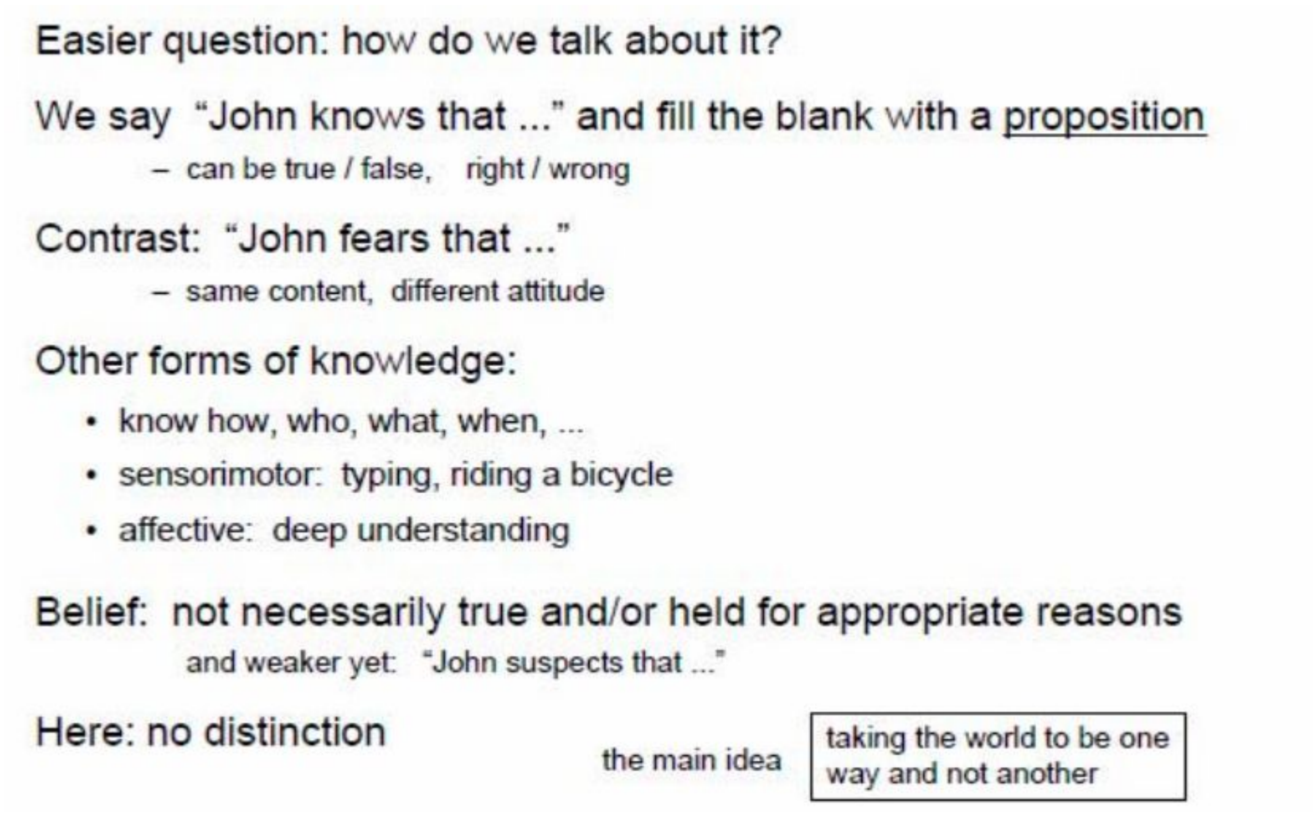
Knowledge, Reasoning, and Planning

Course Code: AFI 124

Knowledge



Easier question: how do we talk about it?

We say "John knows that ..." and fill the blank with a proposition

– can be true / false, right / wrong

Contrast: "John fears that ..."

