Unit 3: Knowledge Representation

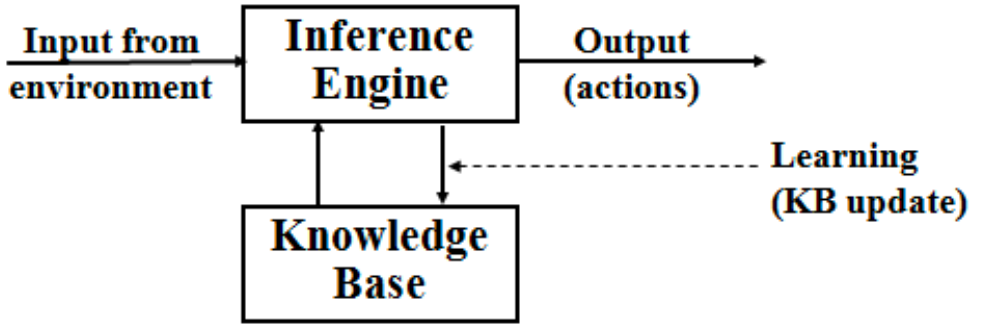
By: Sudhan Kandel

Knowledge

- Knowledge is a theoritical or practical understanding of a subject or a domain.
- Knowledge is also the sum of what is currently known.
- **Types of Knowledge**
  - **Classification-based Knowlwdge:** Ability to classify information
  - **Decision-Oriented Know:** Choosing the best option
  - **Descriptive knowlede :** State of some world world (heuristic)
  - **Procedural Knowledge:** How to do something
  - **Reasoing Knowledge:** What conclusion is valid in what situation?
  - **Assimilative knowledge:** What its impact is?

A Knowledge Based Agent

- A knowledge based agent consiste of a knowledge base (KB) and an inference engine(IE)
- A knowledge base is a set of sentences of what one knows about the world.
- The inference engine derives new sentences from the input and KB.
- The agent operates as follows:
  - It receives percepts from environment
  - It computes what action it should perform by IE and KB
  - It performs the chosen action.



Contd..

- Properties for knowledge representation systems:
  - The following properties should be possessed by a knowlwdge 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 Effficiecy**
    - The ability to direct the inferential mechanisms into the most pruductive directions by storing appropriate guides.
  - **Acquistitional Efficiency**
    - The ability to acquire new knowledge using automatic methods whenever possible rather than reliance on human intervention.

Knowledge Representation

- The objective of knowledge representation is to express the knowledge about the world in a computer-tractable from.
- This computer- tractable from of knowledge help the agent to identify patterns of good reasoning and patterns of bad reasoning, so the agent know which to follow and which to avoid.
- A forrmal language is required to represent knowledge in a computer tractable from and reasoning process are required to manipulate this knowledge to deduce new facts.
- Key aspects of knowledge representation language are:
  - **Syntax:** Describes how sentences are formed in the language.
  - **Semantics:** Describe the meaning of sentences, what is it the sentence refers to in the real world.
  - **Proof theory (inference Method):** Set of rules for generating new sentences that are necessarily true given that the old sentences are true.

Knowledge Representation Using Logic

- Logic is defined as a formal language for expressing knowledge and way of reasoning.
- Therefore, it should have syntax, semantic and inference method.
  - **Syntax:** Describe how sentences are formed in the LOGIC
  - **Semantics:** Describe the meanning of sentences.
  - **Inference Method:** Set of Rules for generating new sentences.

Contd..

