1) I(2) A(4) H(4)	second goal

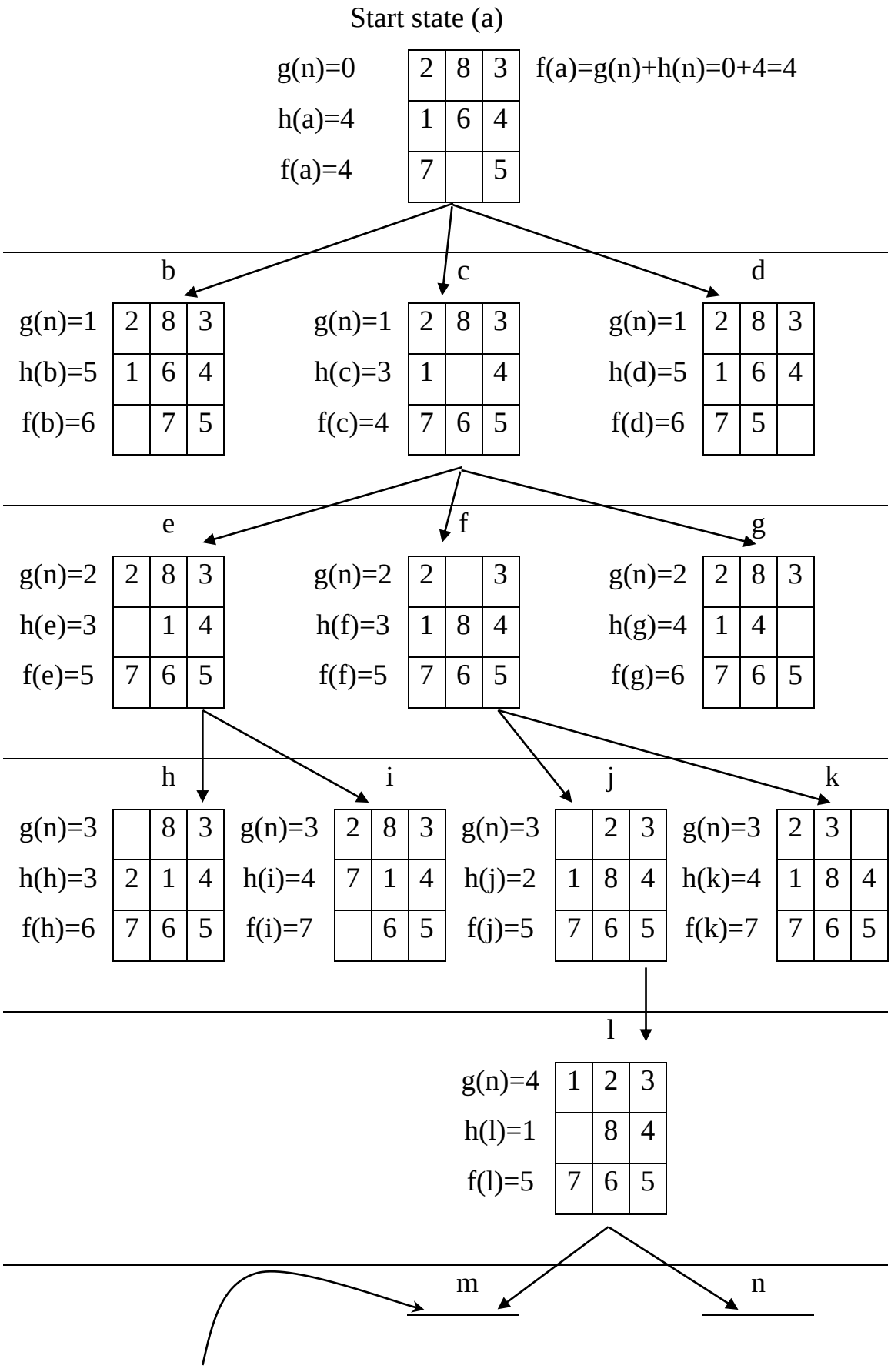
d) **A_ Algorithm**

best-first search & heuristic search

EX:

- g(n)=actual distance from n to the start state.بعد الخطوة عن البداية
- g(n)=start state.معناه ما هو العدد الذي يبعده عن حالة البداية
- h(n)=number of tiles out of palce. معناه أي واحدة من الارقام ليس في مكانه في كل مربع
- f(n)=g(n)+h(n)

ملاحظة: الفراغ لايحسب ولا يجوز تكرار الحالات.


Start state (a)
g(n)=0
h(a)=4
f(a)=4
2 8 3
1 6 4
7 5
f(a)=g(n)+h(n)=0+4=4
b
c
d
g(n)=1
h(b)=5
f(b)=6
2 8 3
1 6 4
7 5
g(n)=1
h(c)=3
f(c)=4
2 8 3
1 4
7 6 5
g(n)=1
h(d)=5
f(d)=6
2 8 3
1 6 4
7 5
e
f
g
g(n)=2
h(e)=3
f(e)=5
2 8 3
1 4
7 6 5
g(n)=2
h(f)=3
f(f)=5
2 3
1 8 4
7 6 5
g(n)=2
h(g)=4
f(g)=6
2 8 3
1 4
7 6 5
h
i
j
k
g(n)=3
h(h)=3
f(h)=6
8 3
2 1 4
7 6 5
g(n)=3
h(i)=4
f(i)=7
2 8 3
7 1 4
6 5
g(n)=3
h(j)=2
f(j)=5
2 3
1 8 4
7 6 5
g(n)=3
h(k)=4
f(k)=7
2 3
1 8 4
7 6 5
l
g(n)=4
h(l)=1
f(l)=5
1 2 3
8 4
7 6 5
m
n

goal

g(n)=5	1	2	3
h(m)=0	8		4
f(m)=5	7	6	5

g(n)=5	1	2	3
h(n)=2	7	8	4
f(n)=7		6	5

Implementing Heuristic Evaluation Function:-

من مساوئ خوارزمية البحث عن الافضل (Best – First – Search) انها تركز عملية البحث على فرع في فضاء البحث على حساب بقية الافرع . ويحدث ذلك اذا استمر الاقتران الاجتهادي (كلفة ال node) في اعطاء كلف او قيم اجتهادية جيدة للحالات الموجودة على فرع ما. وقد يكون ذلك سيئا لعدم وجود الحالة الهدفية على ذلك الفرع من فضاء البحث. او بسبب وجود حالة هدفية اقرب (ذات مسار اقصر) على فرع اخر.

ولتجنب هذه المشكلة ينصح عادة باستخدام اقتران اجتهادي (Heuristic function) تأخذ بنظر الاعتبار مقدار بعد الحالة المختارة عن حالة البداية بحيث تفضل الحالة الأقرب. وذلك لتجنب الاستمرار في البحث في فرع واحد على حساب بقية الافرع . وستكون Heuristic Function لها الصيغة التالية:-

$$Fn = g(n) + h(n)$$

g(n):- Measures the actual length of path from any state (n) to the start state.

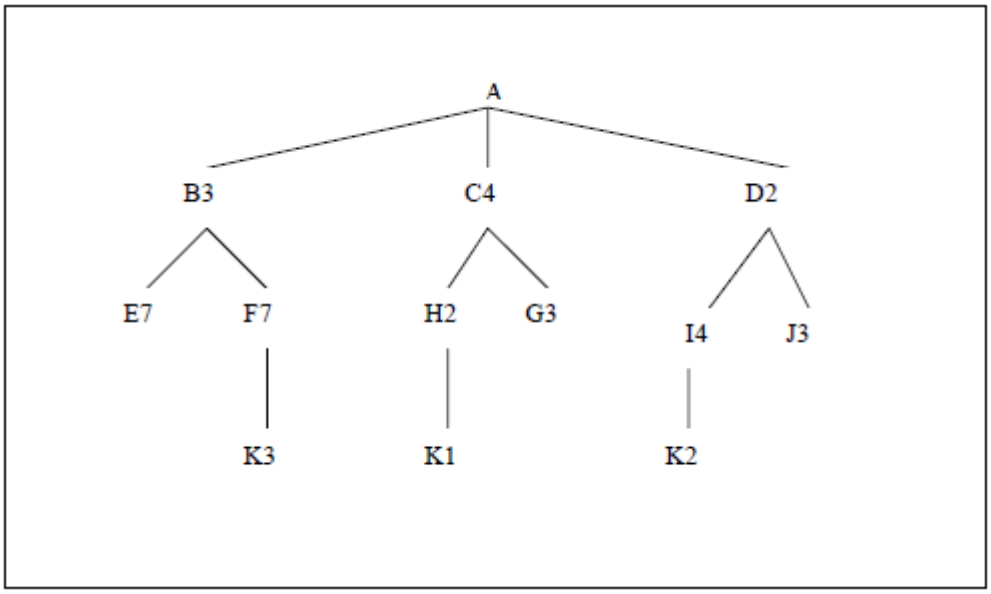
H(n):- is a heuristic estimate of the distance from state n to the goal.

- عندما تستخدم خوارزمية Best-First مع Heuristic Function كما في الصيغة (١) فان الخوارزمية الناتجة تسمى A- Algorithm.

