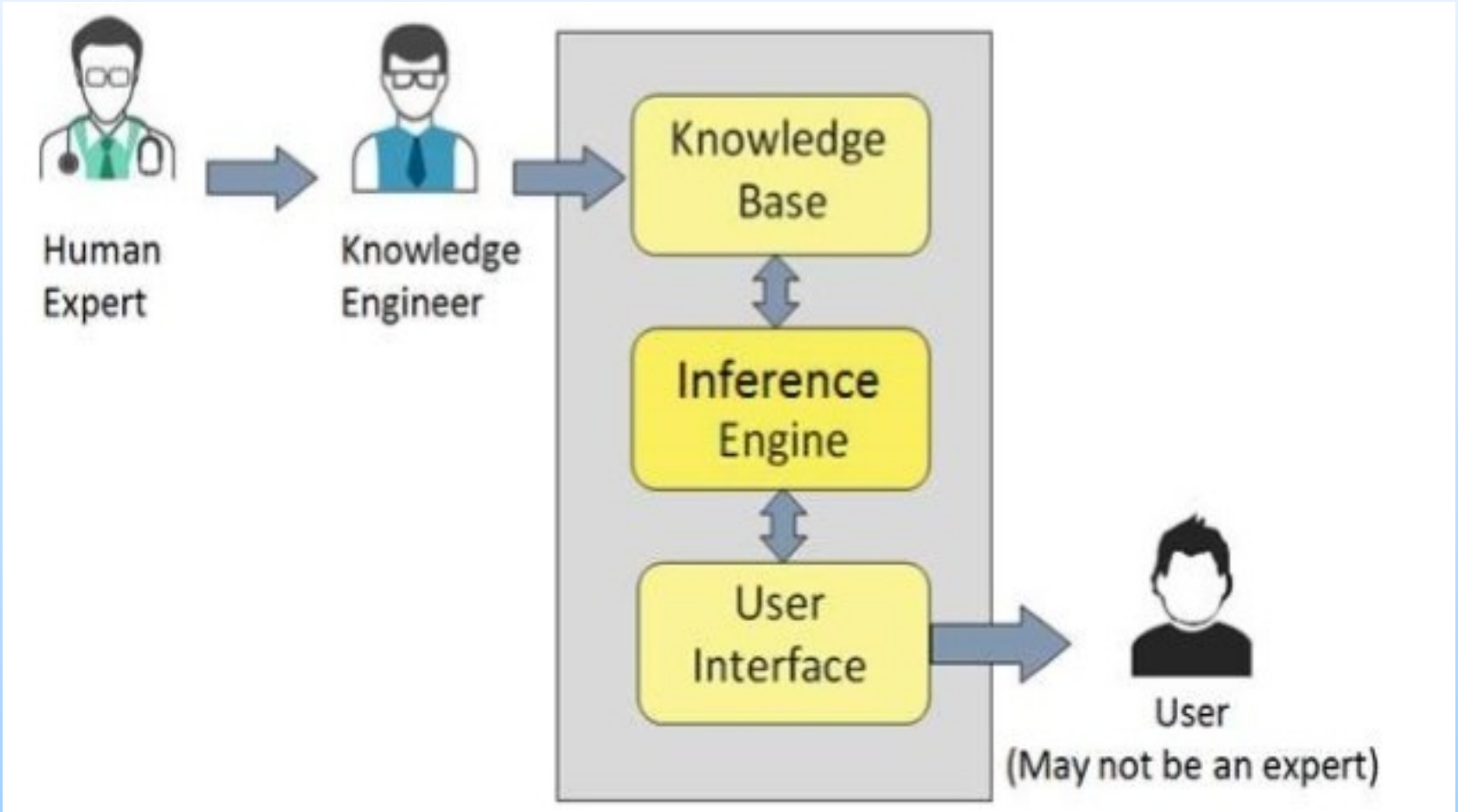
Chapter: 2

Knowledge Representation

Expert Systems

They are programs that contain a huge amount of information owned by a human expert in a particular field of knowledge

The Core Components of an Expert System in AI



Knowledge

- **Knowledge is information and understanding about a subject which a person has, or which all people have.**
- **Knowledge is awareness gained by experiences of facts, data, and situations.**
- **It includes concepts, facts, and objects.**

Knowledge Base

- **The knowledge base contains the knowledge necessary for understanding, formulating, and solving problems**

- **Two basic elements of knowledge base**
  - **Facts.**
  - **Rules that use of knowledge.**

Inference Engine

- The ***brain*** of the ES.
- The control structure.
- Provides methodology for reasoning

User Interface

- **An interface between the user and the system and is as menus and graphics.**

The Human Element in ES.