- Compare to natural language (expressive but context sensitive) and programming language (good for concrete data structures but not expressive) logic combines the advantages of netural languages and formal languages.
- So Logic is:
  - Concise, Unambigious, Context insensitive, Expressive, Efficitive for inferences
- **Example of Logics are:**
  - **Propositional Logic**
  - **Predicate Logic**
  - **Fuzzy Logic**

Propositional Logic

- Propositional logic is the simplest fromal logic for the representation of the knowledge in terms of propositions.
  - Proposition is a declarative statement that is either true or false but not both.
  - If a proposition is true then we say it has a truth value of “TRUE” if a proposition is false, its truth value is “FALSE”.
- Some examples of propositions are given below:
    - “Man is mortal”, it returns truth value “TRUE”
    - “12+9=3-2” ot returns truth value “FALSE”
- The Following sentences are not proposition:
  - “A is less than 2”. It is because unless we give a specific value of A, we cannot say whether the statement is true or false.
  - Also the sentences “Chlose the door” is not propositions.

Syntax of Propositional Logic

- Proposational logic can be formed by combining atomic formulas with the following connectives

Name of Connective	Connective Word	Symbol
Negation	Not	] or ~ or ‘ or –
Conjunction	And	∧
Disjunction	Or	∨
Conditional	If-then	→
Biconditional	If and only if	↔

contd..

- Order of Precedence of logical connectors

Operator	Precedence
$\neg$	1
$\wedge$	2
$\vee$	3
$\rightarrow$	4
$\leftrightarrow$	5

e.g. $\neg P \vee Q \wedge R \Rightarrow S$ is equivalent to $((\neg P) \vee (Q \wedge R)) \Rightarrow S$

- A BNF (Backus_Naur Form) grammar of sentences in propositional logic is shown in below:

  Sentence → Atomic_Sentence | Complex_Sentence

  Atomic_Sentence → True | False | Symbol

  Symbol → P | Q | R …

  Complex_Sentence → ¬Sentence

  (Sentence ∧ Sentence)

  (Sentence ∨ Sentence)

  (Sentence → Sentence)

  (Sentence ↔ Sentence)

Contd..

- Examples of PL sentences:
  - P means "It is hot"
  - Q means "It is humid"
  - R means "It is raining"
  - P ^ Q → R
    - "If it is hot and humid, then it is raining”
  - Q → P
    - "If it is humid, then it is hot"

Formal logic – connectives:

For statement P = It is raining and Q = I am indoors.

- ✓ It is raining and I am indoors $(P \wedge Q)$
- ✓ If it is raining, then I am indoors $(P \vee Q)$
- ✓ It is raining if I am indoors $(Q \rightarrow P)$
- ✓ It is raining if and only if I am indoors $(P \leftrightarrow Q)$
- ✓ It is not raining $(\neg P)$

Semantics of Propositional Logic

- ✓ In propositional logic all sentences are **constructed from atomic sentences and the five connectives**
- ✓ So semantics of propositional logic requires rules to define the truth of atomic sentences and rules to define the truth of sentences formed with each of the five connectives

Rules for evaluating truth with respect to a model:

i. ¬ S is true if, S is false

ii. S1 ^ S2 is true if, S1 is true and S2 is true

iii. S1 V S2 is true if, S1 is true or S2 is true

iv. S1 → S2 is true if, S1 is false or S2 is true

v. S1 ↔ S2 is true if, S1 → S2 is true and S2 → S1 is true

Semantics of Propositional Logic

- Truth Table showing the evaluation of semantics of complex sentences:

P	Q	$\neg$P	P$\wedge$Q	P$\vee$Q	P$\Rightarrow$Q	P$\Leftrightarrow$Q
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

Propositional Logic

Properties:

- ✓ Validity(Tautology)
- ✓ Satisfiability (contingency)
- ✓ Un-Satisfiability (Contradictory)
- ✓ Equivalent
- ✓ Entailment
- ✓ Soundness
- ✓ Completeness

Contd..

- **Validity**

✓ A sentence is valid if it is true in all models,

- e.g., True, A v ¬ A, A → A

✓ Valid sentences are also known as tautologies.

✓ Every valid sentence is logically equivalent to True

- **Example** : Prove [(A→B)∧A]→B is a tautology

Solution: The truth table is as follows

A	B	A → B	(A → B) ∧ A	[(A → B) ∧ A] → B
True	True	True	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	True

As we can see every value of [(A→B)∧A]→B is "True", it is a tautology.