– same content, different attitude

Other forms of knowledge:

- know how, who, what, when, ...
- sensorimotor: typing, riding a bicycle
- affective: deep understanding

Belief: not necessarily true and/or held for appropriate reasons

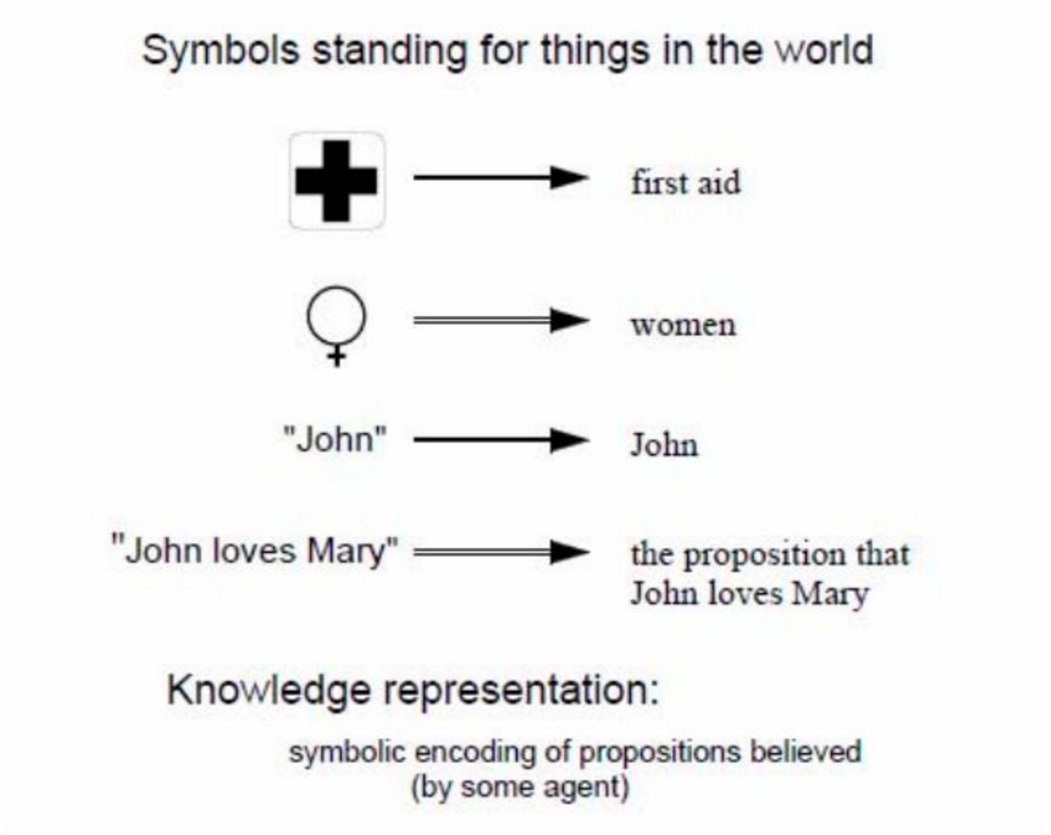
and weaker yet: "John suspects that ..."

Here: no distinction

the main idea

taking the world to be one way and not another

Representation

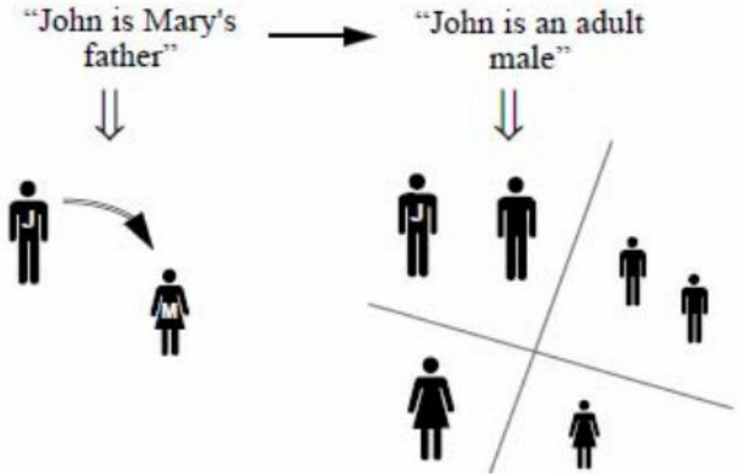

Symbols standing for things in the world
first aid
women
"John"
John
"John loves Mary"
the proposition that John loves Mary
Knowledge representation:
symbolic encoding of propositions believed (by some agent)

Reasoning

Manipulation of symbols encoding propositions to produce representations of new propositions

Analogy: arithmetic

$$\begin{array}{ccccc} \text{“1011”} & + & \text{“10”} & \rightarrow & \text{“1101”} \\ \Downarrow & & \Downarrow & & \Downarrow \\ \text{eleven} & & \text{two} & & \text{thirteen} \end{array}$$



Benefits of Reasoning

- Given
  - Patient X allergic to medication M
  - Anyone allergic to medication M is also allergic to medication M’
- Reasoning helps us derive
  - Patient X is allergic to medication M’

Logical Agents

Agents with some representation of complex knowledge about the world/environment and uses inference(with the help of old knowledge, taking new inputs and generating new concepts) to deliver new information from the knowledge combined with new inputs.