- **Expert**
- **Knowledge Engineer**
- **User**
- **Others**

How Expert Systems Work

Major Activities of ES Implementation and Use

- **Development**
- **Consultation**
- **Improvement**

ES Shell

- **Includes All General ES Components**

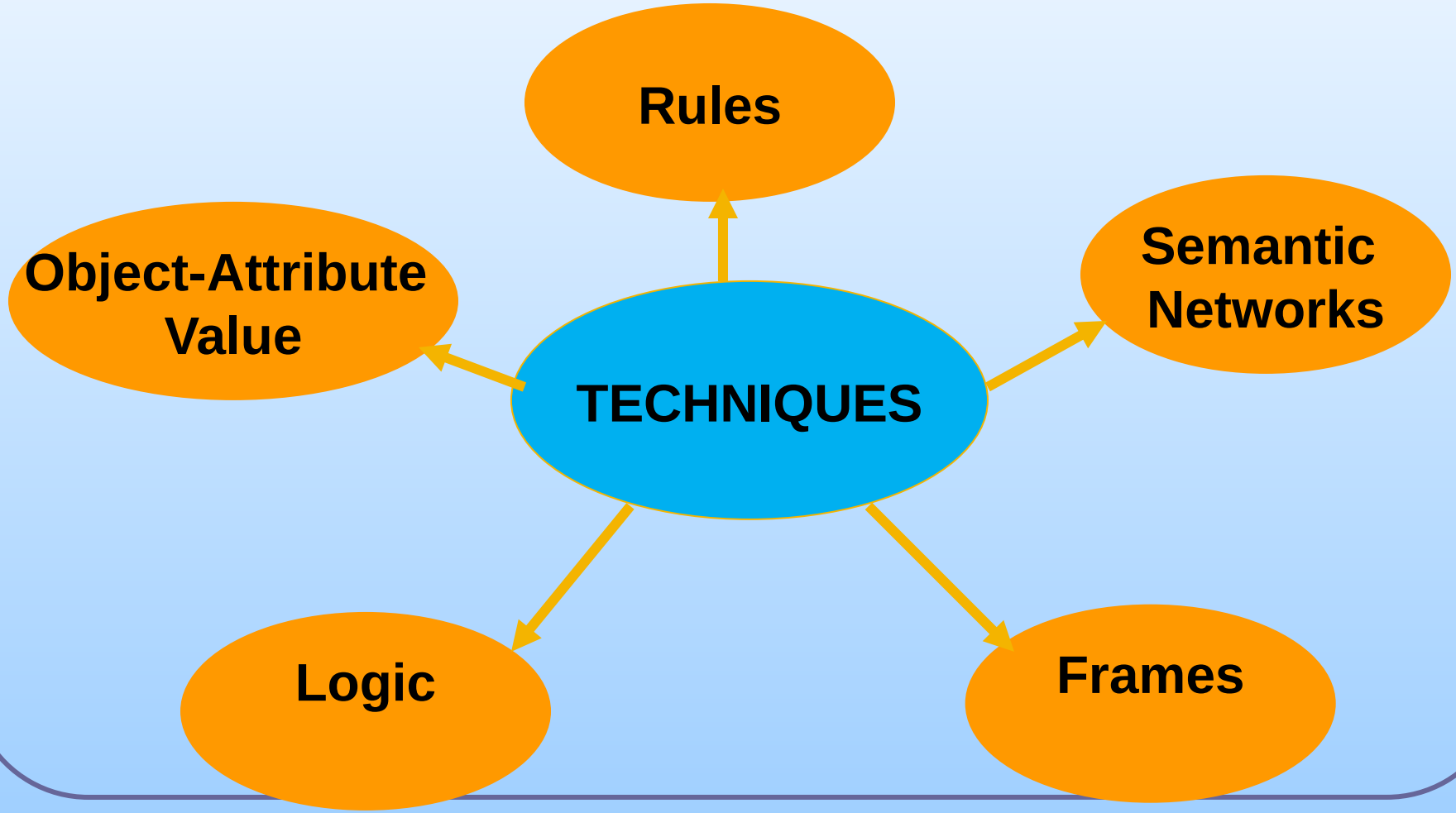
Knowledge Representation

- **It is a way used to represent the knowledge that a computer can understand and use this knowledge to solve the complex problems such as diagnosis a medical cases or communicating with humans in natural language, and ...etc.**

Knowledge Representation

- **It is also a way which describes how we can represent the knowledge in artificial intelligence systems.**
- **It is responsible to create the knowledge base (KB) which is a part of the components of AI systems.**

Knowledge Representation Techniques



Logic

- **Implemented using PROLOG, LISP language.**
- **Representing knowledge and building its rules requires the conversion of these sentences into formats that are easily represented within computer systems.**
- **Propositional logic is either simple or complex sentences**

First-order predicate logic(FOPL)

Following are the basic elements of FOPL syntax:

1. **Constant** 1, 2, A, John, Mumbai, cat,....
2. **Variables** x, y, z, a, b,....
3. **Predicates** Brother, Father,....
4. **Function** sqrt, LeftLegOf, ...
5. **Connectives** $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$
6. **Equality** ==
7. **Quantifier** $\forall$, $\exists$
8. **Parentheses** { }

First-order predicate logic(FOPL)

9. Logical Operators:

General Name	Formal Name	Symbols
Not	Negation	$\neg$
And	Conjunction	$\wedge$
Or	Disjunction	$\vee$
If… Then/Implies	Conditional	$\rightarrow$
If and only if	Biconditional	$\leftrightarrow$

First-order predicate logic(FOPL)

10. 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, as:**

  Predicate (term1, term2, ..., term n).

- **Examples:**

1. **Hany and Saied are brothers: => Brothers(Hany, Saied).**
2. **Tom is a cat: => cat (Tom).**

