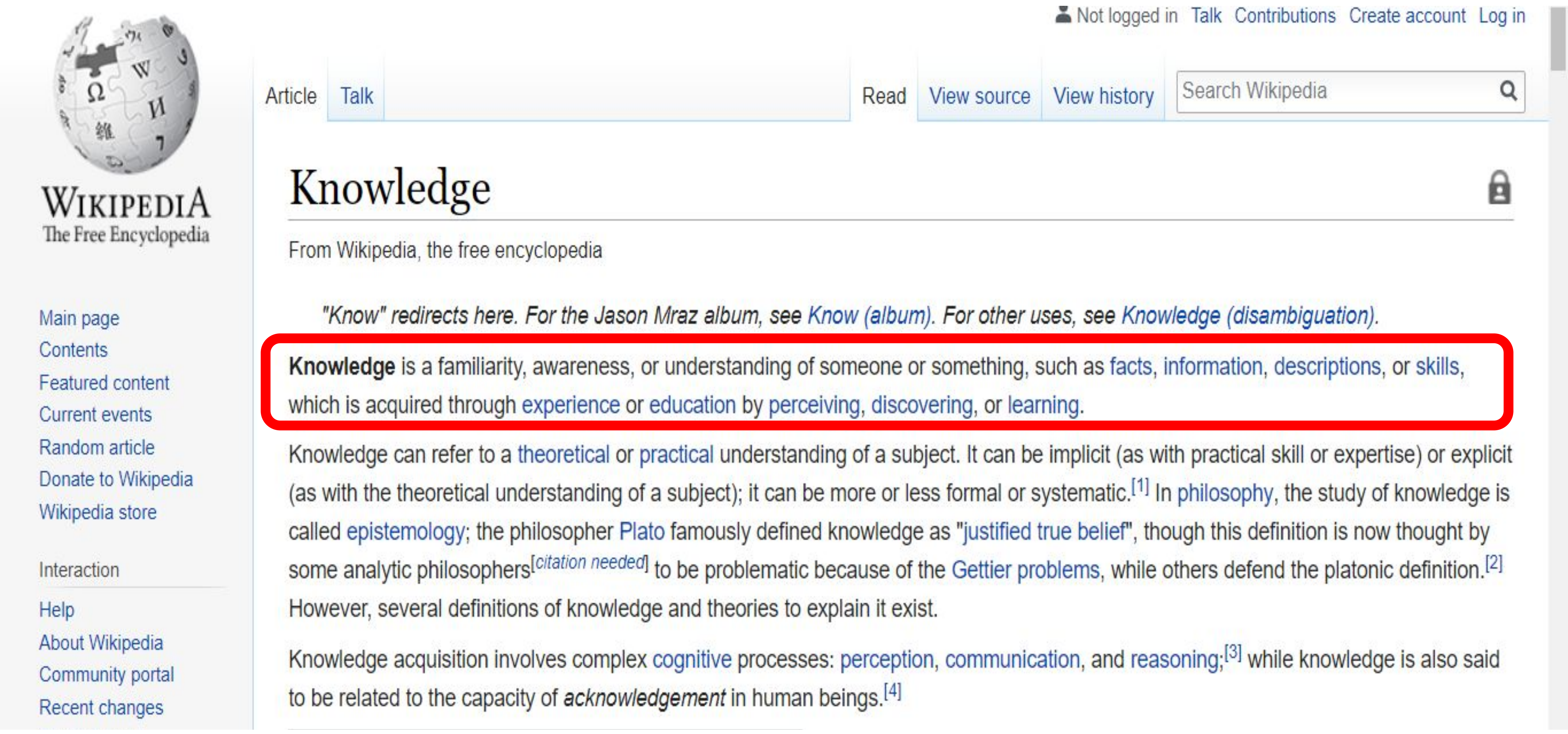
Knowledge Representation



Knowledge

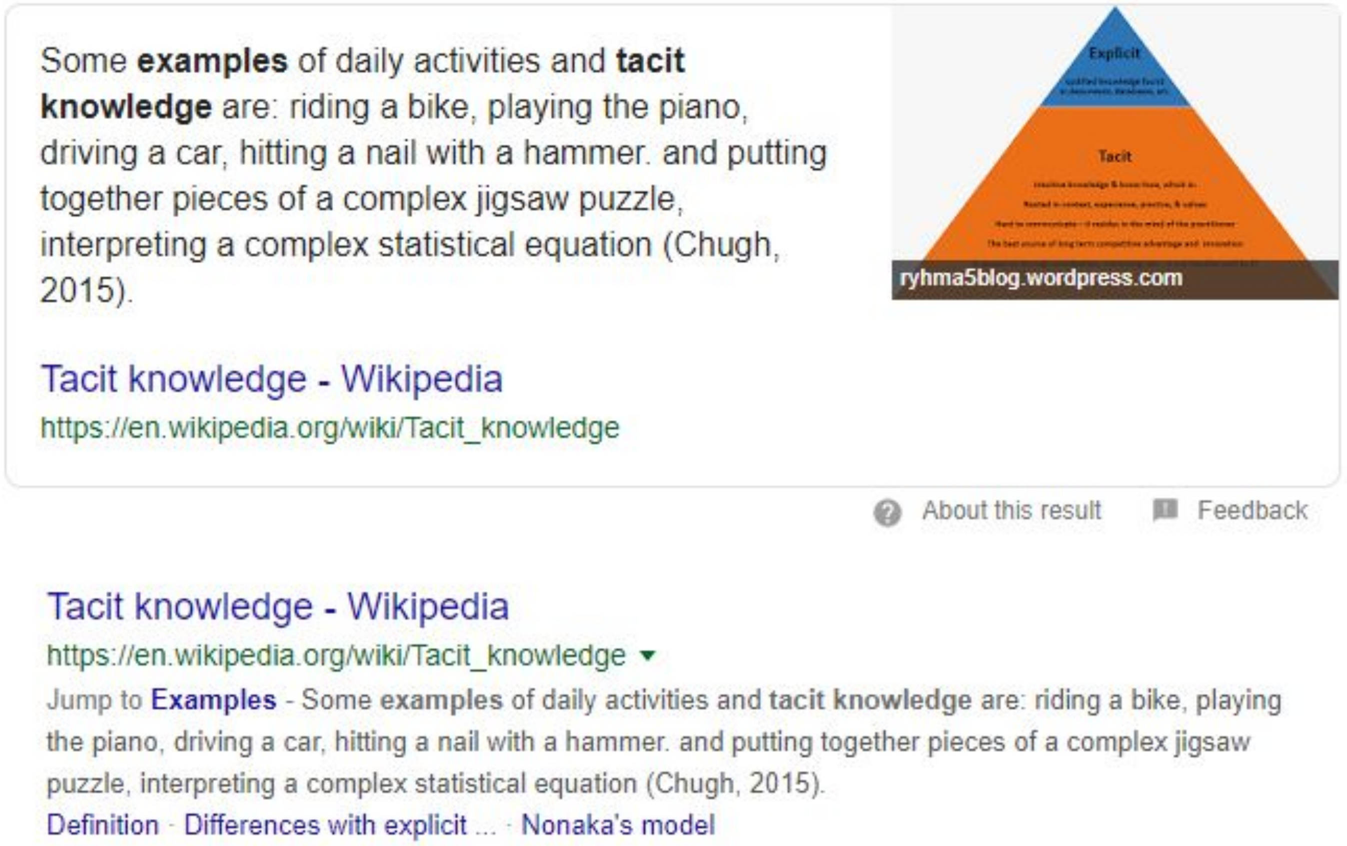
From Wikipedia, the free encyclopedia

"Know" redirects here. For the Jason Mraz album, see Know (album). For other uses, see Knowledge (disambiguation).