A Algorithm

Trace of A- Algorithm (Search

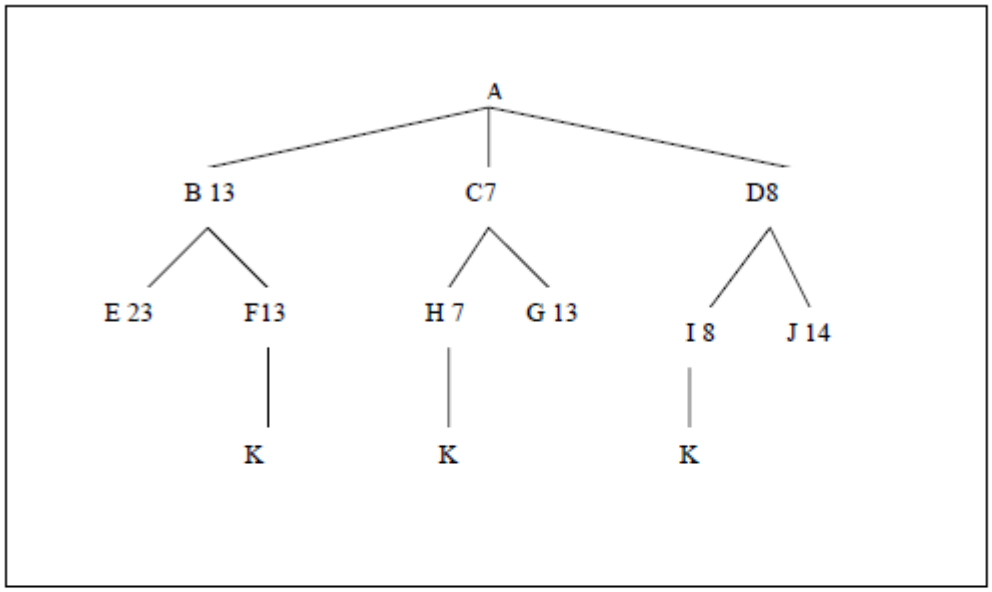
Connect(a,b,3)
Connect(a,c,4)
Connect(a,d,2)
Connect(b,e,7)
Connect(b,f,7)
Connect(c,g,3).
Connect(c,h,2).
Connect(d,I,4)
Connect(d,j,3).
Connect(f,k,3).
Connect(h,k,1).
Connect(I,k,2).


A
B3
C4
D2
E7
F7
H2
G3
I4
J3
K3
K1
K2

The tree after evaluation function calculation is :
Connect(a,b,13) Connect(a,c,7) Connect(a,d,8) Connect(b,e,23)

Connect(b,f,13) Connect(c,g,14). Connect(c,h,7). Connect(d,I,8)

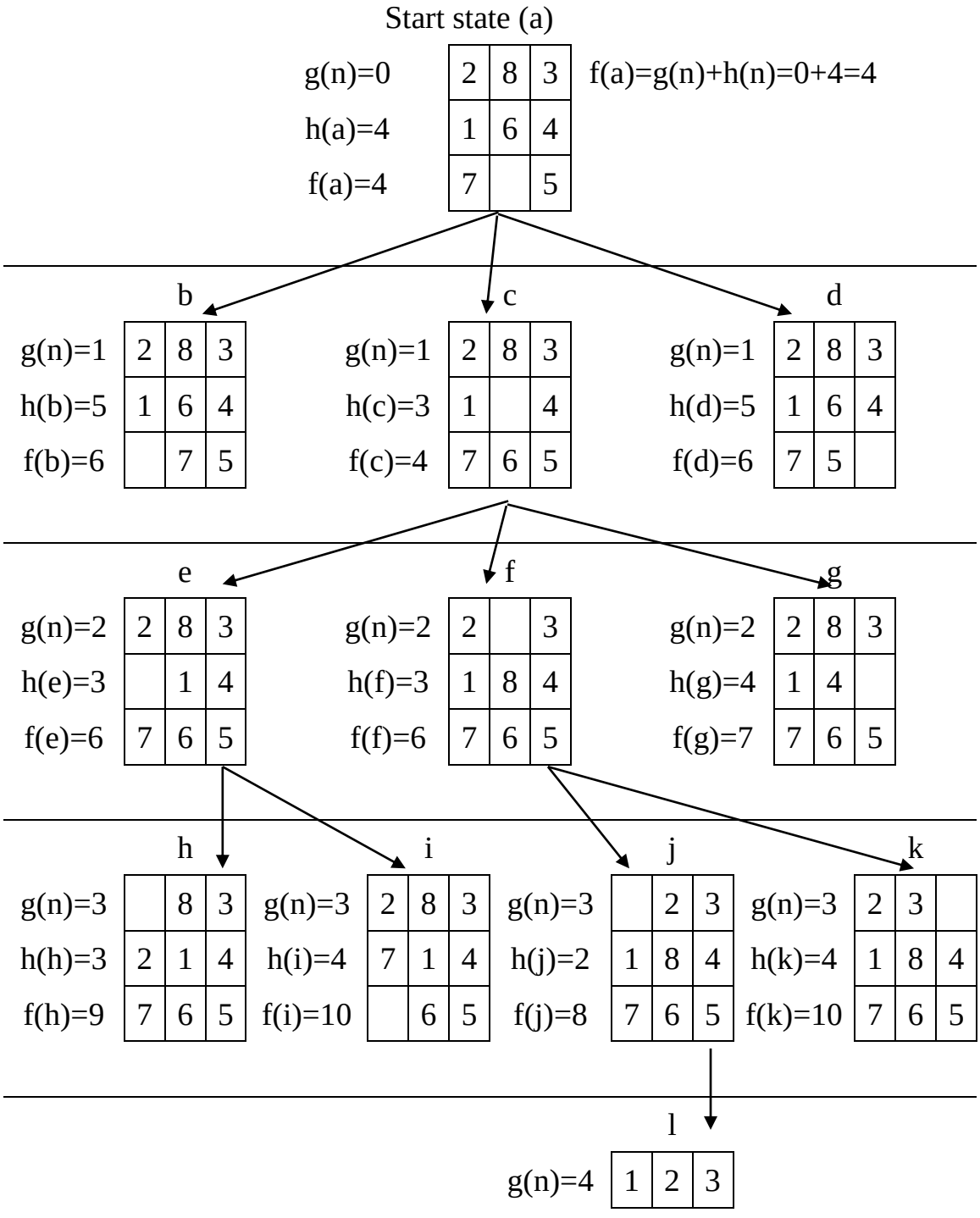
Connect(d,j,13). Connect(f, k). Connect(h, k). Connect(I, k).

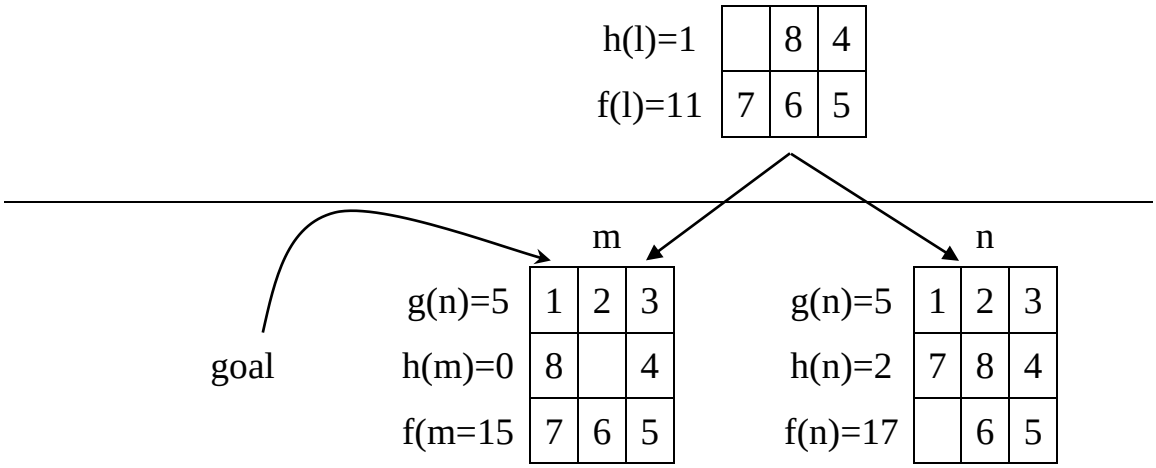


Open	Close
[A]	[ ]
[C7 D8 B13]	[A]
[H7 D8 B13 G13]	[C7 A]
[K D8 B13 G13]	[H7 C7 A]
	[K H7 C7 A]

The path is : A-C7- H7- K

A*-Algorithm


Start state (a)
g(n)=0
h(a)=4
f(a)=4
2 8 3
1 6 4
7 5
f(a)=g(n)+h(n)=0+4=4
b
g(n)=1
h(b)=5
f(b)=6
2 8 3
1 6 4
7 5
c
g(n)=1
h(c)=3
f(c)=4
2 8 3
1 4
7 6 5
d
g(n)=1
h(d)=5
f(d)=6
2 8 3
1 6 4
7 5
e
g(n)=2
h(e)=3
f(e)=6
2 8 3
1 4
7 6 5
f
g(n)=2
h(f)=3
f(f)=6
2 3
1 8 4
7 6 5
g
g(n)=2
h(g)=4
f(g)=7
2 8 3
1 4
7 6 5
h
g(n)=3
h(h)=3
f(h)=9
8 3
2 1 4
7 6 5
i
g(n)=3
h(i)=4
f(i)=10
2 8 3
7 1 4
6 5
j
g(n)=3
h(j)=2
f(j)=8
2 3
1 8 4
7 6 5
k
g(n)=3
h(k)=4
f(k)=10
2 3
1 8 4
7 6 5
l
g(n)=4
1 2 3



F*(node)=h*(node)+g*(node)+g*(node to the start)

F*(a)=4+0+0=4

F*(b)=5+1+0=6

F*(c)=3+1+0=4

F*(d)=5+1+0=6

F*(e)=3+2+1=6

F*(f)=3+2+1=6

F*(g)=4+2+1=7

F*(h)=3+3+2+1=9

F*(i)=4+3+2+1=10

F*(j)=2+3+2+1=8

F*(k)=4+3+2+1=10

F*(l)=1+4+3+2+1=11

F*(m)=0+5+4+3+2+1=15

F*(n)=2+5+4+3+2+1=17

Best first search

Open close

A ------

C4B6D6 A

E6F6G7 C

F6 E

J8 K10 F

L11 J

M0F17 THIS IS GOAL

Production system أنظمة الإنتاج

Each production system consist of three elements

1- **PRODUCTION RULRES IF------THEN**
2- **WORKING MEMORY (WM)**
   It is the the place where the system work carnet description of problem .
3- **CONTROL STRATEGY OR RECOGNIZE ACT CYCLE**

ملاحظة:- في الحل يجب ان تكون الثلاث عناصر موجودة ، ممكن أن تكون أول نقطة غير متوفرة فيجب إيجادها قبل البدء بالحل.