First-order predicate logic(FOPL)

Complex Sentences:

- **Complex sentences are made by combining atomic sentences using connectives.**
- **Example:**

Saied mother is married to Khalid father

married(father(X, khaled), mother(Y, saied))

First-order predicate logic(FOPL)

First-order logic statements can be divided into two parts:

- **Subject: Subject is the main part of the statement.**
- **Predicate: A predicate can be defined as a relation, which links two atoms together in a statement.**

First-order predicate logic(FOPL)

Consider the statement:

"x is an integer.",

it consists of two parts:

The first part: x is the subject of the statement

The second part: "is an integer," is known as a predicate

First-order predicate logic(FOPL)

Quantifiers in (FOPL):

There are two types of quantifier:

1) **Universal Quantifier,**
   (for all, everyone, everything)

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

First-order predicate logic(FOPL)

1) Universal Quantifier:

Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.

First-order predicate logic(FOPL)

- **The Universal quantifier is represented by a symbol ∀.**
- **Note: In universal quantifier we use implication "→".**
- **If x is a variable, then ∀x is read as:**
- **For all x**
- **For each x**
- **For every x.**

First-order predicate logic(FOPL)

- **Example:**

All man drink coffee.

- **FOPL: $\forall x \{man(x) \rightarrow drink(x, coffee)\}$.**

- **It will be read as:**

There are all x where x is a man who drink coffee.

First-order predicate logic(FOPL)

2) Existential Quantifier:

Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.

First-order predicate logic(FOPL)

- **It is denoted by the logical operator ∃.**
- **Note: In Existential quantifier we always use AND or Conjunction symbol (∧).**
- **If x is a variable, And it will be read as:**
- **There exists a 'x.'**
- **For some 'x.'**
- **For at least one 'x.'**

First-order predicate logic(FOPL)

- **Example:**

Some boys are intelligent.