Knowledge is a familiarity, awareness, or understanding of someone or something, such as facts, information, descriptions, or skills, which is acquired through experience or education by perceiving, discovering, or learning.

Knowledge can refer to a theoretical or practical understanding of a subject. It can be implicit (as with practical skill or expertise) or explicit (as with the theoretical understanding of a subject); it can be more or less formal or systematic.[1] In philosophy, the study of knowledge is called epistemology; the philosopher Plato famously defined knowledge as "justified true belief", though this definition is now thought by some analytic philosophers[*citation needed*] to be problematic because of the Gettier problems, while others defend the platonic definition.[2] However, several definitions of knowledge and theories to explain it exist.

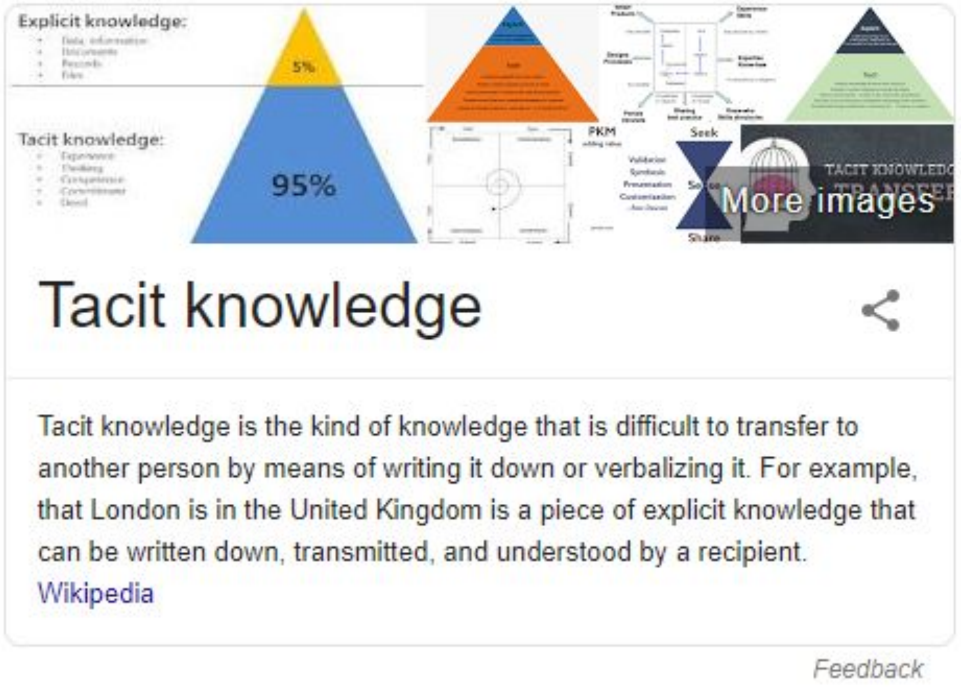
Knowledge acquisition involves complex cognitive processes: perception, communication, and reasoning;[3] while knowledge is also said to be related to the capacity of *acknowledgement* in human beings.[4]



Some **examples** of daily activities and **tacit knowledge** are: riding a bike, playing the piano, driving a car, hitting a nail with a hammer. and putting together pieces of a complex jigsaw puzzle, interpreting a complex statistical equation (Chugh, 2015).

Tacit knowledge - Wikipedia
https://en.wikipedia.org/wiki/Tacit_knowledge

Tacit knowledge - Wikipedia
https://en.wikipedia.org/wiki/Tacit_knowledge ▾
Jump to **Examples** - Some **examples** of daily activities and **tacit knowledge** are: riding a bike, playing the piano, driving a car, hitting a nail with a hammer. and putting together pieces of a complex jigsaw puzzle, interpreting a complex statistical equation (Chugh, 2015).
Definition · Differences with explicit ... · Nonaka's model



Tacit knowledge

Tacit knowledge is the kind of knowledge that is difficult to transfer to another person by means of writing it down or verbalizing it. For example, that London is in the United Kingdom is a piece of explicit knowledge that can be written down, transmitted, and understood by a recipient.
Wikipedia

Knowledge Category

Knowledge is categorized into two major types: *Tacit* and *Explicit*.

- term "**Tacit**" corresponds to *"informal" or "implicit"* type of knowledge,
- term "**Explicit**" corresponds to *"formal"* type of knowledge.

Tacit knowledge	**Explicit knowledge**
◇ Exists within a human being; it is embodied.	◇ Exists outside a human being; it is embedded.
◇ Difficult to articulate formally.	◇ Can be articulated formally.
◇ Difficult to communicate or share.	◇ Can be shared, copied, processed and stored.
◇ Hard to steal or copy.	◇ Easy to steal or copy
◇ Drawn from experience, action, subjective insight.	◇ Drawn from artifact of some type as principle, procedure, process, concepts.

Knowledge is a general term.

Knowledge is a progression that starts with *data* which is of limited utility.

- By organizing or analyzing the data, we understand what the data means, and this becomes *information*.
- The interpretation or evaluation of information yield *knowledge*.
- An understanding of the principles embodied within the knowledge is *wisdom*.