EXAMPLE 1:-

Build construct production system for sorting string composed of a,b,c.

Rules

1- ba------→ab
2- ca-----→ac
3- cb-----→bc

W.M

cbaca

Control strategy

Cycle the rules in order until no more matched

Iteration	W.M	conflict set	firing
0	cbaca	1,2,3	1
1	cabca	2	2
2	acbac	1,3	1
3	acabc	2	2
4	aacbc	3	3
5	aabcc	halt	

EXAMPLE 2:-

Build construct production system for sorting string composed of a,r,w.

Rules

1- ra------→ar
2- wa-----→aw
3- wr-----→rw

W.M

rawaa

Control strategy

Cycle the rules in order until no more matched

Iteration	W.M	conflict set	firing
0	rawaa	1,2	1
1	arwaa	2	2
2	arawa	1,2	1
3	aarwa	2	2
4	aaraw	1	1
5	aaarw	halt	

Example 3-Knight-tour(حركة الفرس)

1	2	3
4	5	6
7	8	9

Rules

1- if 1 then 8
2- if 1 then 6
3- if 2 then 9
4- if 2 then 7
5- if 3 then 8
6- if 3 then 4
7- if 4 then 9
8- if 4 then 3
9- if 6 then 1
10- if 6 then 7
11- If 7 then 6
12- If 7 then 2
13- If 8 then 3
14- If 8 then 1

15- If 9 then 2

16- If 9 then 4

W.M

Start 1 Goal 2

Control Strategy

Cycle the rule in order without looping.

Iteration	W.M	conflict set	firing
0	1	1,2	1
1	8	13,14	13
2	3	5,6	6
3	4	7,8	7
4	9	15,16	15
5	2 goal		

Artificial Intelligence

SECOND COURSE

3RD CLASS

مدرس المادة

م. نادية محمد

2020-2021

Knowledge Representation:-تمثيل المعرفة

1- **Mathematical logic**:-

a-**prepositional logic**.

b-**predicate logic**.

2- **Rule**.

3- **Semantic network& conceptual graph**.

4- **Frame**

1- **Mathematical logic**:-المنطق الرياضي

a-**prepositional logic**. هي عملية ترميز العبارات واستنتاج عبارة جديده قد تكون صحيحه او خاطئة اعتمادا على صحة العبارات المستنتجة منها او خطأ هذه العبارات.

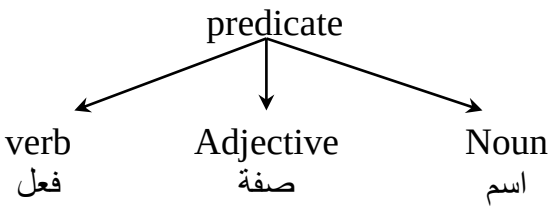
Ex:-

- the sky is cloudy.(sc)
- It is wet air.(wa)
- It is raining day.(rd)

Sc	Wa	SC^WA(RD)
T	T	T
F	F	F
T	F	F
F	T	F
حسب طبيعة العلاقة وحسب صحة الأقوال تعتمد النتيجة		النتيجة

b-**Predicate logic**:-

هو لفظ يعبر عن كلمة مكتوبة يمثل العلاقة ما بين عنصرين او يصف عنصر معين بتعبير اخر تحويل اللغة من الجملة الطبيعية (اللغة الطبيعية) الىlogic predicate



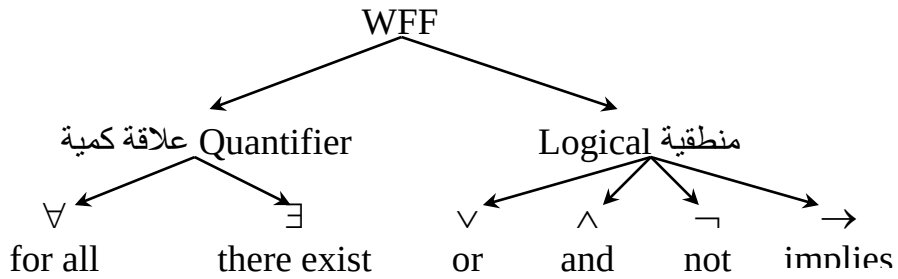
تسلسل البحث في الحدث:

- اذا وجد الفعل يكون الـpredicate.
- اذا لم يوجد الفعل و وجدت الصفة تكون هي الـpredicate.
- اذا لم يوجد الفعل ولا الصفة ووجد الاسم سكون الاسم هو الـpredicate.

* تمثيل المعرفة او المشاكل لتحويل الوصف باللغة الطبيعية الى مقاطع معرفية وصفية دقيقة ممكن التعامل معها حاسوبيا عن طريق المنطق الرياضي الذي هو الـ logic الن المنطق الرياضي الاكثر استخداما في بحوث الذكاء هو الـ predicate logic .

Well formed formula (WFF) هي الجملة المصاغة صياغة لغوية جيدة والتي سوف نتعامل معها

Ex(1): Marcus was man.



Ex(2): Marcus was a Pompeian.

Pompeian(Marcus).

Ex(3): Caesar was a ruler.

Ruler(Caesar).

Ex(4): Marcus tried to assassinate Caesar.

Assassinate(Marcus, Caesar).

Ex(5): Marcus was born in 40 A.D.

Born(Marcus, 40).

Ex(6):امثلة للجمل المركبة

If it is raining then it is not sunny. اذا كانت تمطر فانها غير مشمسة

It is raining it is sunny

Raining(x) sunny(x) x هو الجو

Raining(x) →¬ sunny(x).

Ex(7): there is a person who wrote computer chess. يوجد شخص الذي كتب لعبة الكومبيوتر الشطرنج

∃x: write(x, computer-chess). x هو الشخص

Or

∃x: person(x) → write(x, computer-chess). هنا تم تعريف المتغير

Ex(8): all elephants are of color gray. جميع الفيلة لونها رصاصي

∀x: elephants(x) →color(x, gray). تقرأ x لونه رصاصي

Ex(9): all pompeians were Romans.

∀x: Pompeian (x) → Roman(x).

Ex(10): all Romans were either loyal to Caesar or hated him.

∀x: Roman(x) → loaylto(x, Caesar) ∨ hate(x, Caesar).

Ex(11): everyone is loyal to someone.

∀x: ∃y: loyalto(x, y).

Ex(12): people only try to assassinate rulers they are not loyal to.

∀x: ∀y: person(x) ∧ ruler(y) ∧ tryassassinate(x, y) →¬ loyalto(x, y).

Or

∀x: ∀y: people(x) ∧ ruler(y) ∧ ¬ loyalto(x, y)→ assassinate(x, y).

Ex(13): all men are people.

∀x: man(x) → people(x).

ملاحظة: علاقة الكمية ∀ يجب ان تعرف العلاقة الخاصة بالمتغير بعد الـ∀ اما في الـ∃ فلا ضرورة لذلك.

اذا كان كل من المتغيرات في نفس الجملة من نفس النوع فلا ضرورة لوصف هذه المتغيرات.

2-**Rule**:-

وهي عبارة تتكون من جزئين الجزء الاول هو راس القاعدة والجزء الثاني هو جسد القاعدة، ويحتوي راس القاعدة على الفرضية المطلوب اثباتها او تحققها ، اما جسد القاعدة فيحتوي على المقدمات او الشروط الواجب تحققها لغرض اثبات الفرضية ويحكم كل ذلك قاعدة عامة هي :(if---------then)

Body	Head
* if……..	then
* then …….	If……
Head	Body

Rule

Ex:- brother(X,Y) if father(Z,X),father(Z,Y),male(X)

راس القاعدة جسد القاعدة

Rule

3- **Semantic network**

هي شبكة من المعلومات (object) ذات العلاقة فيما بينها بحيث يمكن استنتاج عبارة جديدة ذات معنى.