FOLP: $\exists x\{boys(x) \wedge intelligent(x)\}$

- **It will be read as:**

There are some x where x is a boy who is intelligent.

Examples of FOPL

1. All birds fly.

- **In this question the predicate is "fly(bird)."**
- **since there are all birds who fly so it will be represented as follows.**

FOPL: $\forall x \{bird(x) \rightarrow fly(x)\}$.

Examples of FOPL

2. Every man respects his parent.

- **In this question, the predicate is "respect (x, y)," where x=man, and y= parent.**
- **Since there is every man so will use ∀, and it will be represented as follows:**

FOPL: ∀x{ man(x) → respects (x, parent)}.

Examples of FOLP

3. Some boys play football.

- **In this sentence, the predicate is "play(x, y)," where x= boys, and y= football.**
- **Since there are some boys so we will use ∃, and it will be represented as:**

FOPL: ∃x {boys(x) ∧ play(x, football)}.

Examples of FOLP

4. Not all students like both Mathematics and Science.

- **In this question, the predicate is "like(x, y)," where x= student, and y= subject.**
- **Since there are not all students, so we will use ∀ with negation, so following representation for this:**

FOPL:¬∀(x){student(x) → like(x, Mathematics) ∧ like(x, Science)}.

Examples of FOLP

5. Only one student failed in Mathematics.

- **In this sentence, the predicate is "failed(x, y),' where x= student, and y= subject.**
- **Since there is only one student who failed in Mathematics, so we will use following representation for this:**

FOPL:∃(x){student(x) ∧ failed(x,Mathematics)}

Exercise:1

Write FOPL for the Following sentences:

- **“If it doesn’t rain today, Saied will go to school”**
- **“All volleyball players are tall”**
- **“Some people like fish.”**
- **“Nobody likes wars“**

Exercise:1

Normal: “If it doesn’t rain today, Saied will go to school”

FOPL: rain(today)□ go_to(saied, school)

Normal: “All volleyball players are tall”

FOPL: (x){volleyball_player (x) □ tall (x)}

Normal: Some people like fish.

FOPL: (x) {person(x) likes(x, fish)}

Normal: Nobody likes wars

FOPL: (x){ likes (x, wars)}

Exercise:2

Write FOPL for the Following sentences:

- **For all two numbers, one of them is maximum if is greater than other.**

- **For any two persons, are brothers if there is another person who is their parent.**