- **Knowledge Progression**

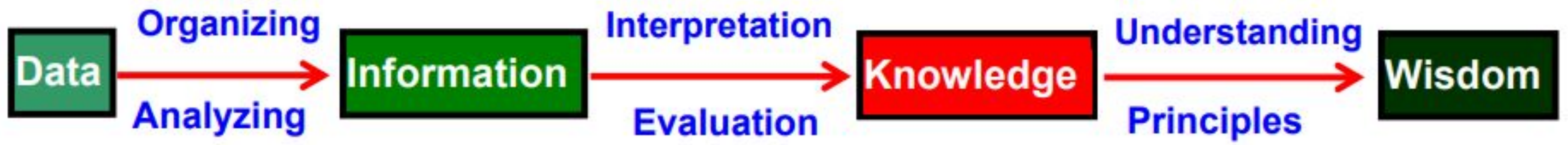
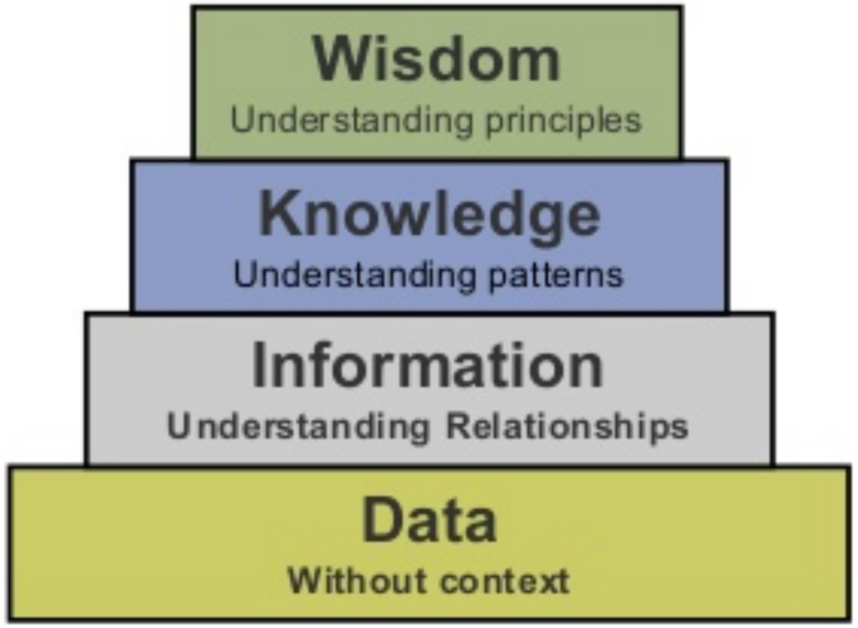


Fig 1 Knowledge Progression

- Take Example of ImageNet Dataset

What is knowledge?

Knowledge exists on a hierarchy.



- Explicit knowledge has been documented and is available for sharing.
- Tacit knowledge exists only in the minds of people.
- Typically 80% of organizational knowledge is tacit.
- When the people leave, so does the knowledge.

Knowledge preservation creates an enduring asset.

How do we Represent what we know ?

- **Knowledge** is a general term.

  An answer to the question, *"how to represent knowledge"*, requires an analysis to distinguish between knowledge *"how"* and knowledge *"that"*.

  - *knowing "how to do something".*

    e.g. "how to drive a car" is a **Procedural knowledge***.*

  - *knowing "that something is true or false".*

    e.g. "that is the speed limit for a car on a motorway" is a **Declarative knowledge.**

- **knowledge and Representation** *are two distinct entities. They play a central but distinguishable roles in intelligent system.*
  - Knowledge is a description of the world. It determines a *system's competence* by what it knows.
  - Representation is the way knowledge is encoded. It defines a *system's performance* in doing something.

Competence - the ability to do something successfully or efficiently

- Different types of knowledge require different kinds of representation. The Knowledge Representation *models/mechanisms* are often based on:
  - ◇ **Logic**
  - ◇ **Rules**
  - ◇ **Frames**
  - ◇ **Semantic Net**

In simple words, we :

- need to know about *things we want to represent* , and
- need some means by which *things we can manipulate*.

- Different types of knowledge require different kinds of *reasoning*.

Framework of Knowledge Representation (Poole 1998)

Computer requires a well-defined problem description to process and provide well-defined acceptable solution.

To collect fragments of knowledge we need first to formulate a description in our spoken language and then represent it in formal language so that computer can understand. The computer can then use an algorithm to compute an answer. This process is illustrated below.

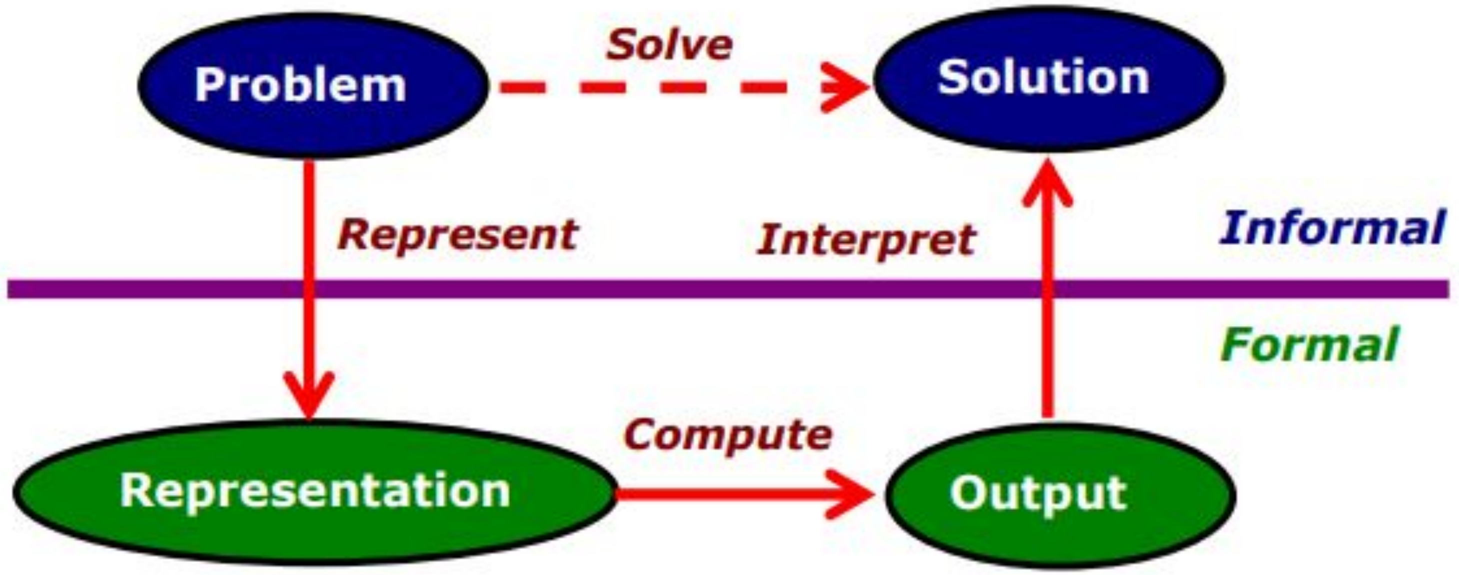


Fig. Knowledge Representation Framework

knowledge representation can be considered at two levels :

(a) *knowledge level* at which facts are described, and

(b) *symbol level* at which the representations of the objects, defined in terms of symbols, can be manipulated in the programs.