العلاقات التي تربط ما بين الـ object

-is-a من صغير الى كبير

-has-a من كبير الى صغير

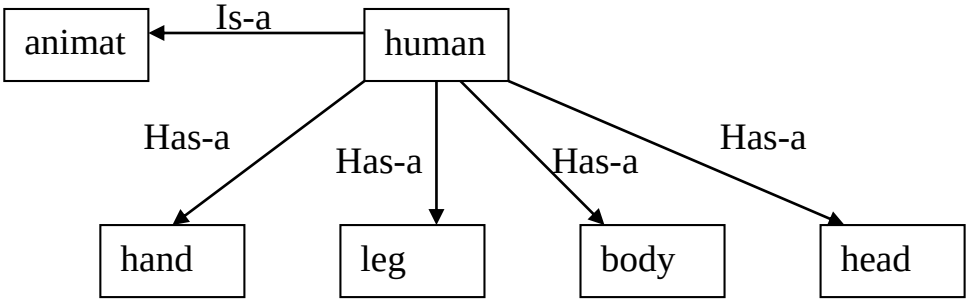
-part-of

-belong-to

- لماذا نعمل semantic network:-

1. للاستنتاج (حقائق جديده من معلومات معينة).
2. لغرض الاختزال في حجم الذاكرة المعدة للخزن.
3. اختصار في الوقت.

ex(1):-



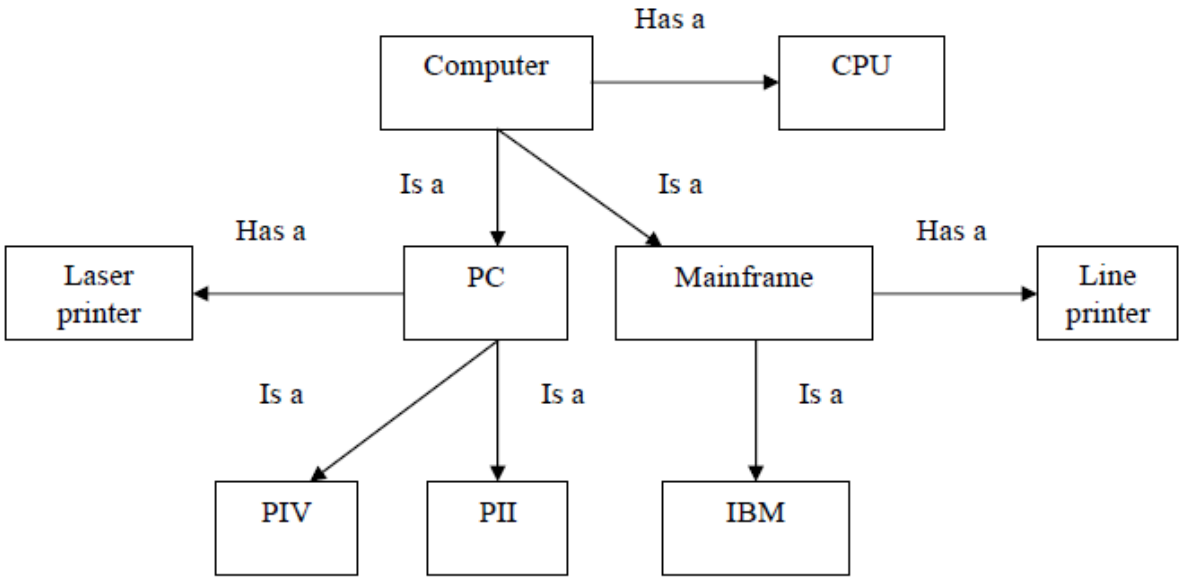
- من صغير الى كبير is-a

Human is a animate

- من كبير الى صغير has-a

Human has a head

Ex(2):-Computer has a many part like a CPU and the computer divided into two kind one is the mainframe and the second is the personal computer. Mainframe has a line printer as a device with al a large sheet but the personal computer has the laser printer and for the example “PIII” and “PIV” and IBM as a example to the mainframe.



Ex(3):- use semantic nets to represent these information.

Consider the following information:-

1-airplane is an aircraft

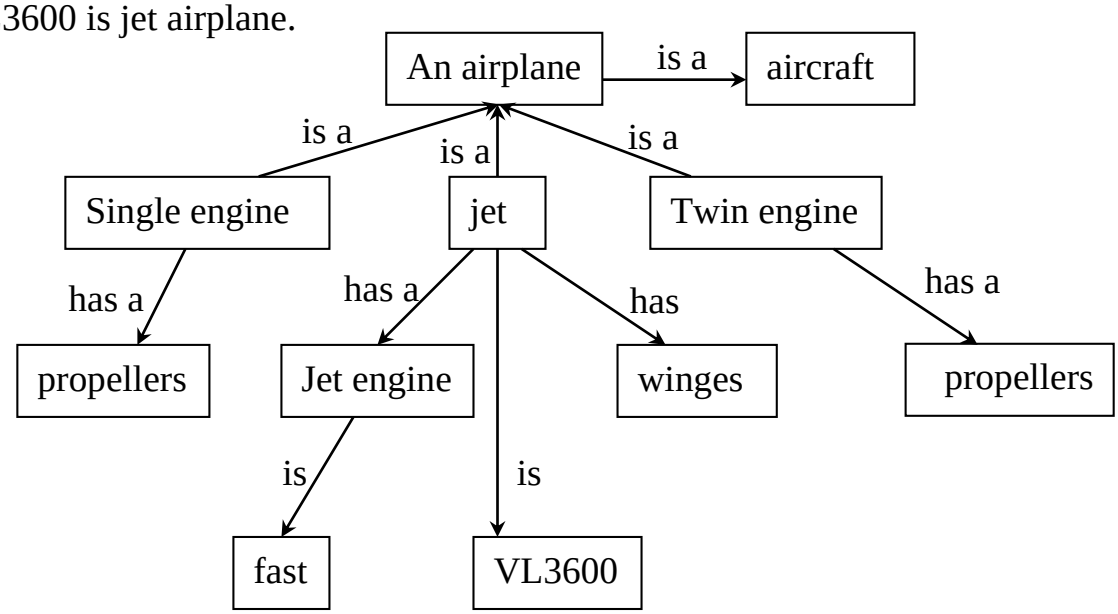
2-jet single engine and twin engine are classified as airplane.

3-jet has jet engine and wings

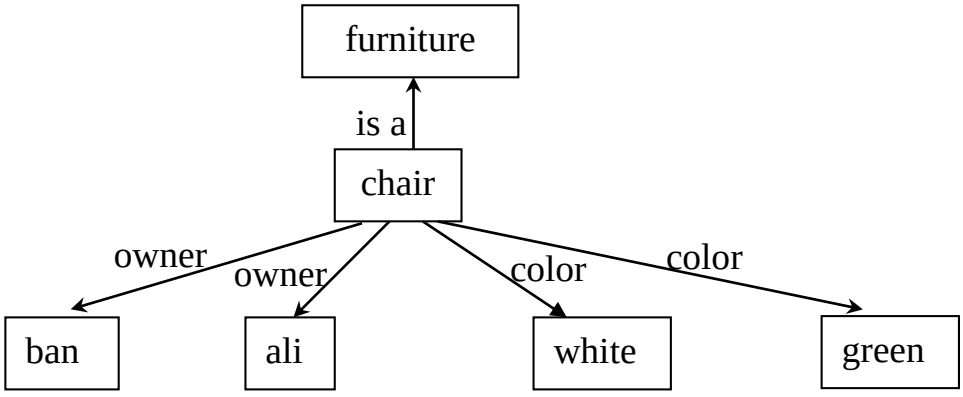
4-single engine and twin engine have propellers.

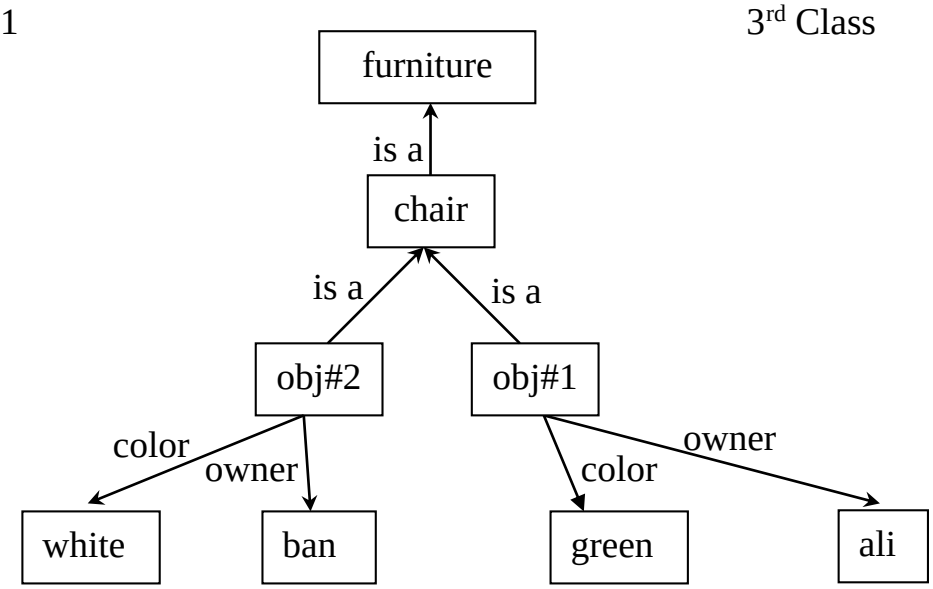
5- jet engine is fast.

6-VL3600 is jet airplane.



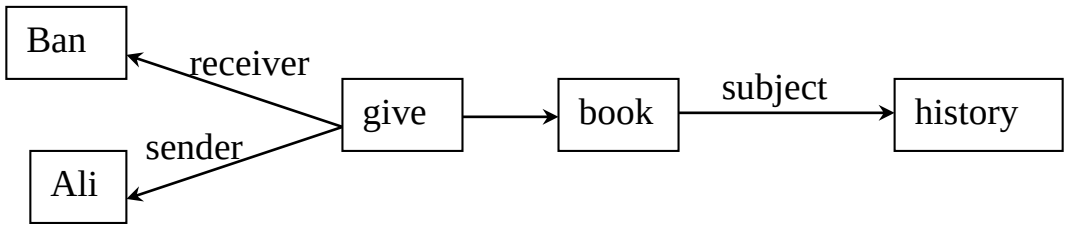
Ex(4): ali's chair is green but ban's chair is white, a chair is furniture.