Knowledge based agents

Knowledge base is a set of sentences in a formal language representing facts of the world.

- Intelligent agents need knowledge about the world to choose good actions/decisions.
- Knowledge = {sentences} in a knowledge representation language
- A sentence is an assertion about the world
- A knowledge based agent is composed of:
    - Knowledge based
    - Inference mechanism

Two examples

Example 1

```
printColour(snow) :- !, write("It's white.").
printColour(grass) :- !, write("It's green.").
printColour(sky) :- !, write("It's yellow.").
printColour(X) :- write("Beats me.").
```

Example 2

```
printColour(X) :- colour(X,Y), !,
        write("It's "), write(Y), write(".").
printColour(X) :- write("Beats me.").

colour(snow,white).
colour(sky,yellow).
colour(X,Y) :- madeof(X,Z), colour(Z,Y).
madeof(grass,vegetation).
colour(vegetation,green).
```

Only the 2nd has a separate collection of symbolic structures à la KR Hypothesis

its <u>knowledge base</u> (or KB)

∴ a small knowledge-based system

Knowledge Representation Systems

Knowledge Representation in AI describes the representation of knowledge. Basically, it is a study of how the beliefs, intentions, and judgments of an intelligent agent can be expressed suitably for automated reasoning. One of the primary purposes of Knowledge Representation includes modeling intelligent behavior for an agent.

Knowledge Representation and Reasoning (KR, KRR) represents information from the real world for a computer to understand and then utilize this knowledge to solve complex real-life problems like communicating with human beings in natural language. Knowledge representation in AI is not just about storing data in a database, it allows a machine to learn from that knowledge and behave intelligently like a human being.

Properties of Knowledge Representation Systems

The following properties should be possessed by a knowledge representation system.

Representational Adequacy

-- the ability to represent the required knowledge;

Inferential Adequacy

- the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original;

Inferential Efficiency

- the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides;

Acquisitional Efficiency

- the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

Types of Knowledge Representation Systems

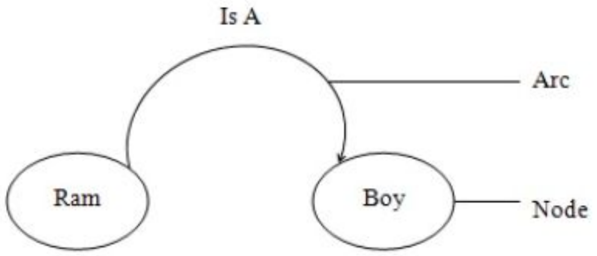
1. Semantic Nets,
2. Frames,
3. Conceptual Dependencies
4. Scripts
5. Rule Based Systems

Types of Knowledge Representation Systems: Semantic Nets

A semantic network allows you to store knowledge in the form of a graphic network with nodes and arcs representing objects and their relationships. It could represent physical objects or concepts or even situations. A semantic network is generally used to represent data or reveal structure. It is also used to support conceptual editing and navigation.

A semantic network is simple and easy to implement and understand. It is more natural than logical representation. It allows you to categorize objects in various forms and then link those objects. It also has greater expressiveness than logic representation.

For example: Ram is a boy.