Contd..

- **Satisfiability**

✓ A sentence is satisfiable if it is true in some model

e.g., A v B

✓ Satisfiable sentences are also known as Contingency.

Example: Prove (A∨B)∧(¬A)a contingency

Solution: The truth table is as follows

A	B	A ∨ B	¬ A	(A ∨ B) ∧ (¬ A)
True	True	True	False	False
True	False	True	False	False
False	True	True	True	True
False	False	False	True	False

As we can see every value of (A∨B)∧(¬A) has both "True" and "False", it is a contingency.

Contd..

- **Un-Satisfiability**
  - ✓ A sentence is unsatisfiable if it is true in no models

    e.g., A ^ ¬ A
  - ✓ Un-Satisfiable sentences are also known as contradictory sentences.

Example: Prove $(A \vee B) \wedge [(\neg A) \wedge (\neg B)]$ is a contradiction

Solution: The truth table is as follows :

A	B	A ∨ B	¬ A	¬ B	(¬ A) ∧ (¬ B)	(A ∨ B) ∧ [(¬ A) ∧ (¬ B)]
True	True	True	False	False	False	False
True	False	True	False	True	False	False
False	True	True	True	False	False	False
False	False	False	True	True	True	False

As we can see every value of $(A \vee B) \wedge [(\neg A) \wedge (\neg B)]$ is "False", it is a contradiction.

Contd..

- **<u>Logical euivalence</u>**
  - ✓Two sentences P and Q are logically equivalent (P ≡ Q) iff true they are true in same set of models
  - ✓Two sentences P and Q are logically equivalent (P ≡ Q) iff P → Q and Q → P.

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Contd..

Example : Prove ¬(A∨B)and[(¬A)∧(¬B)] are equivalent

Testing by 1st method (Matching truth table)

A	B	A ∨ B	¬ (A ∨ B)	¬ A	¬ B	[(¬ A) ∧ (¬ B)]
True	True	True	False	False	False	False
True	False	True	False	False	True	False
False	True	True	False	True	False	False
False	False	False	True	True	True	True

Here, we can see the truth values of ¬(A∨B)and[(¬A)∧(¬B)] are same, hence the statements are equivalent.

Contd..

- **Entailment:**

✓ Entailment means that one thing follows from another:

- KB |= α

✓ Knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true

- E.g., x + y =4 entails 4=x + y

- **Example:**

Let, conclusion **α = A v B** and **KB = (A v C) ^ (B v ¬ C)**

Does KB entail α?

check all possible models; α must be true whenever KB is true

Contd..

- Entailment:

A	B	C	$A \vee C$	$(B \vee \neg C)$	KB	α
True	True	True	True	True	True	True
True	True	False	True	True	True	True
True	False	True	True	False	False	True
True	False	False	True	True	True	True
False	True	True	True	True	True	True
False	True	False	False	True	False	True
False	False	True	True	False	False	False
False	False	False	False	True	False	False

- Therefore, KB entails α

Contd..

- **Soundness:**
- **Soundness vs Validity**

✓ A logically valid argument is *one where the conclusion follows from the premises.*

- This is an example of a valid argument.
  - All fire-breathing rabbits live on Mars
  - All humans are fire-breathing rabbits
  - Therefore, all humans live on Mars

✓ The first premise is false, yet the conclusion is still valid.

✓ This argument is valid but not sound.

✓ In order for a deduction to be sound, the deduction must be valid and the premise must all be true.

Contd..

- **Soundness:**
- **Soundness vs Validity**

✓ Let’s take one of the above examples.

- 1. All monkeys are primates
- 2. All primates are mammals
- 3. All monkeys are mammals

✓ This is a sound argument because it is actually true in the real world.

Inference Method

- ✓ The process by which a conclusion is drawn from given premises.
- ✓ But logic is concerned with: does the truth of the conclusion follow from that of the premises?