Exercise:2

- **For all two numbers, one of them is maximum if is greater than other.**

  Fopl: (x) (y){greater-than(x,y)->max(x)}

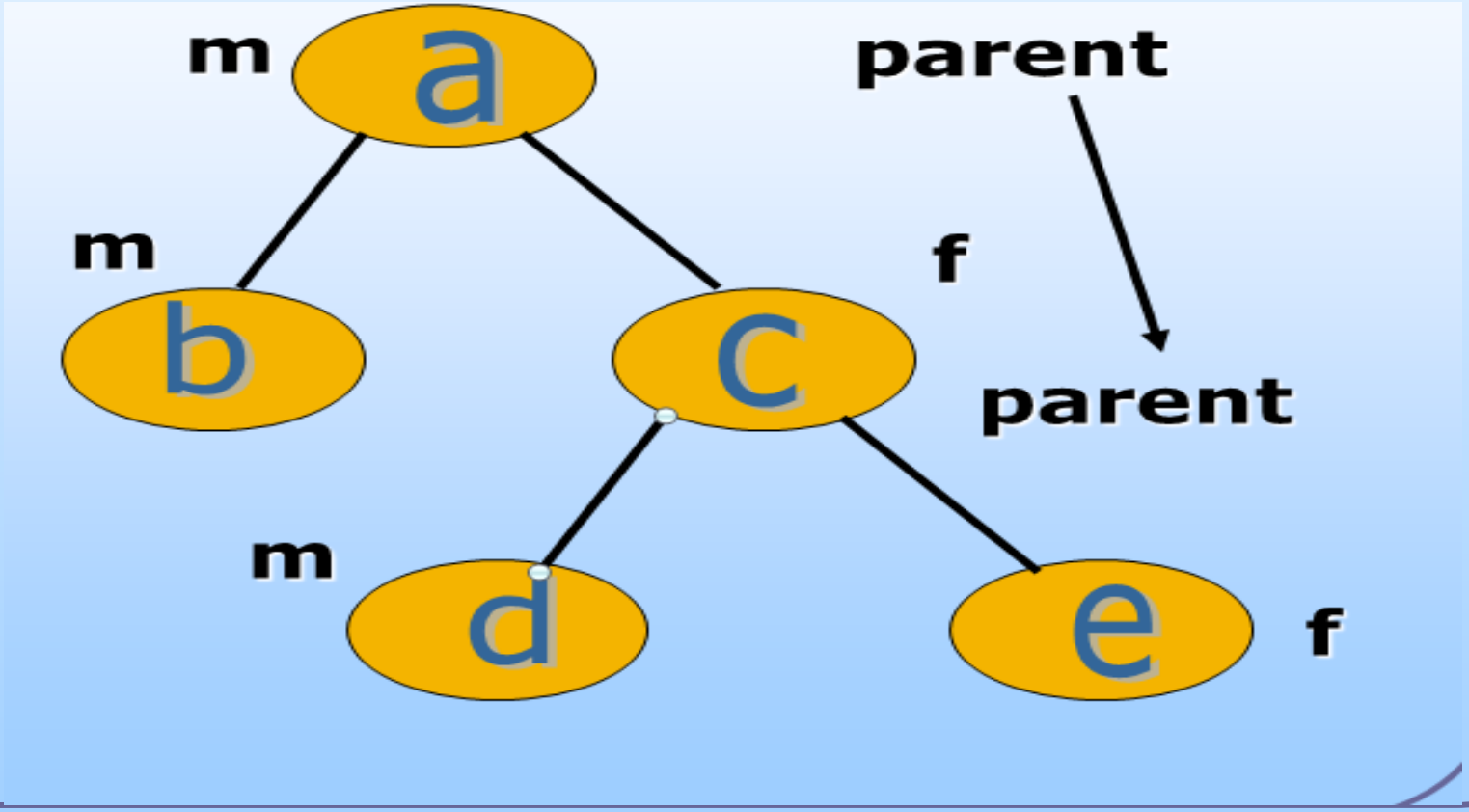
  Fopl: (x) (y){greater-than(y,x)->max(y)}

- **For any two persons, are brothers if there is another person who is their parent.**

  Fopl: (x) (y) (z) {parent(z,x) parent(z,y) male(x) ->brother(x,y)}

Exercise:3

Consider the following family tree and



Exercise:3

- **write the FOPL for the following predicates:**
  1. **Father**
  2. **Uncle**
  3. **Sister**
  4. **grandfather**

Introduction to PROLOG

PROgramming in **LOG**ic

Introduction

- **Used for symbolic and non-numerical computation**
- **Has a built in intelligent search mechanism**
- **Can handle complex problems in compact programs**
- **Writing a program in Prolog means writing facts and rules which together comprise knowledge base**
- **Facts and rules use predicates which represent relationships among data objects.**

The Basics

- **Numbers**
  - **Include integers and real numbers**

    3.14 ,0.0035 - ,3131 ,1

- **Variables**
  - **represents some unspecified element of the system and always start with an upper-case letter or an underscore:**

    X, Y, Child, _a23, Student_List

The Basics

- **Constant**

  always start with a lower-case letter or digit:

  hany, john, jan

- **A predicate**

  represents some relation or property in the system. Predicates and constants always start with a lower-case letter or digit:

  likes, read, and plays

The Basics

- **Logical symbols**

if -:

and ,

or ;

- **Single line comments use the “%” character**

- **Multi-line comments use /* and */**

The Basics

- **Each line in a Prolog program is called a clause. There are two types of clauses - facts and rules:**
  - **Rules are clauses which contain the ":-" symbol. Each *rule* consists of a predicate, followed by a ":-" symbol, followed by a list of predicates separated by "," or ";"**
  - **Facts are clauses which don't the ":-" symbol. Each *fact* consists of just one predicate**