Note : A good representation enables fast and accurate access to knowledge and understanding of the content.

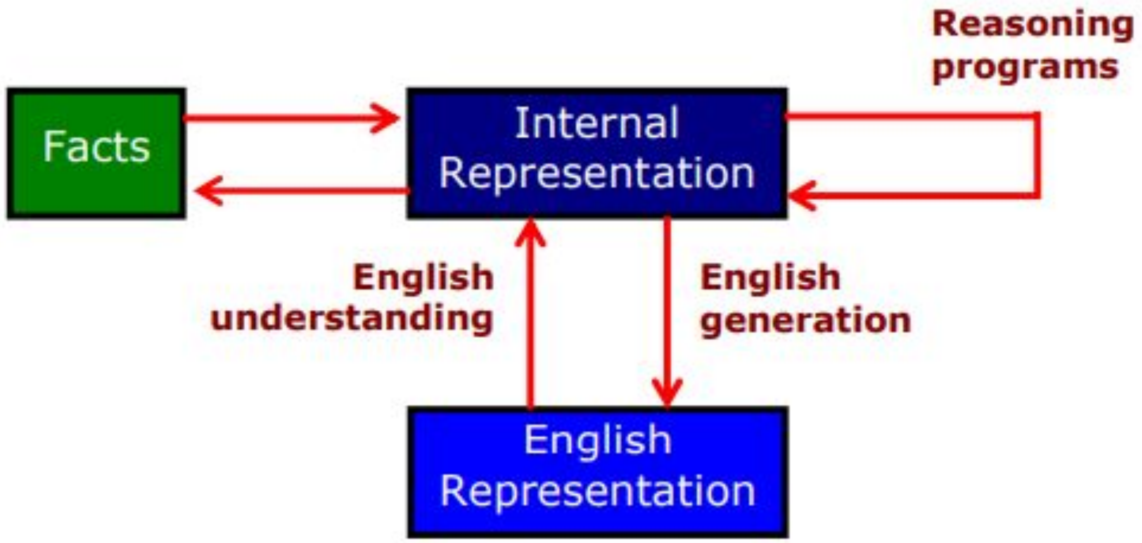
Mapping between Facts and Representation

Knowledge is a collection of *"facts"* from some domain.

We need a representation of *"facts"* that can be manipulated by a program. Normal English is insufficient, too hard currently for a computer program to draw inferences in natural languages.
Thus some symbolic representation is necessary.

Therefore, we must be able to map *"facts to symbols"* and *"symbols to facts"* using *forward and backward representation mapping.*

Example : Consider an English sentence



Facts	Representations
◇ **Spot is a dog**	A *fact* represented in *English sentence*
◇ **dog (Spot)**	Using *forward mapping function* the above *fact* is represented in *logic*
◇ **$\forall$ x : dog(x) $\rightarrow$ hastail (x)**	A *logical representation* of the *fact* that *"all dogs have tails"*

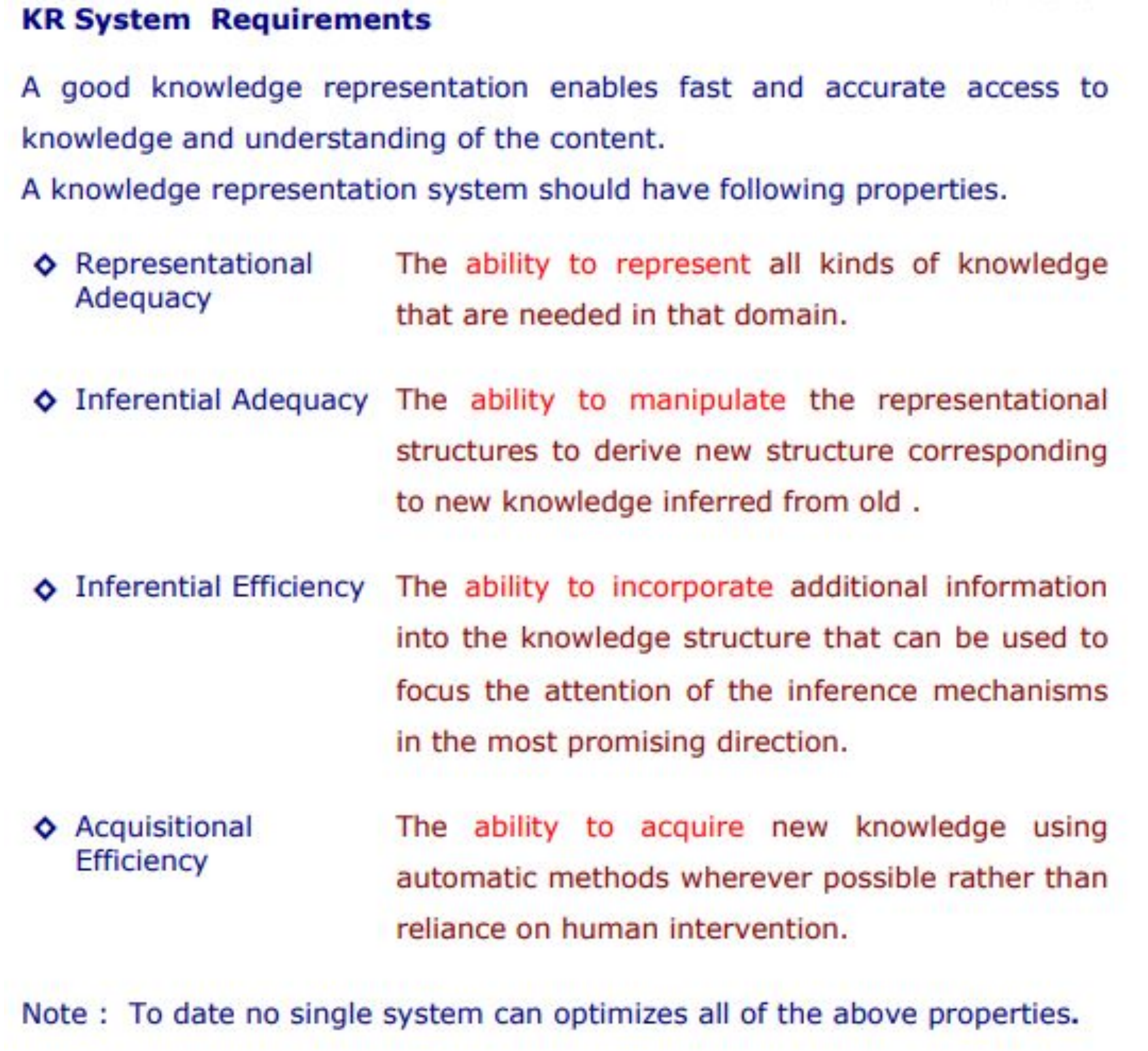
Now using ***deductive mechanism*** we can generate a new representation of object :

◇ **hastail (Spot)**	A new object representation
◇ **Spot has a tail** ***[it is new knowledge]***	Using *backward mapping function* to generate English sentence

Properties of

Adequacy - the fact of being enough/acceptable/sufficient or satisfactory for a particular purpose

Inferential - characterized by or involving conclusions reached on the basis of evidence and reasoning



KR System Requirements

A good knowledge representation enables fast and accurate access to knowledge and understanding of the content.