- ✓ Several basic methods for determining whether a given set of premises propositionally entails a given conclusion.

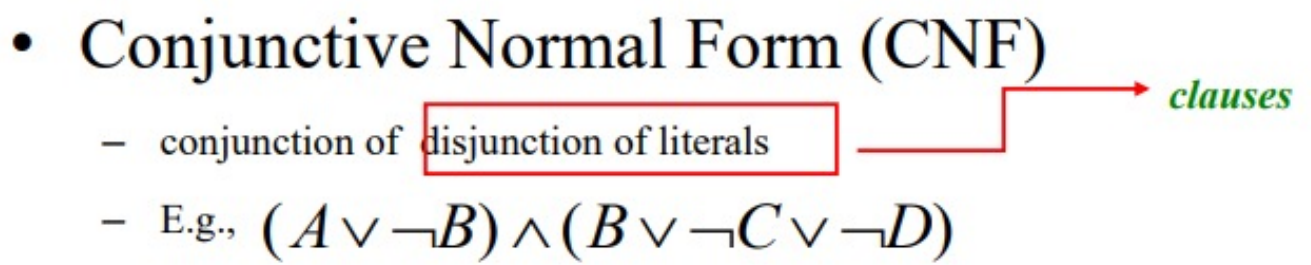
i. Truth Table Method(Enumeration Method)

ii. Deductive (Proof) Systems

iii. Resolution

Conjunctive Normal Form (CNF)

✓ A sentence that is expressed as a **conjunction of disjunctions of literals** is said to be in conjunctive normal form (CNF).



Remember:

✓ A *literal* is either an atomic sentence or a negation of an atomic sentence. **For e.g. p, ¬p**

✓ A *clause expression* is either a literal or a disjunction of literals. **For e.g. p, ¬p, p v q**

Conversion to CNF

Algorithm:

- ✓ Eliminate ↔ rewriting P↔Q as (P→Q)∧(Q→P)
- ✓ Eliminate → rewriting P→Q as ¬P∨Q
- ✓ Use De Morgan's laws to push ¬ inwards:
  - rewrite ¬(P∧Q) as ¬P∨¬Q
  - rewrite ¬(P∨Q) as ¬P∧¬Q
- ✓ Eliminate double negations:
  - rewrite ¬¬P as P
- ✓ Use the distributive laws to get CNF:
  - rewrite (P∧Q)∨R as (P∨R)∧(Q∨R)
- ✓ Flatten nested clauses:
  - (P∧Q) ∧ R as P∧Q ∧ R
  - (P∨Q)∨R as P∨Q∨R

Conversion to CNF by using an example. B ↔ (A ∨ C)

Conversion to CNF

Example:

✓ Let’s illustrate the conversion to CNF by using an example. $B \leftrightarrow (A \vee C)$

✓

$$B \Leftrightarrow (A \vee C)$$

- Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow ß$ with $(\alpha \Rightarrow ß) \wedge (ß \Rightarrow \alpha)$.
  - $(B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B)$
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow ß$ with $\neg \alpha \vee ß$.
  - $(\neg B \vee A \vee C) \wedge (\neg(A \vee C) \vee B)$
- Move $\neg$ inwards using de Morgan's rules and double-negation:
  - $(\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B)$
- Apply distributivity law ($\wedge$ over $\vee$) and flatten:
  - $(\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B)$

Inference using Resolution

Unit Resolution rule: This rule takes a clause (a disjunction of literals) and a literal and produces a new clause. Single literal is also called unit clause.

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

For Example: $\dfrac{\alpha \vee \beta, \quad \neg\beta}{\alpha}$

Generalized resolution rule: This rule takes two clauses of any length and produces a new clause as below.

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

where ℓ_i and m_j are complementary literals.

For Example:

$$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Proof by Resolution

Resolution algorithm

i. Convert KB into CNF

ii. Add negation of sentence to be entailed into KB i.e. (KB $\wedge \neg \alpha$)

iii. Then apply resolution rule to resulting clauses.

iv. The process continues until:

There are no new clauses that can be added

Hence KB does not entail α

Two clauses resolve to entail the empty clause.