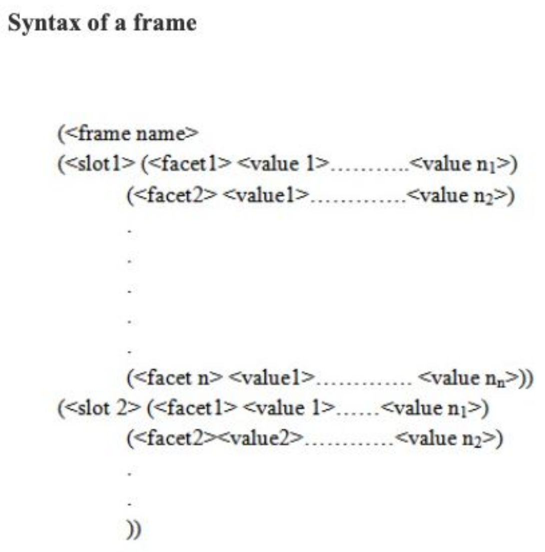


Figure

Types of Knowledge Representation Systems: Frames

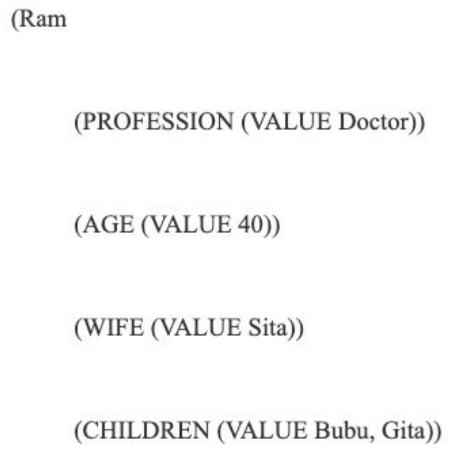
A frame is a collection of attributes and its associated values, which describes an entity in the real world. It is a record like structure consisting of slots and its values. Slots could be of varying sizes and types. These slots have names and values. Or they could have subfields named as facets. They allow you to put constraints on the frames.

There is no restraint or limit on the value of facets a slot could have, or the number of facets a slot could have or the number of slots a frame could have. Since a single frame is not very useful, building a frame system by collecting frames that are connected to each other will be more beneficial. It is flexible and can be used by various AI applications.



Syntax of a frame

(<frame name>
(<slot1> (<facet1> <value 1>..........<value n_1>)
    (<facet2> <value1>............<value n_2>)
    .
    .
    .
    .
    .
    (<facet n> <value1>............ <value n_n>))
(<slot 2> (<facet1> <value 1>......<value n_1>)
    (<facet2><value2>...........<value n_2>)
    .
    .
    ))



(Ram

(PROFESSION (VALUE Doctor))

(AGE (VALUE 40))

(WIFE (VALUE Sita))

(CHILDREN (VALUE Bubu, Gita))

Types of Knowledge Representation Systems: Conceptual Dependencies

Conceptual Graph: It is a knowledge representation technique which consists of basic concepts and the relationship between them. As the name indicates, it tries to capture the concepts about the events and represents them in the form of a graph. A concept may be individual or generic. An individual concept has a type field followed by a reference field. For example person : Ram. Here person indicates type and Ram indicates reference.

"Ram is eating an apple "

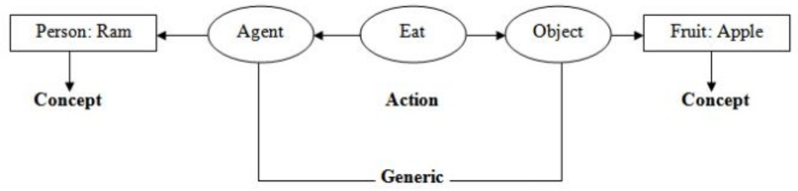
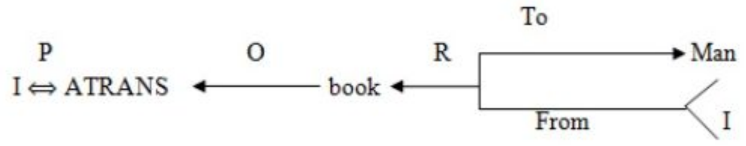


Figure Graphical Representation

$$[\text{Person: Ram}] \leftarrow (\text{Agent}) \leftarrow [\text{Eat}] \rightarrow (\text{Object}) \rightarrow [\text{Fruit: Apple}]$$

Conceptual Dependency: It is an another knowledge representation technique in which we can represent any kind of knowledge. It is based on the use of a limited number of primitive concepts and rules of formation to represent any natural language statement. Conceptual dependency theory is based on the use of knowledge representation methodology was primarily developed to understand and represent natural language structures. The conceptual dependency structures were originally developed by Roger C SChank in 1977.