A knowledge representation system should have following properties.

◇ Representational Adequacy	The ability to represent all kinds of knowledge that are needed in that domain.
◇ Inferential Adequacy	The ability to manipulate the representational structures to derive new structure corresponding to new knowledge inferred from old .
◇ Inferential Efficiency	The ability to incorporate additional information into the knowledge structure that can be used to focus the attention of the inference mechanisms in the most promising direction.
◇ Acquisitional Efficiency	The ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

Note : To date no single system can optimizes all of the above properties.

Approaches to KR/ Types of KR/ Schemes of KR

- Relational Knowledge
- Inheritable Knowledge
- Inferential Knowledge
- Declarative Knowledge

Relational Knowledge :

This knowledge associates elements of one domain with another domain.

- Relational knowledge is made up of objects consisting of attributes and their corresponding associated values.
- The results of this knowledge type is a mapping of elements among different domains.

The table below shows a simple way to store facts.

- The facts about a set of objects are put systematically in columns.
- This representation provides little opportunity for inference.

Table - Simple Relational Knowledge

Player	Height	Weight	Bats - Throws
Aaron	6-0	180	Right - Right
Mays	5-10	170	Right - Right
Ruth	6-2	215	Left - Left
Williams	6-3	205	Left - Right

‡ Given the facts it is not possible to answer simple question such as :

" Who is the heaviest player ? ".

but if a procedure for finding heaviest player is provided, then these facts will enable that procedure to compute an answer.

‡ We can ask things like who "bats – left" and "throws – right".

Inheritable Knowledge :

Here the knowledge elements inherit attributes from their parents.

The knowledge is embodied in the design hierarchies found in the functional, physical and process domains. Within the hierarchy, elements inherit attributes from their parents, but in many cases not all attributes of the parent elements be prescribed to the child elements.
The *inheritance* is a powerful form of inference, but not adequate. The basic KR needs to be augmented with inference mechanism.

The KR in hierarchical structure, shown below, is called *"semantic network"* or a collection of *"frames" or "slot-and-filler structure"*. The structure shows property inheritance and way for insertion of additional knowledge.
Property inheritance : The objects or elements of specific classes inherit attributes and values from more general classes. The classes are organized in a generalized hierarchy.

Baseball knowledge

- *isa : show class inclusion*
- *instance : show class membership*

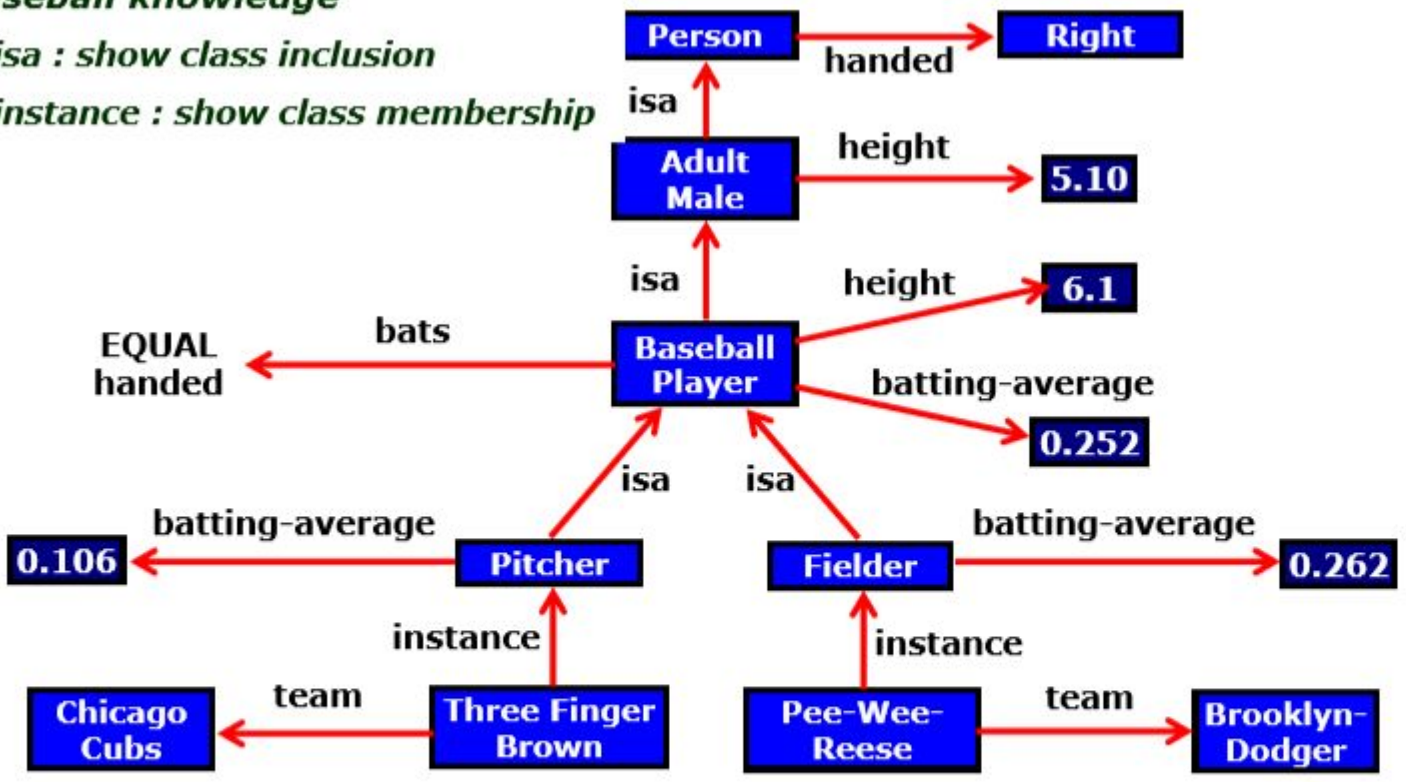


Fig. Inheritable knowledge representation (KR)

‡ The directed arrows represent *attributes* (*isa, instance, team*) originates at object being described and terminates at object or its value.

‡ The box nodes represents *objects* and *values* of the attributes.