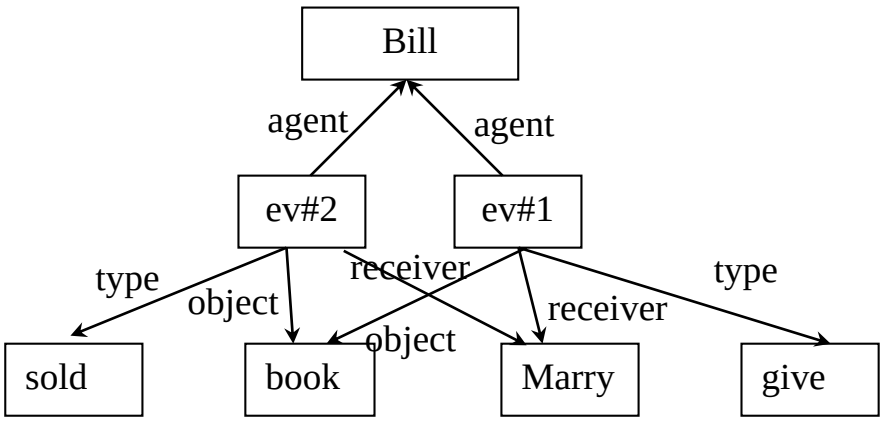


Representing events in semantic net:

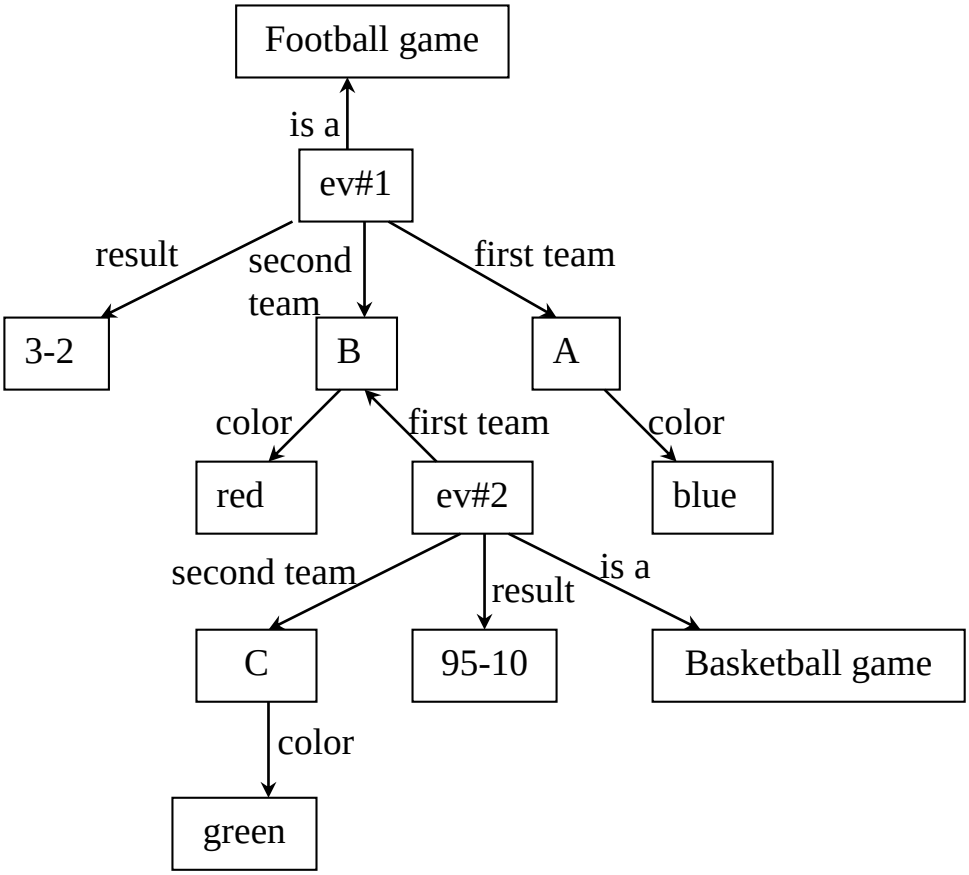
Ex(1): ali gives ban a book about history



Ex(2): Bill give the book to Marry. Bill sold the book to Marry.

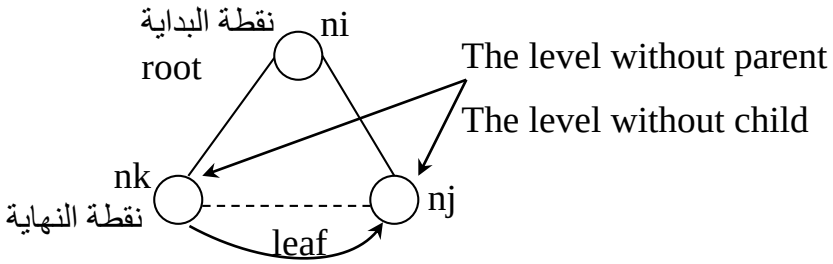


Ex(3): the blue team beats the red team 3-2 in a football match the red team beats the green team 95-10 on a basketball game.



Concept of graph:

Basic definitions of graph theory:



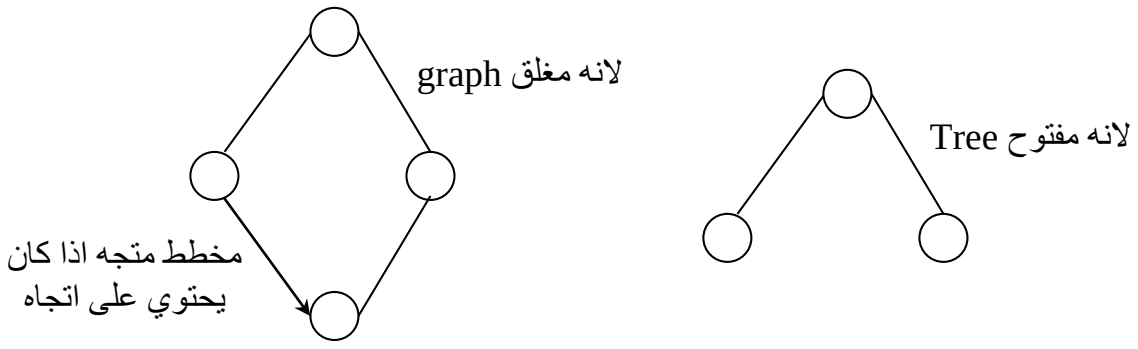
Graph: is a set of nodes connected by arrows if there is a direction it is called directed graph.

Root: is a node without parent.

Leaf: is a node without chilled.

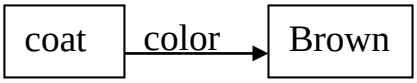
Tree: is a graph without cycle each node connected to other nodes by unique path.

Path: is an ordered set of sequence of nodes (ni……nn) where ni is a parent at ni+1 and path have length n

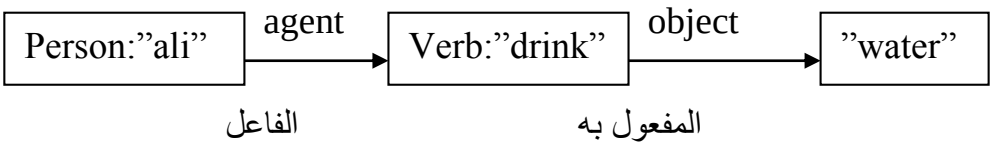


Semantic Graph & Conceptual graph:-

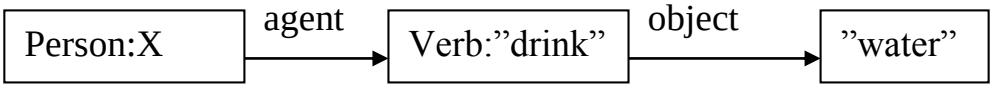
ex1:- it is brown coat.



Ex2:- Ali drink a water.



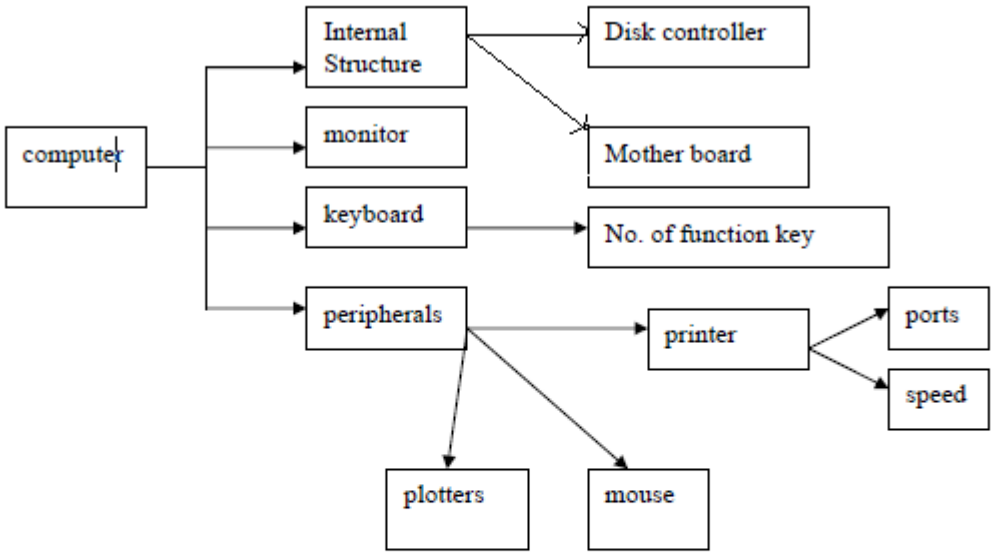
Ex2:- He drink a water.