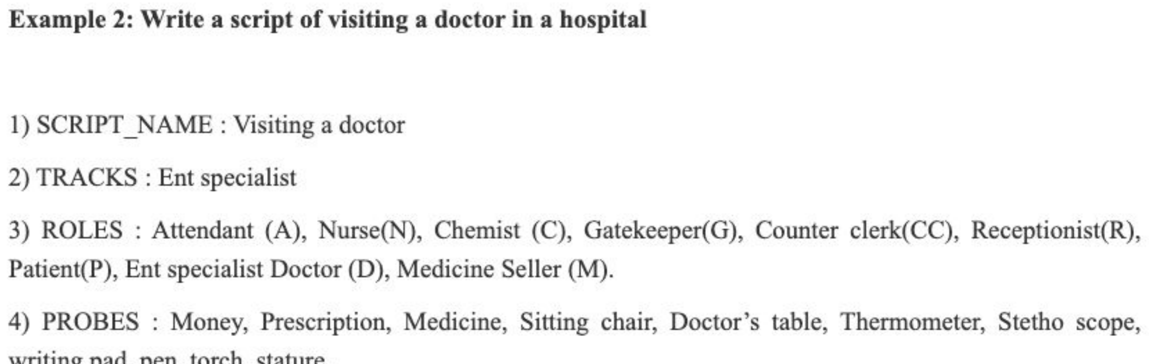
For example



Types of Knowledge Representation Systems: Scripts

It is an another knowledge representation technique. Scripts are frame like structures used to represent commonly occurring experiences such as going to restaurant, visiting a doctor. A script is a structure that describes a stereotyped sequence of events in a particular context. A script consist of a set of slots. Associated with each slot may be some information about what kinds of values it may contain as well as a default value to be used if no other information is available. Scripts are useful because in the real world, there are no patterns to the occurrence of events. These patterns arise because of clausal relationships between events. The events described in a script form a giant casual chain. The beginning of the chain is the set of entry conditions which enable the first events of the script to occur. The end of the chain is the set of results which may enable later events to occur. The headers of a script can all serve as indicators that the script should be activated.

Once a script has been activated, there are a variety of ways in which it can be useful in interpreting a particular situation. A script has the ability to predict events that has not explicitly been observed. An important use of scripts is to provide a way of building a single coherent interpretation from a collection of observation. Scripts are less general structures than are frames and so are not suitable for representing all kinds of knowledge. Scripts are very useful for representing the specific kinds of knowledge for which they were designed.