- **Viewing a node as a frame**

  Example : Baseball-player

isa :	*Adult-Male*
Bates :	*EQUAL handed*
Height :	*6.1*
Batting-average :	*0.252*

- **Algorithm :** Property Inheritance

  Retrieve a value **V** for an attribute **A** of an instance object **O**.

  Steps to follow:

  1. Find object **O** in the knowledge base.
  2. If there is a value for the attribute **A** then report that value.
  3. Else, if there is a value for the attribute instance; If not, then fail.
  4. Else, move to the node corresponding to that value and look for a value for the attribute **A**; If one is found, report it.
  5. Else, do until there is no value for the "**isa**" attribute or until an answer is found :

     (a) Get the value of the "**isa**" attribute and move to that node.

     (b) See if there is a value for the attribute **A**; If yes, report it.

This algorithm is simple. It describes the basic mechanism of inheritance. It does not say what to do if there is more than one value of the instance or "**isa**" attribute.

This can be applied to the example of knowledge base illustrated, in the previous slide, to derive answers to the following queries :

- team (Pee-Wee-Reese) = Brooklyn–Dodger
- batting–average(Three-Finger-Brown) = 0.106
- height (Pee-Wee-Reese) = 6.1
- bats (Three Finger Brown) = right

Inferential Knowledge :

This knowledge generates new information from the given information.

This new information does not require further data gathering form source, but does require analysis of the given information to generate new knowledge.

Example :

- given a set of relations and values, one may infer other values or relations.
- a predicate logic (a mathematical deduction) is used to infer from a set of attributes.
- inference through predicate logic uses a set of logical operations to relate individual data.
- the symbols used for the logic operations are :

  " → " (implication), " ¬ " (not), " ∨ " (or), " ∧ " (and), " ∀ " (for all), " ∃ " (there exists).

Examples of predicate logic statements :

1. *"Wonder"* is a name of a dog : **dog (wonder)**
2. All dogs belong to the class of animals : **$\forall$ x : dog (x) $\rightarrow$ animal(x)**
3. All animals either live on land or in water : **$\forall$ x : animal(x) $\rightarrow$ live (x, land) V live (x, water)**

From these three statements we can infer that :

" *Wonder* lives either on land or on water."

Note : If more information is made available about these objects and their relations, then more knowledge can be inferred.

Declarative/Procedural Knowledge

Declarative knowledge :

Here, the knowledge is based on declarative facts about *axioms* and *domains* .

- axioms are assumed to be true unless a counter example is found to invalidate them.
- domains represent the physical world and the perceived functionality.
- axiom and domains thus simply exists and serve as declarative statements that can stand alone.

Procedural knowledge:

Here, the knowledge is a mapping process between domains that specify **"what to do when"** and the representation is of **"how to make it"** rather than *"what it is"*. The procedural knowledge :

- may have inferential efficiency, but no inferential adequacy and acquisitional efficiency.
- are represented as small programs that know how to do specific things, how to proceed.

Example : A parser in a natural language has the knowledge that a noun phrase may contain articles, adjectives and nouns. It thus accordingly call routines that know how to process articles, adjectives and nouns.

Issues in Knowledge Representation

- The fundamental goal of Knowledge Representation is to facilitate inference (conclusions) from knowledge.
- The issues that arise while using KR techniques are many. Some of these are explained below.

1. **Important Attributes :**
    - Any attribute of objects so basic that they occur in almost every problem domain?
    - There are two attributes "instance" and "isa", that are of general significance. These attributes are important because they support property inheritance.

2. **Relationship among attributes:**
    - Any important relationship that exists among object attributes?
    - The attributes we use to describe objects are themselves entities that we represent.
    - The relationship between the attributes of an object, independent of specific knowledge they encode, may hold properties like:
        i. Inverses - This is about consistency check, while a value is added to one attribute. The entities are related to each other in many different ways.
        ii. Existence in an *isa* hierarchy - This is about generalization-specialization, like, classes of objects and specialized subsets of those classes, there are attributes and specialization of attributes. For example, the attribute height is a specialization of general attribute physical-size which is, in turn, a specialization of physical-attribute. These generalization-specialization relationships are

important for attributes because they support inheritance.

iii. Techniques for reasoning about values - This is about reasoning values of attributes not given explicitly. Several kinds of information are used in reasoning, like,

    height : must be in a unit of length,

    Age: of person cannot be greater than the age of person's parents.

    The values are often specified when a knowledge base is created.

iv. Single valued attributes - This is about a specific attribute that is guaranteed to take a unique value. For example, a baseball player can at time have only a single height and be a member of only one team. KR systems take different approaches to provide support for single valued attributes.

3. **Choosing Granularity :**
   - At what level of detail should the knowledge be represented?
   - Regardless of the KR formalism, it is necessary to know :
     - At what level should the knowledge be represented and what are the primitives?"
     - Should there be a small number or should there be a large number of low-level primitives or High-level facts.
     - High-level facts may not be adequate for inference while Low-level primitives may require a lot of storage.
   - Example of Granularity :
     - Suppose we are interested in following facts:

       John spotted Sue.

       This could be represented as

       Spotted (agent(John), object (Sue))

       – Such a representation would make it easy to answer questions such are :
       - Who spotted Sue?

       Suppose we want to know :
       - Did John see Sue?

       – Given only one fact, we cannot discover that answer.

       – We can add other facts, such as

       Spotted (x , y) → saw (x , y)

       – We can now infer the answer to the question.

4. Set of objects :

- How should sets of objects be represented?
- There are certain properties of objects that are true as member of a set but not as individual;
  - Example : Consider the assertion made in the sentences :

    "there are more sheep than people in Australia", and

    "English speakers can be found all over the world."

- To describe these facts, the only way is to attach assertion to the sets representing people, sheep, and English.
- The reason to represent sets of objects is: If a property is true for all or most elements of a set, then it is more efficient to associate it once with the set rather than to associate it explicitly with every elements of the set .
- This is done,
  - in logical representation through the use of universal quantifier, and
  - in hierarchical structure where node represent sets and inheritance propagate set level assertion down to individual.

5. Finding Right structure :

- Given a large amount of knowledge stored in a database, how can relevant parts are accessed when they are needed?
- This is about access to right structure for describing a particular situation.
- This requires, selecting an initial structure and then revising the choice.
- While doing so, it is necessary to solve following problems :
    - how to perform an initial selection of the most appropriate structure.
    - how to fill in appropriate details from the current situations.
    - how to find a better structure if the one chosen initially turns out not to be appropriate.
    - what to do if none of the available structures is appropriate.
    - when to create and remember a new structure.
- There is no good, general purpose method for solving all these problems. Some knowledge representation techniques solve some of these issues.