The Basics

- **Every clause is terminated by a “.” (full-stop).**
- **In a rule, the predicate before the “:-” is called the head of the rule**
- **The predicates coming after the ``:-'' are called the body**

The Basics

- **Simple I/O in Prolog**
  - **Use the write statement**
    - **write(‘hello’)**
    - **write(‘Hello’), write(‘World’)**
  - **Use a Newline**
    - **write(‘hello’), nl, write(‘World’)**

The Basics

- **Reading a value from stdin**

- **Prolog Syntax:**
  - **read(X)**
- **Example**

.read(X), write(X)

The Basics

- **Using Arithmetic**
  - **Different to what you may have seen with other languages.**
  - **Operators**
    - **< <= == != => >**
    - **+ - * /**
  - **Arithmetic is done via evaluation then unification**

The Basics

- **Arithmetic Example**

- **X is Y**
  - **Compute Y then unify X and Y**
    - **X is Y * 2**
    - **N is N - 1**

The Basics

- **X == Y**
  - **This is the identity relation. In order for this to be true, X and Y must both be identical variables (i.e. have the same name), or both be identical constants, or both be identical operations applied to identical terms**
- **X = Y**
  - **This is unification**
  - **It is true if X is unifiable with Y**

The Basics

- **X=:=Y**
  - **This means “compute X, compute Y, and see if they both have the same value”**
  - **both X and Y must be arithmetic expressions**

The Basics

- **Arithmatic Example: (factorial)**

$$fact(x) = \begin{cases} 1, & \text{when } x = 0 \\ x * fact(x-1), & \text{when } x > 0 \end{cases}$$

fact(0,1).

fact(X,F) :- X>0, X1 is X-1, fact(X1,F1), F is X*F1.