Example 2: Write a script of visiting a doctor in a hospital

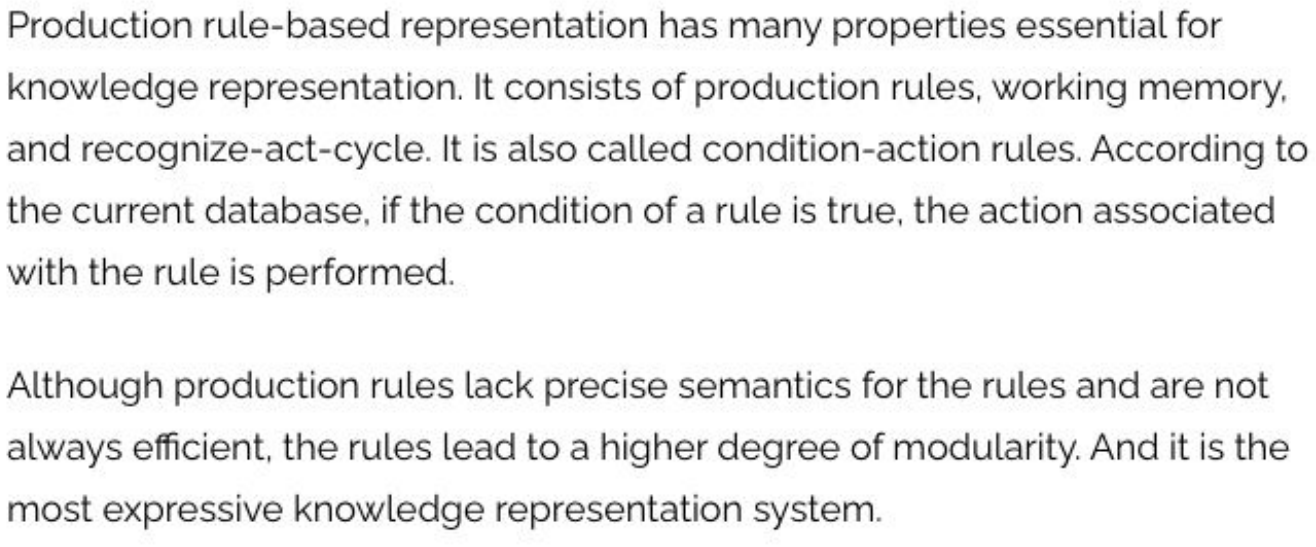
1) SCRIPT_NAME : Visiting a doctor

2) TRACKS : Ent specialist

3) ROLES : Attendant (A), Nurse(N), Chemist (C), Gatekeeper(G), Counter clerk(CC), Receptionist(R), Patient(P), Ent specialist Doctor (D), Medicine Seller (M).

4) PROBES : Money, Prescription, Medicine, Sitting chair, Doctor's table, Thermometer, Stetho scope, writing pad, pen, torch, stature.

Types of Knowledge Representation Systems: Rule Based Systems



Production rule-based representation has many properties essential for knowledge representation. It consists of production rules, working memory, and recognize-act-cycle. It is also called condition-action rules. According to the current database, if the condition of a rule is true, the action associated with the rule is performed.

Although production rules lack precise semantics for the rules and are not always efficient, the rules lead to a higher degree of modularity. And it is the most expressive knowledge representation system.

Propositional Logic

A proposition is a declarative sentence with a truth value. Truth value can be either true or untrue, but it must be assigned to one of the discrete choices and not be unclear. The goal of using predicate logic is to analyze a statement, either individually or in aggregate.
In simple Terms,

- The literal meaning of a proposition is to put across one’s views, ideas, suggestions, expression or judgment. The proposition can be done through a formal document or oral communication (Informal). It can either address a positive or negative connotation.
- A proposition in logic includes Boolean functionalities in a sentence to make it either True or False and also adds reasoning techniques and proofing methods to make it much more comprehensive. This logic is a very old and widely adopted one.
- This logic was readily embraced by the modern search algorithm in Artificial Intelligence applications and Computer-aided tools. It’s use cases in AI include planning, decision making, smart control, diagnosis and problem-solving areas in Business, Medical, Education fields.

Propositional Logic

- Syntax,
- Semantics,
- Formal logic -connectives,
- truth tables,
- tautology,
- validity,
- well -formed -formula,
- Inference using Resolution,
- Backward Chaining and Forward Chaining

Propositional Logic:Syntax

The **syntax** of propositional logic defines the allowable sentences. The **atomic sentences** consist of a single **proposition symbol**. Each such symbol stands for a proposition that can be true or false. We use symbols that start with an uppercase letter and may contain other letters or subscripts, for example: , , , , and *FacingEast*. The names are arbitrary but are often chosen to have some mnemonic value.

There are two proposition symbols with fixed meanings: *True* is the always-true proposition and *False* is the always-false proposition.

^, v, →, ↔, ¬ are used to represent AND, OR,Implies, bi-conditional and NOT condition.

Propositional Logic:Semantics

The semantics defines the rules for determining the truth of a sentence with respect to a particular model. In propositional logic, a model simply sets the **truth value**—*true* or *false*— for every proposition symbol. For example, if the sentences in the knowledge base make use of the proposition symbols , and , then one possible model is

$$m_1 = \{P_{1,2} = false, P_{2,2} = false, P_{3,1} = true\} .$$

Forward Chaining and Back

Forward Chaining and Backward Chaining are the two most important strategies in the field of Artificial Intelligence and lie in the Expert System Domain of AI. Forward and Backward chaining is the strategies used by the Inference Engine in making the deductions.

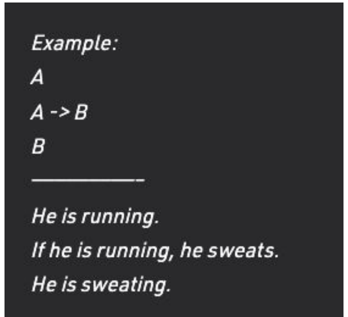
Inference Engine:

Inference Engine is a component of the expert system that applies logical rules to the knowledge base to deduce new information. It interprets and evaluates the facts in the knowledge base in order to provide an answer.