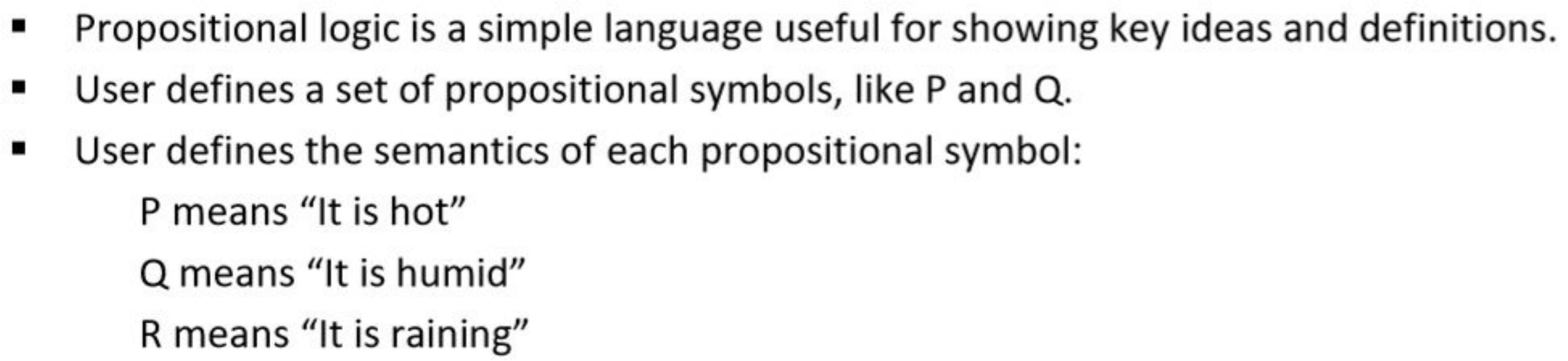
Proposition

- A proposition is a statement, or a simple declarative sentence.
- For example, “the book is expensive” is a proposition.
- A proposition can be either true or false. **But not both**

Propositional logic

- Logical constants: true, false
- Propositional symbols: P, Q, S,... (atomic sentences)
- Propositions are combined by connectives:

Connective	*Symbols*					**Read as**
assertion	**P**					**"p is true"**
negation	**¬p**	**~**	**!**		**NOT**	**"p is false"**
conjunction	**p ∧ q**	**·**	**&&**	**&**	**AND**	**"both p and q are true"**
disjunction	**P v q**	**||**	**|**		**OR**	**"either p is true, or q is true, or both "**
implication	**p → q**	**⊃**	**⇒**		**if ..then**	**"if p is true, then q is true"** **" p implies q "**
equivalence	**↔**	**≡**	**⇔**		**if and only if**	**"p and q are either both true or both false"**



- Propositional logic is a simple language useful for showing key ideas and definitions.
- User defines a set of propositional symbols, like P and Q.
- User defines the semantics of each propositional symbol:

  P means “It is hot”

  Q means “It is humid”

  R means “It is raining”

If it is humid then it is hot.

Q ⍰ P

If it is hot and humid then it is not raining.

P ^ Q ⍰ ¬R

- All cars have 4 wheels.
- Some auto have 4 wheels.
  - X1, X2,..Xn (Not possible PL)

1. The sun rises in the East and sets in the West.
2. 1 + 1 = 2
3. 'b' is a vowel.

All of the above sentences are propositions, where the first two are Valid(True) and the third one is Invalid(False).

1. What time is it?
2. Go out and play.
3. x + 1 = 2.

The above sentences are not propositions as the first two do not have a truth value, and the third one may be true or false.

- Consider the following two statements:

  Every CSE student must study discrete mathematics.

  Jackson is a CSE student.
- It looks "logical" to deduce that therefore, Jackson must study discrete mathematics.
- However, this cannot be expressed by propositional logic...you may try it, but you can already notice that none of the logical operators we have learnt are applicable here.
- We need new tools! ***Predicate Logic***

Predicate Logic

The propositional logic, is not powerful enough for all types of assertions;

Example : The assertion **"x > 1"**, where **x** is a variable, is not a proposition because it is neither true nor false unless value of **x** is defined.

For **x > 1** to be a proposition ,

- either we substitute a specific number for **x** ;
- or change it to something like

  "There is a number x for which x > 1 holds";
- or **"For every number x, x > 1 holds"**.

All men are mortal.
Socrates is a man.
∴Socrates is mortal.

These cannot be expressed in propositional logic as a finite and logically valid argument (formula).

We need languages : that allow us to describe properties (*predicates*) of objects, or a relationship among objects represented by the variables .

Predicate logic satisfies the requirements of a language.

- *Predicate logic* is powerful enough for expression and reasoning.
- *Predicate logic* is built upon the ideas of *propositional logic.*

- **Predicate :**

Every complete "sentence" contains two parts : a "subject" and a "predicate".

The *subject* is what (or whom) the sentence is about.

The *predicate* tells something about the subject;

Example :

A *sentence* **"Judy {runs}".**

The subject is **Judy** and the predicate is **runs** .

Predicate, always includes verb, tells something about the subject.

Predicate is a verb phrase template that describes a property of objects, or a relation among objects represented by the variables.

Example:

"The car Tom is driving *is blue*" ;

"The sky *is blue*" ;

"The cover of this book *is blue*"

Predicate is **"is blue"** , describes property.

Predicates are given names; Let **'B'** is name for predicate **"is_blue"**.

Sentence is represented as **"B(x)"** , read as **"x is blue"**;

Symbol **"x"** represents an arbitrary Object .

Example: Nate is a student at UT.
What is the subject? What is the predicate?

- Manay is tall.
  - tall(manay).