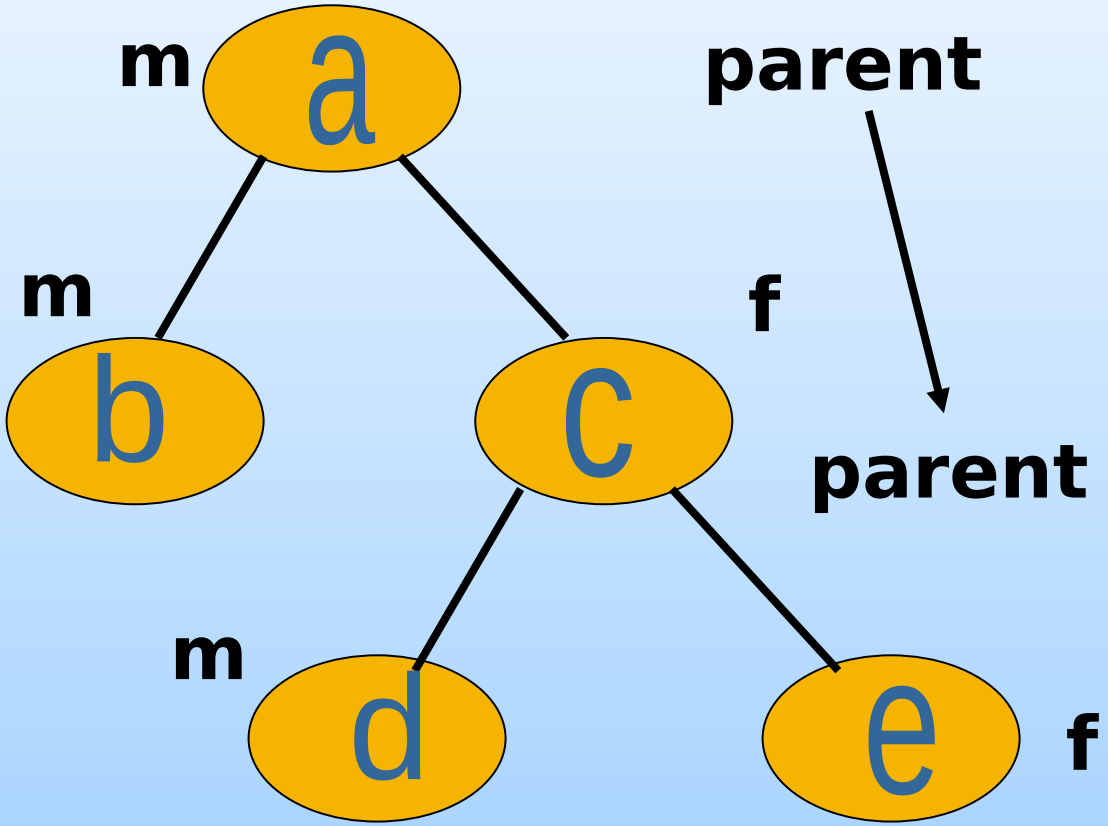
The Basics

- **Recursive rules**
- **Example**
  - **Rule 1: predecessor(X,Z):- parent (X,Z).**
  - **Rule 2: predecessor(X,Z):- parent(X,Y),predecessor(Y,Z)**

The Basics

- **Queries**
  - **Means of asking the program what things are true**
  - **Have empty head**
  - **Example: ? parent(tom, bob).**
    - **Is tom parent of bob ?**
  - **Example: ? parent(tom, X).**
    - **Find X such that tom is parent of X.**

Example:1



Example:1

male (a).
male (b).
male (d).
female (c).
female (e).
parent (a,b).
parent (a,c).
parent (c,d).
parent (c,e).

Facts

Example:1

Rules

Father (x,y):- parent (x,y) , male (x).

Brother (x,y):- father (z,x) , father (z,y) , male (x).

Sister (x,y):- father (z,y) , father (z,x) , female (x).

Uncle (x,y):- father (z,y) , brother (x,z).

Example:1

Goals:

Father (X,d)
uncle (c,d)
sister (X,b)
brother (X,b)
Parent(c,X)
Grandfather(a,X).
female (x)

:Exercises

- **Create the knowledge base for the following problems**:

Problem:1

- **Summation of two numbers**
- **Multiplication of two numbers**
- **Division of two numbers**
- **Subtraction of two numbers**

Problem:2

- **Maximum number between two numbers.**
- **Maximum number among three numbers.**
- **Summation of the odd numbers**
- **Factorial of number**

Problem:3

- **Car Showroom, we can query about the following things for any car :**
  1. **Brand**
  2. **Model**
  3. **Color**
  4. **Price.**

.

Problem:4

Consider the following table (tab1) and explain by KBS how to classify the students to the following departments:

1. **Biology.**
2. **Chemistry.**
3. **Physics.**
4. **Mathematics.**

Problem:4

tab1

name	markbi	markch	markphy	markmath
Khaled	95	80	70	75
Yaser	88	95	77	66
Hoda	87	70	85	95
Hamdy	90	85	65	70
Ranya	65	78	95	87
Hany	85	95	60	65

Problem:4

- **Requirements for the Biology Department.**
  1. **Biology mark>=95.**
  2. **Chemistry mark>=85.**
  3. **Total summation>=280.**

- **Requirements for the Chemistry Department.**
  1. **Biology mark>=85.**
  2. **Chemistry mark>=95.**
  3. **Total summation>=300.**

Problem:4

- **Requirements for the Physics Department**
  1. **Physics mark>=95.**
  2. **Mathematic mark>=85.**
  3. **Total summation>=285.**

- **Requirements for the Mathematics Department**
  1. **Physics mark>=85.**
  2. **Mathematic mark>=95.**
  3. **Total summation>=300**

The KBS:

- **Facts:**

mark(90,Khaled,biology).

mark(88,yaser,biology).

……..

mark(80,Khaled,chemistry).

mark(95,yaser,chemistry).

……

The KBS:

mark(70,Khaled,physics).

mark(77,yaser,physics).

……..

mark(75,Khaled,mathematic).

mark(66,yaser,mathematic).

……

The KBS:

- **Rules**

sum(X,Y):-

mark(B, Y, biology),

mark(C, Y, chemistry),

mark(P, Y, physics),

mark(M, Y, mathematic),

X is B+C+P+M.