Hence KB does entail α

Resolution Example1

Example: Consider the knowledge base given as: $KB = (B \Leftrightarrow (A \vee C)) \wedge \neg B$
Prove that $\neg A$ can be inferred from above KB by using resolution.

Solution:

At first, convert KB into CNF

$B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B) \wedge \neg B$

$(\neg B \vee A \vee C) \wedge (\neg(A \vee C) \vee B) \wedge \neg B$

$(\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B) \wedge \neg B$

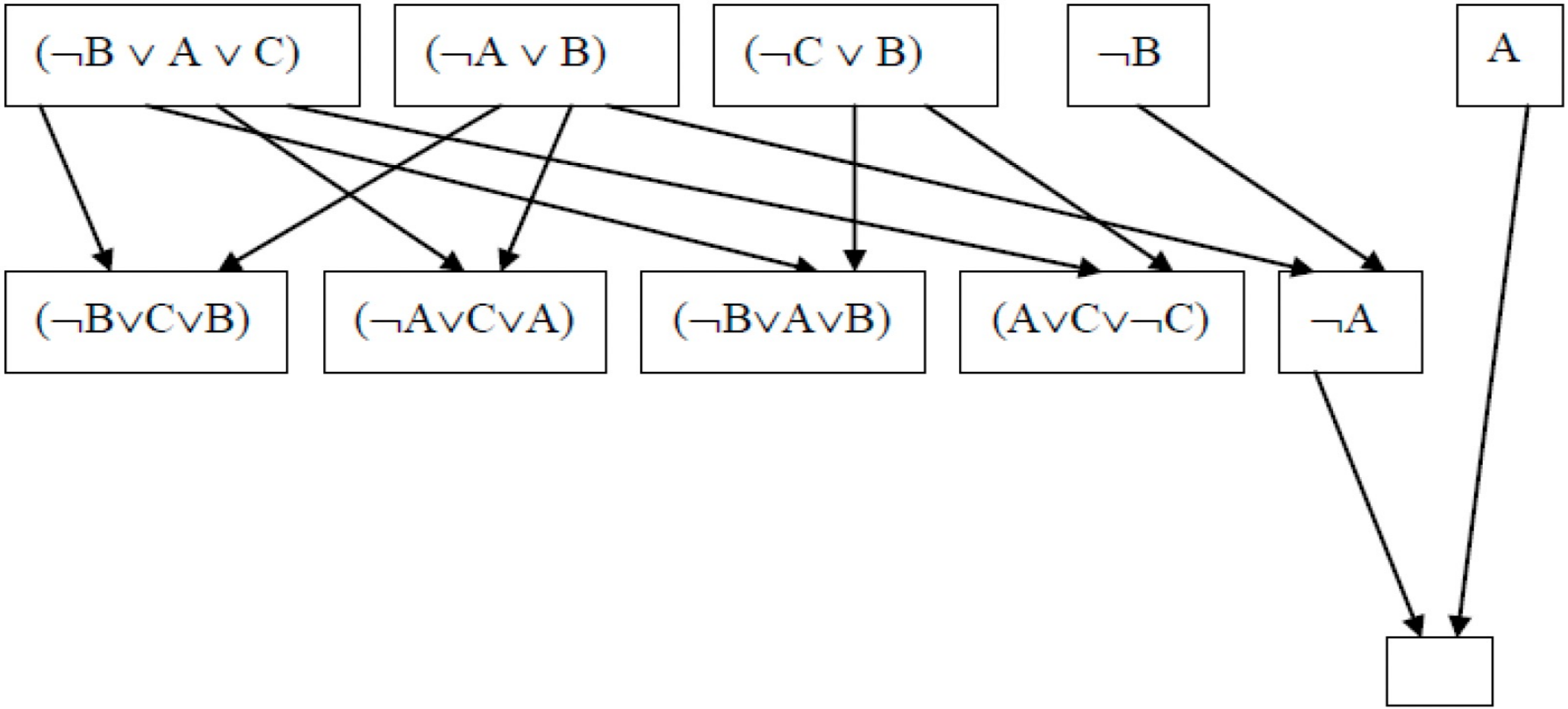
$(\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B) \wedge \neg B$