A knowledgebase is a structured collection of facts about the system's domain.

Forward Chaining:

Forward Chaining the Inference Engine goes through all the facts, conditions and derivations before deducing the outcome i.e When based on available data a decision is taken then the process is called as Forwarding chaining, It works from an initial state and reaches to the goal(final decision).



Example:
A
A -> B
B
———————–
He is running.
If he is running, he sweats.
He is sweating.

Backward Chaining

In this, the inference system knows the final decision or goal, this system starts from the goal and works backwards to determine what facts must be asserted so that the goal can be achieved, i.e it works from goal(final decision) and reaches the initial state.

Example:

B

A -> B

A

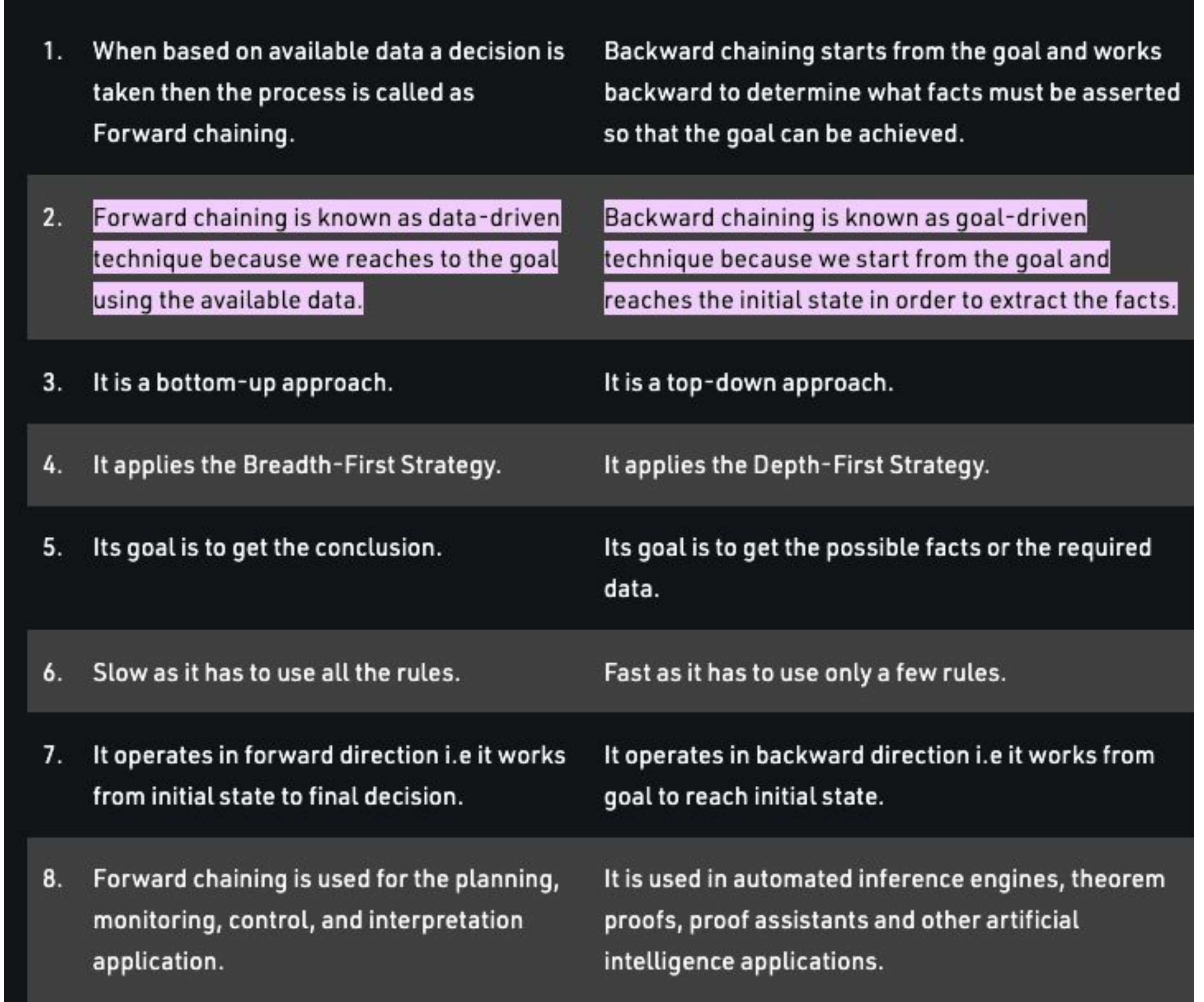
____

He is sweating.