3) Frame:

Frame-list(node-name, parent, [child]).
Slot-list(node-name, parent).

Example:



Frame –list(computer,_ ,[Internal structure, monitor, keyboard , plotters]).

Frame-list(Internal structure, computer, [disk controller, mother board]).

Frame- list(printer, peripheral, [speed, ports]).

Slot-list(motherboard, Internal structure).

Slot-list(mouse, peripheral).

Clauses form:

Ex(1): all Romans who known Marcus either hate Caesar or think that anyone hates anyone is crazy

$\forall(X)$: [Romans(X) $\wedge$ known (X, Marcus)] $\rightarrow$ [hate(X, Caesar) $\vee$ $\forall$Y: ($\exists$Z: hates (Y, Z) $\rightarrow$ thinkcrazy(X, Y))]

algorithm convert to clauses form:

1) eliminate $\rightarrow$ using the fact that (a$\rightarrow$b) is equivalent to $\neg$ a $\vee$ b

تحويل عباره فيها سهم

-تنفى العبارة التي قبل السهم

-يحول السهم الى or($\vee$)

-العباره التي بعد السهم تنزل كما هي.

Ex(1):-$\forall(X)$: $\neg$ [Romans $\wedge$ known (X, Marcus)] $\vee$ [hate(X, Caesar) $\vee$ $\forall$Y: $\neg$ ($\exists$Z: hates (Y, Z) $\vee$ thinkcrazy(X, Y))]

Ex2:- [(a(X) $\vee$ b(Y) $\rightarrow$ c(X,Y)]

$\neg$ (a(X) $\vee$ b(Y) $\vee$ c(X,Y)

2) reduce the scope of each $\neg$ to a single term using:

معالجة النفي

-النفي اذا دخل على قوس فيه or $\vee$ يتحول الـ $\vee$ الى $\wedge$ and

- النفي اذا دخل على قوس فيه$\wedge$ and يتحول الـ $\wedge$ الى $\vee$ or

$\neg$ ($\neg$ p)=p

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

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

[$\neg\forall$ X: p(X)]= $\exists$ X: $\neg$ p(X)

[$\neg$ $\exists$ X: p(X)]= $\forall$ X: $\neg$ p(X)

$\forall(X)$: [$\neg$ Romans(X) $\vee$ $\neg$ known (X, Marcus)] $\vee$ [hate(X, Caesar) $\vee$ $\forall$Y: $\forall$Z: $\neg$ hates (Y, Z) $\vee$ thinkcrazy(X, Y))]

3) standardized variables so that each quantifier binds a unique variable.

عندما تكون هناك عبارتين مرتبطتين $\vee$ او $\wedge$ لا يجوز الجمع ما بين عبارتين مختلفتين لحواصر ($\forall$،$\exists$) لنفس المتغير للعبارتين لذا يتم تغيره في العباره الاخرى.

Ex-$\forall X\ (a(X) \vee b(Y)) \vee \forall X\ a(X)$

-$\forall X\ (a(X) \vee b(Y)) \vee \forall Z\ a(Z)$

Ex: $\forall$ X: p(X) $\vee$ $\forall$ X: q(X) $\rightarrow$ $\forall$ X: p(X) $\vee$ $\forall$ Y: q(Y)

لم نجري أي تعديل على المثال لانه لم يحتاج لهذه الخطوة.

4) eliminate existential quantifier for examples:

انقل جميع الحواصر الى الجهة اليسرى وبالتسلسل.

Ex-$\forall X(a(X) \wedge b(Y)) \vee \exists Y\ a(Y) \vee \forall Z\ a(Z)$

-$\forall X \exists Y\ (a(X) \wedge b(Y)) \vee a(Y) \vee \forall Z\ a(Z)$

$\forall$ X $\exists$ Y father(Y, X) $\rightarrow$ $\forall$ X father(sun(X), X)

لم نجري أي تعديل على المثال لانه لم يحتاج لهذه الخطوة.

5) move all quantifiers to the left of the formula without changing their relative order

يجب نقل كل الكميات التي في الجملة التي حول for all يجب ان تنقل الى بداية الجملة.

ex-$\forall X(a(X) \wedge b(Y)) \vee \exists Y\ a(Y) \vee \forall Z\ a(Z)$

-$\forall X \exists Y\ \forall z\ (a(X) \wedge b(Y)) \vee a(Y) \vee a(Z)$

$\forall$(X): $\forall$Y: $\forall$Z: [$\neg$ Romans(X) $\vee$ $\neg$ known (X, Marcus)] $\vee$ [hate(X, Caesar) $\vee$ $\neg$ hates (Y, Z) $\vee$ thinkcrazy(X, Y))]

6)a) drop $\exists$ from the sentence

ازالة $\exists$ ،عند ازالة $\exists$ سوف يتم ازالة المتغير المرتبط معها في كل مكان ويتم التعويض عن هذا المتغير بالمتغير الموجود قبله

ex-$\forall X \exists Y\ \forall z\ (a(X) \wedge b(Y)) \vee a(Y) \vee a(Z)$

-$\forall X\ \forall Z\ (a(X) \wedge b(f(X))) \vee a(f(X)) \vee a(Z)$

- في هذا المثال نعوض عن Y بدلالة X

$Y=f(X)$ لانه قبل $\exists Y$ يوجد $\forall X$

Ex-$\forall X\ \forall Z \exists Y\ (a(X,Y,Z))$

-∀X ∀Z (a(X,f(X,Z),Z))

[¬ Romans(X) ∨ ¬ known (X, Marcus)] ∨ [hate(X, Caesar) ∨ ¬ hates (Y, Z) ∨ thinkcrazy(X, Y))]

6)b) drop ∀ from the sentence

ex-∀X ∀Z (a(X) ∧b(Y)) ∨ a(Y) ∨ a(Z)

- (a(X) ∧b(f(X))) ∨ a(f(X)) ∨ a(Z)

7) use distributive lows

$(a \wedge b) \vee c \rightarrow (c \vee a) \wedge (c \vee b)$ $(a \vee b) \wedge c \rightarrow (c \wedge a) \vee (c \wedge b)$

لم نجري أي تعديل على المثال لانه لم يحتاج لهذه الخطوة.

8) create a separate clauses corresponding to each conjunct

فصل المقاطع المربوطة بـ ∧ كل واحدة على حدة

Ex:-(a(X) ∨ a(K) ∨a(Z) ∨ b(S)) ∧(b(K) ∨a(K) ∨ a(Z) ∨ b(S)

a) (a(X) ∨a(K) ∨a(Z) ∨ b(S))

b) (b(K) ∨a(K) ∨ a(Z) ∨ b(S)

لم نجري أي تعديل على المثال لانه لم يحتاج لهذه الخطوة.

9) standardization the variables

a) (a(x) ∨a(k) ∨a(z) ∨ b(s))

b) (b(t) ∨ a(t) ∨ a(n) ∨ b(g))

*العبارة الأولى تنزل كما هي

أي متغير في العبارة الثانية مستخدم في الخطوة الأولى يتم تبديله.

لا يجوز استخدام المتغيرات في العبارة الأولى نفسها في العبارة الثانية.

Ex(2):

$\forall X [p(X) \rightarrow [\forall Y [p(Y) \rightarrow p(f(X, Y))] \wedge \neg \forall Y [q(X, Y) \rightarrow p(Y)]]]$

1) $\forall X [\neg p(X) \vee [\forall Y [\neg p(Y) \vee p(f(X, Y))] \wedge \neg \forall Y [\neg q(X, Y) \vee p(Y)]]]$

2) $\forall X [\neg p(X) \vee [\forall Y [\neg p(Y) \vee p(f(X, Y))] \wedge \exists Y [\neg [\neg q(X, Y) \vee p(Y)]]]]$

$\forall X [\neg p(X) \vee [\forall Y [\neg p(Y) \vee p(f(X, Y))] \wedge \exists Y [q(X, Y) \wedge \neg p(Y)]]]$

3) $\forall X [\neg p(X) \vee [\forall Y [\neg p(Y) \vee p(f(X, Y))] \wedge \exists W [q(X, W) \wedge \neg p(W)]]]$

4) $\forall X [\neg p(X) \vee [\forall Y [\neg p(Y) \vee p(f(X, Y))] \wedge [q(X, g(X)) \wedge \neg p(g(X))]]]$

5) $\forall X \forall Y [\neg p(X) \vee [[\neg p(Y) \vee p(f(X, Y))] \wedge [q(X, g(X)) \wedge \neg p(g(X))]]]$

6) $[[\neg p(X) \vee \neg p(Y) \vee p(f(X, Y))] \wedge [\neg p(X) \vee [q(X, g(X)) \wedge \neg p(g(X))]]]$

$[[\neg p(X) \vee \neg p(Y) \vee p(f(X, Y))] \wedge [\neg p(X) \vee q(X, g(X)) \wedge [\neg p(X) \vee \neg p(g(X))]]]$

7) $[[\neg p(X) \vee \neg p(Y) \vee p(f(X, Y))] \wedge [\neg p(X) \vee q(X, g(X)) \wedge [\neg p(X) \vee \neg p(g(X))]]]$

8) $\neg p(X) \vee \neg p(Y) \vee p(f(X, Y))$

$\neg p(X) \vee q(X, g(X))$

$\neg p(X) \vee \neg p(g(X))$

resolution:

1. **resolution techniques**
2. **resolution strategies**

1) resolution techniques (resolution by refutation):-

resolution proves a goal by negation the statement to be proved and adding it to the set of clauses to be true.