The KBS:

dept(biology, Y):-

mark(B, Y, biology),

mark(C, Y, chemistry),

Sum(S,Y),

B>=95, C>=85, S>=280.

The KBS:

dept(chemistry, Y):-

mark(B, Y, biology),

mark(C, Y, chemistry),

Sum(S,Y),

B>=85, C>=95, S>=300.

:BY KBS

dept(physics, Y):-

mark(P, Y, physics),

mark(M, Y, mathematic),

sum(S,Y),

P>=95, M>=95, S>=285.

The KBS:

dept(mathematic, Y):-

mark(P, Y, physics),

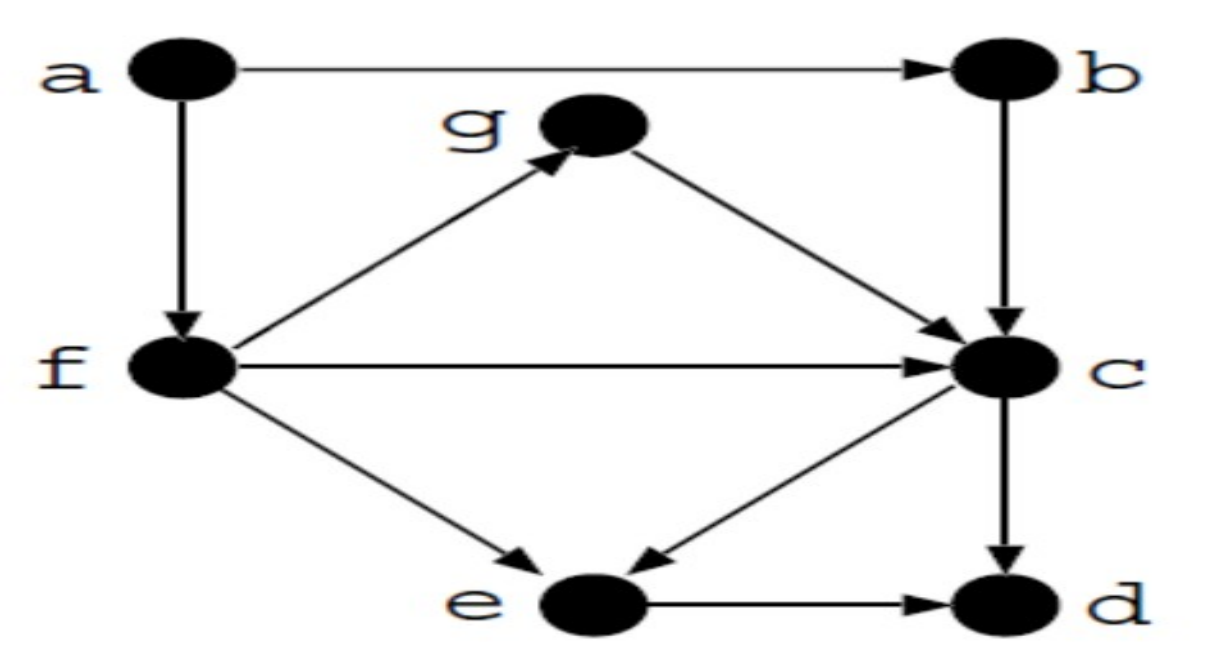
mark(M, Y, mathematic),

sum(S,Y),

P>=85, M>=85, S>=300.

Problem:5

Write a prolog program to find the paths between two nodes in the following graph



Problem:6

Q. We have a seven students and each student has four marks in four subjects(Database, AI, Networks, Programming) as the following:

Omr, 70, 75, 90, 80,

Saied, 65, 70, 60, 80,

Hanaa, 75,70, 60,70,

Nadia, 55,70, 90, 80,

Hany, 85, 90, 90, 68,

Hamdy, 75,70, 80,70,

Marwa, 60, 50, 40, 45

Problem:6

By using Prolog Language, write a KB to answer the following questions:

- **Display the names of successful students (sum of marks>=240).**
- **Display the names of students who obtained 90 in artificial intelligence**
- **Display the names of the students whose score are A.**
- **Display the number of the students whose score are C.**
- **Display the names of failed students in the Database.**