If he is running, he sweats.

He is running.

Difference between Forward and Backward Chaining



1.	When based on available data a decision is taken then the process is called as Forward chaining.	Backward chaining starts from the goal and works backward to determine what facts must be asserted so that the goal can be achieved.
2.	Forward chaining is known as data-driven technique because we reaches to the goal using the available data.	Backward chaining is known as goal-driven technique because we start from the goal and reaches the initial state in order to extract the facts.
3.	It is a bottom-up approach.	It is a top-down approach.
4.	It applies the Breadth-First Strategy.	It applies the Depth-First Strategy.
5.	Its goal is to get the conclusion.	Its goal is to get the possible facts or the required data.
6.	Slow as it has to use all the rules.	Fast as it has to use only a few rules.
7.	It operates in forward direction i.e it works from initial state to final decision.	It operates in backward direction i.e it works from goal to reach initial state.
8.	Forward chaining is used for the planning, monitoring, control, and interpretation application.	It is used in automated inference engines, theorem proofs, proof assistants and other artificial intelligence applications.

To be Contd..

Predicate Logic

in propositional logic, we can only represent the facts, which are either true or false. PL is not sufficient to represent the complex sentences or natural language statements. so we required some more powerful logic, such as first-order logic.

The basic syntactic elements of first-order logic are the symbols that stand for objects, relations, and functions. The symbols, therefore, come in three kinds:

- **constant symbols**, which stand for objects;
- **predicate symbols**, which stand for relations; and
- **function symbols**, which stand for functions.

NOTE: (First-order logic—also known as predicate logic)

Predicate Logic also known as First Order Logic

First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:

- **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, wumpus, ...
- **Relations: It can be unary relation such as:** red, round, is adjacent, **or n-any relation such as:** the sister of, brother of, has color, comes between
- **Function:** Father of, best friend, third inning of, end of, ...

Predicate Logic has two such quantifiers: **∀ (the universal quantifier) and ∃ (the existential quantifier)**. Since a predicate can combine with more than one variable, it is necessary to write the variable immediately after the quantifier to indicate which variable the quantifier interacts with.

Quantifier: ∀, ∃	Predicates: Brother, Father, >
Connectives: ∧, ∨, ¬, ⇒, ⇔	Variables: x, y, z, a, b

Key aspects of knowledge representation languages are:

Syntax has to do with what ‘things’ (symbols, notations) one is allowed to use in the language and in what way; there is/are a(n):

- Alphabet
- Language constructs
- Sentences to assert knowledge

Semantics, Formal meaning, which has to do what those sentences with the alphabet and constructs are supposed to mean.

Predicate Logic: Syntax

The syntax tells, how programs must be written, where blanks are allowed, where semi|colons must be written, whether to use brackets or parentheses, etc.), a well|formed expression in a programming language is a program that passed the compiler without úsyntax errorø.

Predicate Logic: Semantics

Semantics refers to the meaning of expressions in a language. The semantics of an expression is based on its syntax, and also the actions that can be attached to an expression~such as proving, computing, and solving~are guided by the syntactical structure of expressions. Hence, a clear understanding of the syntax is the basis, and often the key, for understanding mathematics

Predicate Logic: Quantification

Quantifiers are phrases that refer to given quantities, such as "for some" or "for all" or "for every", indicating how many objects have a certain property.

Two kinds of quantifiers:

– Universal Quantifier: represented by , “for all”, “for every”, “for each”, or “for any”.

$\forall x\ \forall y$ LOVE(x,y) Everything loves everything

– Existential Quantifier: represented by , “for some”, “there exists”, “there is a”, or “for at least one”.

$\exists x\ \exists y$ LOVE(x,y) Something loves something

Inference with FOPL

[chapter 9 Pg 542]

Issues in Knowledge Representation

- Relationship Issue
- Granularity Issue
- Attribute Issue

NOTES

https://genuinenotes.com/wp-content/uploads/2021/05/unit4.pdf