Then uses resolution role of inference to show that this leads to a contradiction.

Once the theorem prove shows that the negation of the goal is inconsistent with the given set of axioms, it follows that the original goal must be true.

However if no contradiction (null statement) can be derived and no new clauses can be generated then we stop and goal is false.

Language $\rightarrow$ predicate logic (WFF) $\xrightarrow{\text{9-seteps}}$ CFF $\rightarrow$ resolution

ملاحظة:- عندما ننفي الـ goal نحصل على تناقض بالمنطق معناه ان الـ goal هو صحيح واذا لم نحصل على تناقض معناه ان الـ goalهو خطأ.

Negation goal →تناقض بالمنطق →true

Negation goal →لا يوجد تناقض بالمنطق →false

Ex(1):

1) "fido is a dog, all dogs are animals, all animals will die" using resolution tp prove (fido will die).

WFF:

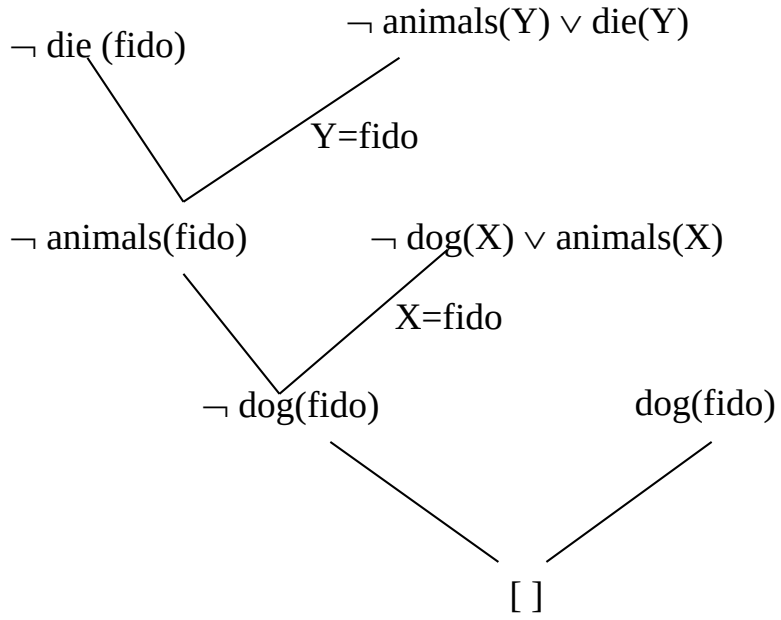
1. dog(fido) fact
2. $\forall$ X (dog(X) $\rightarrow$ animals(X)) rule
3. $\forall$ Y (animals(Y) $\rightarrow$ die(Y)) rule
4. $\neg$ die(fido)? fact دائما السؤال يكون مسبوق بنفي

CFF: هنا الحقائق تبقى على حالها والقواعد تنفى

1. dog(fido)

2. ¬ dog(X) ∨ animals(X)
3. ¬ animals(Y) ∨ die(Y)
4. ¬ die (fido)?

ملاحظة: نبدا الحل من السؤال الموجود في االـCFF ونبحث في CFF عن أي قاعدة تحوي على اسم الحقيقة التي في السـؤال. ونبدا بالمطابقة للحصـول على قيم للمتغيرات (أي نقوم بعمل unification) وهكذا نسـتمر الى ان نصل الى قائمة فارغة.



∴ yes, fido will be die

ملاحظة:- تكون الإجابة بنعم دائما في حالة الوصـول إلى قائمة فارغة. وفي حالة عدم الوصـول الى قائمة فارغة تعني الإجابة no.

ملاحظة:- دائما السؤال او الاستفسار يكون مسبوق بنفي. واذا كان السؤال او الاستفسار في الاصل منفي ونضع له نفي يصبح مثبت.

Ex:- prove the fido will not die

¬ ¬ die (fido) → die(fido)

Ex(2):

Consider we want to prove a form of the following send by using resolution:

1. anything anyone play and isn't hunt by a game
2. yosif play football and isn't hunt by
3. tennis is a game
4. swimming is a game
5. zeki play all game

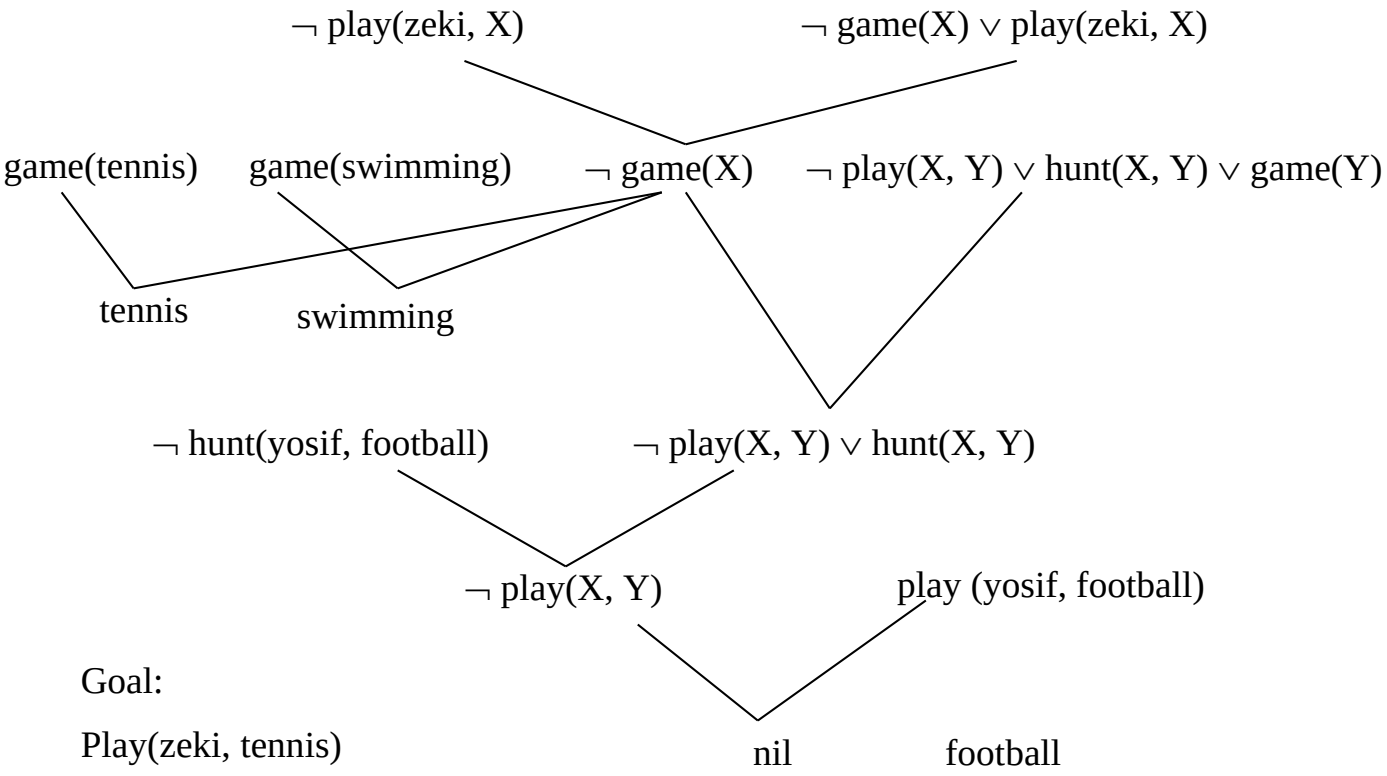
goal: what is zeki play? Play(zeki, X)

WFF:

1. $\forall$ X: $\forall$ Y: play(X, Y) $\wedge \neg$ hunt(X, Y) $\rightarrow$ game(Y)
2. play (yosif, football) $\wedge \neg$ hunt(yosif, football)
3. game(tennis)
4. game(swimming)
5. $\forall$ X: game(X) $\rightarrow$ play(zeki, X)

CFF:

1. $\neg$ play(X, Y) $\vee$ hunt(X, Y) $\vee$ game(Y)
2. play (yosif, football)
3. $\neg$ hunt(yosif, football)
4. game(tennis)
5. game(swimming)
6. $\neg$ game(X) $\vee$ play(zeki, X)
7. $\neg$ play(zeki, X)



Goal:

Play(zeki, tennis)

Play(zeki, swimming)

Play(zeki, football)

2) resolution strategies: (backword and forword control)

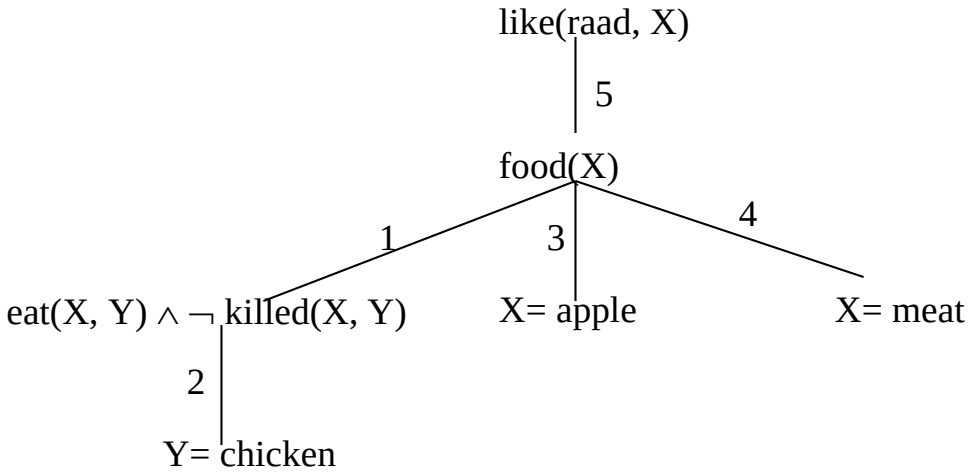
example:

1. anything anyone eats and not killed by is a food.
2. saad eat chicken and still a live.
3. apple is a food.
4. meat is a food.
5. raad like all food.