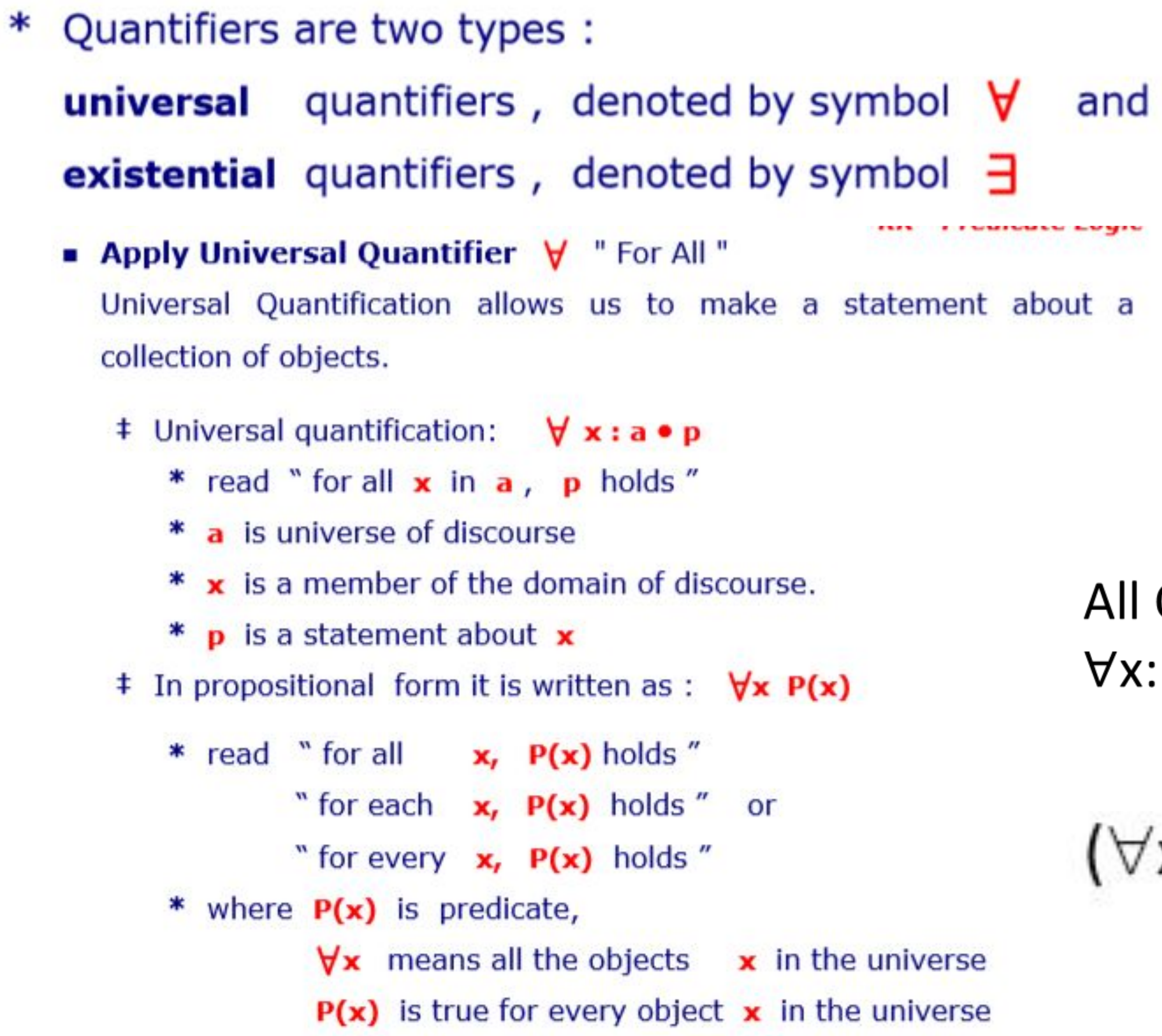
- Cat is black.
  - Cat(x) ⍰ Black(x)

Definition: A predicate is a property that a variable or a finite collection of variables can have. A predicate becomes a proposition when specific values are assigned to the variables. P(x1, x2, ..., xn) is called a predicate of n variables or n arguments.
Example: She lives in the city.
P(x,y): x lives in y.
P(Mary, Austin) is a proposition: Mary lives in Austin

- Some CSE boys like programming.
    - like(girls, non-veg)

- All CSE boys like programming.
    - like(CSE_boys, cricket)

    - Here meaning is totally different but predicate is same. WHICH IS NOT VALID.



* Quantifiers are two types :

universal quantifiers , denoted by symbol $\forall$ and

existential quantifiers , denoted by symbol $\exists$

- **Apply Universal Quantifier** $\forall$ " For All "

  Universal Quantification allows us to make a statement about a collection of objects.

  ‡ Universal quantification: $\forall\ \mathbf{x : a \bullet p}$

  * read " for all **x** in **a** , **p** holds "

  * **a** is universe of discourse

  * **x** is a member of the domain of discourse.

  * **p** is a statement about **x**

  ‡ In propositional form it is written as : $\forall \mathbf{x}\ \mathbf{P(x)}$

  * read " for all **x**, **P(x)** holds "

  " for each **x**, **P(x)** holds " or

  " for every **x**, **P(x)** holds "

  * where **P(x)** is predicate,

  $\forall$**x** means all the objects **x** in the universe

  P(x) is true for every object **x** in the universe

All CSE boys like programming.

$\forall$x: CSE_boys(x) → like(x, programming)

$$(\forall x)\ \text{dolphin}(x) \rightarrow\ \text{mammal}(x)$$

- **Apply Existential Quantifier ∃** " There Exists "

Existential Quantification allows us to state that an object does exist without naming it.

‡ Existential quantification: **∃ x : a • p**

- * read " there exists an **x** such that **p** holds "
- * **a** is universe of discourse
- * **x** is a member of the domain of discourse.
- * **p** is a statement about **x**

‡ In propositional form it is written as : **∃ x P(x)**

- * read " there exists an **x** such that **P(x)** " or
  " there exists at least one **x** such that **P(x)** "
- * Where **P(x)** is predicate
  ∃x means at least one object **x** in the universe
  P(x) is true for least one object **x** in the universe

Some girls like non-veg.

∃x: girls(x) ^ like(x, non-veg)

$(\exists\, x)\ \text{mammal}(x) \wedge \text{lays-eggs}(x)$

- **1. All birds fly.**
  In this question the predicate is "**fly(bird)**."
  And since there are all birds who fly so it will be represented as follows.
  ∀x bird(x) →fly(x).
- **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:
  ∀x man(x) → respects (x, parent).
- **3. Some boys play cricket.**
  In this question, the predicate is "**play(x, y)**," where x= boys, and y= game. Since there are some boys so we will use ∃**, and it will be represented as**:
  ∃x boys(x) → play(x, cricket).
- **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:
  ¬∀ (x) [student(x) → like(x, Mathematics) ∧ like(x, Science)].
- **5. Only one student failed in Mathematics.**
  In this question, 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:
  ∃(x) [student(x) → failed (x, Mathematics) ∧∀ (y) [¬(x==y) ∧ student(y) → ¬failed (x, Mathematics)].

- Marcus was a man
  - Man(Marcus)
- Marcus was a Pompeian
  - Pompeian(Marcus)
- All Pompeians were Romans
  - ∀x [Pompeian(x) □ Roman(x)]
- Caesar was a ruler
  - Ruler(Caesar)
- All Romans were either loyal to Caesar or hated him
  - ∀x [Roman(y) □ (LoyalTo(x,Caesar) ∨ Hate(x,Caesar))]
- People only try to assassinate rulers they aren't loyal to
  - ∀x∀y[(Person(x) ∧ Ruler(y) ∧ TryAssassinate(x,y)) □¬LoyalTo(x,y)]
- Marcus tried to assassinate Caesar
  - TryAssassinate(Marcus, Caesar)
- Everyone is loyal to someone
  - ∀x ∃y LoyalTo(x,y)