Answer the question (what like raad) by using the resolution strategies?

Solution:

1. convert to predicate logic:
   - $\forall$ X: $\forall$ Y: eat(X, Y) $\wedge \neg$ killed(X, Y) $\rightarrow$ food(Y).
   - eat(saad, chicken) $\wedge$ live(saad, chicken).
   - food(apple).
   - food(meat).
   - $\forall$ X: food(X) $\rightarrow$ like(raad, X).
2. by using backword chaining:



* then raad like chicken
  apple
  meat

OR by using forword chaining:

Facts:

eat(saad, chicken)

live(saad, chicken)

food(apple)

food(meat)

rules:

R1= ∀ X: ∀ Y: eat(X, Y) ∧ ¬ killed(X, Y) → food(Y).

R2= ∀ X: food(X) → like(raad, X)

R3= ¬ killed(X, Y) → live(X, Y)

no	facts	rule	find
1	eat(saad, chicken)	R1	R4: ¬ killed(X, Y) → food(Y)
2	live(saad, chicken)	R3	New fact: ¬ killed(saad, chicken)
3	¬ killed(saad, chicken)	R4	food(chicken)
4	food(chicken)	R2	like(raad, chicken)
5	food(apple)	R2	like(raad, apple)
6	food(meat)	R2	like(raad, meat)

* then raad like chiken

apple

meat

من الممكن اضافة بعض الـ rules التي توازن قاعدة البيانات من دون ان تغير المعنى

As example

¬ killed(X, Y) → live(X, Y)

killed(X, Y) →¬ live(X, Y)

¬ killed(X, Y) → ¬ live(X, Y)

live(X, Y)→¬ killed(X, Y)

¬live(X, Y)→killed(X, Y)

¬live(X, Y)→¬killed(X, Y)

النظم الخبيرة (Expert Systems)

النظم الخبيرة، أحد أقوى فروع الذكاء الإصطناعي الذي يعتبر بدوره أقوى فروع علم الحاسب الآلي.

فما هي النظم الخبيرة (Expert Systems) ؟

هي برامج تُحاكي آداء الخبير البشري في مجال خبرة معين ، وذلك عن طريق تجميع واستخدام معلومات وخبرة خبير أو أكثر في مجال معين.

باختصار هذه النظم أوجدت من أجل استخلاص خبرات الخبراء -وخصوصاً في التخصصات النادرة – وضمها في نظام خبير يحل محل الإنسان ويساعد في نقل هذه الخبرات لأناس آخرين بالإضافة إلى قدرته على حل المشكلات بطريقة أسرع من الخبير البشري

.

*النظم الخبيرة حسب تعريف الباحث الدكتور إدوارد فينجن باوم هي " نظام المعرفة أو النظام الخبير هو ذلك البرنامج الذكي الذي يستخدم القواعد المأخوذة من الخبرة الإنسانية على هيئة شروط ونتائج في مجال معين وإستخدام طرق الإشتقاق والإستدلال لإستخراج وإستنتاج النتائج المعللة بالإسباب والناتجة عن تطابق هذه الشروط أو النتائج مع شرط أو نتيجة ما والخاصة بمشكلة معينة يراد إيجاد حل لها."

من مميزات هذه النظم:

1- أنها سهلة الإستخدام لأي مستخدم سواء مستخدم عادي أو مطوّر.
2- أنها نافعة في مجال التطبيق بشكل واضح.
3- قادرة على التعلم من الخبراء بطريقة مباشرة وغير مباشرة.
4- قادرة على تعليم غير المتخصصين.
5- قادرة على تفسير أي حلول تتوصل إليها مع توضيح طريقة الوصول إليها.
6- قادرة على الإستجابة للأسئلة البسيطة وكذلك المعقدة في حدود التطبيق.
7- وسيلة مفيدة في توفير مستويات عالية من الخبرة في حال عدم توفر خبير.
8- قادرة على تطوير آداء المتخصصين ذوي الخبرة البسيطة.

س. مع كل هذه المميزات القوية إلا أننا لم نلحظ انتشاراً قوياً لمثل هذه الأنظمة فما السبب ؟
الأسباب:

*أنها ذات تكلفة عالية مقارنةً بالتطبيقات التقليدية.
*نظام تطبيقها محدود في النظم الإدارية واسترجاع المعلومات المتكاملة.

إلا أنه ومع هذه المشاكل هناك أسباب قوية تجعل بعض الشركات تتغلب على هذه المشاكل منها:

*الإحتفاظ بالخبرة والمعرفة من الإندثار أو الإنقراض ، وخصوصاً في التخصصات الهامة الكثيرة الإستخدام أو النادرة.
*حل المشاكل ، مما يحفظ الوقت و المال والجهد.
*زيادة الخبراء في مجال تطبيق النظام الخبير.

ومن أهم مجالات تطبيقات نظم الخبرة هو التصنيف (classification) حيث يكون مطلوب من النظام تحديد الفئة التي ينتمي إليها الكائن المطلوب تصنيفه ، كما أن النظم الخبيرة دخلت في عدة مجالات أخرى كالطب والزراعة والتنقيب والإلكترونيات والحاسبات والجيولوجيا والهندسة والتعليم والشريعة والقانون والتجارة والإقتصاد وغيرها الكثير

ولإنتاج نظام خبير يجب توفر عنصرين هامين هما:

1- المبرمج الذي يقوم بتحليل المشكلة وكتابة البرنامج في مجال الذكاء الإصطناعي.
2- خبير المجال وهو الشخص المتخصص في مجال معين وليس بالضرورة أن يكون لديه علم بالذكاء الإصطناعي فالمهم مدى خبرته وإلمامه ببواطن الأمور في مجال تخصصه.

ويمر النظام الخبير بعدة مراحل حتى يظهر بالشكل المطلوب وهي كالتالي:

1- تعريف التطبيق : وفيها يتم تحديد مالذي نريده من النظام ومجال الخبرة.
2- تصميم النظام
3- برمجة النظام
4- اختبار النظام وتوثيقه

ولكل خطوة من هذه الخطوات الأشخاص المكلفين بالقيام بها.

ومن الأمثلة على النظم الخبيرة:

نظام Eliza للعلاج النفسي : وهو عبارة عن نظام يُجري حوار مع المستخدم ويجيب على الإستفسارات كطبيب نفسي خبير.

مكونات النظام الخبير:

يتكون النظام الخبير من ثلاثة أجزاء أساسية:
1- قاعدة المعرفة: Knowledge Base

حيث تتضمن قاعدة المعرفة مايلي:

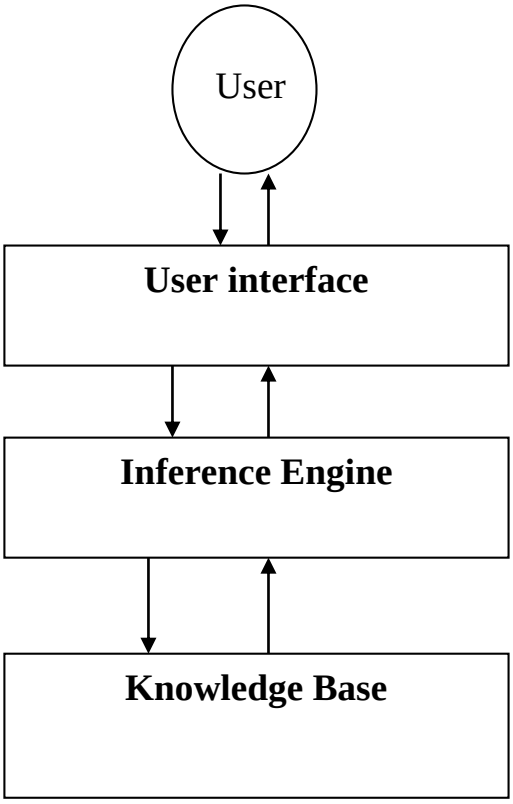
•قاعدة حقائق : Facts Base تصف العلاقة المنطقية بين العناصر والمفاهيم ومجموعة الحقائق المستندة إلى الخبرة والممارسة للخبراء في النظام.

•طرق حل المشكلات وتقديم الاستشارة.

•مجموعة قواعد Rules Base: تنمذج المعرفة في المجال قيد الدراسة و هي غالبا قواعد شرطية تكون مستندة على صيغ رياضية.

2- محرك استدلال Inference Engine (IE): قادر على المحاكمة Reasoning بدءا من معلومات مضمّنة في قاعدة المعرفة.

3-واجهة المستخدم (user interface) : وهي الإجراءات التي تجهز المستخدم بأدوات مناسبة للتفاعل مع النظام خلال مرحلتي التطوير والاستخدام .



Components of Expert systems