Add negation of sentence to be inferred from KB into KB

Now KB contains following sentences all in CNF
$(\neg B \vee A \vee C)$
$(\neg A \vee B)$
$(\neg C \vee B)$
$\neg B$
A (negation of conclusion to be proved)

Now use Resolution algorithm

Contd..



Example2

Given the following hypotheses:

1. If it rains, Joe brings his umbrella
2. If Joe has an umbrella, he doesn't get cold
3. If it doesn't rain, Joe doesn't get cold

Prove that Joes doesn't get cold

Represent using Propositional logic:

1. If it rains, Joe brings his umbrella (r -> u)
2. If Joe has an umbrella, he doesn't get cold (u -> ¬c)
3. If it doesn't rain, Joe doesn't get cold (¬r -> ¬c)

To prove: Joes doesn't get cold (¬c)

Contd..

We first put each hypothesis in CNF:

1. r -> u = (¬ r V u)
2. u -> ¬ c = (¬ u V ¬ c)
3. ¬ r -> ¬ c = (r V¬ c)

Applying resolution

1. ¬ r V u (Premise)
2. ¬ u V ¬ c (Premise)
3. r V ¬ c (Premise)
4. c (Negation of conclusion)
5. ¬ r V ¬ c (Using resolution on 1, 2)
6. ¬c V ¬c (Using resolution on 3, 5)
7. ¬ c (Using resolution on 4, 6)
8. (Using resolution on 4, 7)

Pros and cons of propositional logic

- Propositional logic is declarative
- Propositional logic is compositional:
  - meaning of $B \wedge P$ is derived from meaning of B and of P
- Meaning in propositional logic is context-independent
  - (unlike natural language, where meaning depends on context)
- Propositional logic has very limited expressive power (unlike natural language)

Home Work

- **Comparison Between propositional logic and FOPL**

Predicate Logic

- ✓ Predicate Logic is a more powerful logic (use foundation of propositional logic) by adding more expressive concepts.

Propositional logic is limited in several ways:

- ✓ Propositional logic is declarative
- ✓ Propositional logic is compositional:
    - o meaning of B ∧ P is derived from meaning of B and of P
- ✓ Propositional logic has very limited expressive power
    - o (unlike natural language)

Predicate Logic

- ✓ Propositional logic assumes the world contains facts, whereas first-order logic (like natural language) assumes the world contains:

i. Objects: which are things with individual identities — people, houses, companies…

ii. Properties: that distinguish objects from others — blue, oval, even, large, ...

iii. Relations: that hold among sets of objects — brother of, bigger than, part of, comes between,…

iv. Functions: which are a subset of relations — where there is only one —value‖ for any given—input‖
father of, best friend, one more than, plus, …

Representing knowledge in FOPL

- The basic syntactic elements of first order logics are the symbols.
- Formula in FOPL contains two types of Symbols. They are:

1. User defined symbols

- Constants:
  - 3, John
  - Individuals
- Functions:
  - f,g,h
  - mappings
- Predicates:
  - P(x,y)
  - functions whose range is {True,False}

2. Logic defined symbols

- Variables:

  x,y,z

  Can be instantiated

- Logical Operators()
- Truth Symbols (TRUE, FALSE)

Quantifiers:

- ∀ for all
- ∃ there exists

Quantifier Scope

- The part of a logical expression to which a quantifier is applied is called the scope of this quantifier.
- For example, suppose we want to say
  - "everyone who is alive loves someone"
  - $(\forall x)\ \text{alive}(x) \rightarrow (\exists y)\ \text{loves}(x,y)$
- Here's how we scope the variables

$$(\forall x)\ \text{alive}(x) \rightarrow (\exists y)\ \text{loves}(x, y)$$

Scope of x

Scope of y

Free vs. Bound Variables

- Definition
  - An occurrence of a variable in a formula is bound iff the occurrence is in the scope of a quantifier employing the variable; otherwise it is free.
- Examples
  - $\forall x.P(x, y)$
  - x is bound
  - y is free

Nesting and mixing quantifiers

- Switching the order of multiple universal quantifiers does not change the meaning;

$$\forall X, \forall Y\ P(X,Y) \Leftrightarrow \forall Y, \forall X, P(X,Y)$$

- For example, “Brothers are siblings” can be written as

$$\forall x \forall y \text{ Brother } (x, y) \Rightarrow \text{Sibling}(x, y) .$$

- Consecutive quantifiers of the same type can be written as one quantifier with several variables.

$$\forall x, y \text{ Brother}(x, y) \Leftrightarrow \text{Sibling}(y, x) .$$

Connections between All and Exists

- We can relate sentences involving $\forall$ and $\exists$ using **De Morgan's laws**:
  1. $(\forall x)\ \neg P(x) \leftrightarrow \neg(\exists x)\ P(x)$
  2. $\neg(\forall x)\ P(x) \leftrightarrow (\exists x)\ \neg P(x)$
  3. $(\forall x)\ P(x) \leftrightarrow \neg\ (\exists x)\ \neg P(x)$
  4. $(\exists x)\ P(x) \leftrightarrow \neg(\forall x)\ \neg P(x)$
- Examples
  1. All dogs don't like cats ↔ No dogs like cats
  2. Not all dogs dance ↔ There is a dog that doesn't dance
  3. All dogs sleep ↔ There is no dog that doesn't sleep
  4. There is a dog that talks ↔ Not all dogs can't talk

Translating English to FOL

Convert the following to the language of predicate logic.

- Every apple is either green or yellow

$$\forall X(apple(X) \Rightarrow green(X) \vee red(X))$$

- No apple is blue

$$\forall X(apple(x) \Rightarrow \neg blue(X))$$

- If an apple is green then its tasty

$$\forall X((apple(X) \wedge green(X)) \Rightarrow tasty(X))$$

- Every man likes a tasty apple

$$\forall X \exists Y(man(X) \wedge tastyApple(Y) \Rightarrow likes(X, Y))$$

- Some people like garlic

$$\exists X(person(X) \Rightarrow likes(X, garlic))$$

- **Conversion to CNF**
  - The procedure for conversion to CNF is similar to the propositional case.
  - We illustrate the procedure by translating the sentence "Everyone who loves all animals is loved by someone," or
    - $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{ Loves}(y, x)]$.

Step1: Eliminate Bi-Implications and implications

$\forall x [\neg\forall y \neg\text{Animal}(y) \vee \text{Loves}(x, y)] \vee [\exists y \text{ Loves}(y, x)]$.

Contd..

- Step2: Move ¬ inwards
  - In addition to the usual rules for negated connectives, we need rules for negated quantifiers. Thus, we have
    - ¬∀x p becomes ∃ x ¬p
    - ¬∃x p becomes ∀ x ¬p .
  - Our sentence becomes:
    - ∀ x [∃ y ¬(¬Animal(y) ∨ Loves(x, y))] ∨ [∃ y Loves(y, x)] .
    - ∀ x [∃ y ¬¬Animal(y) ∧ ¬Loves(x, y)] ∨ [∃ y Loves(y, x)] .

Contd..

- Step3: Eliminate double negation (¬¬)
  - ∀ x [∃ y Animal (y) ∧¬Loves(x, y)] ∨ [∃ y Loves(y, x)] .

Contd..

- Step4: Standardize variables (Rename)
  - Rename bound variables so that each only occurs once
  - For sentences like $(\exists x P(x)) \vee (\exists x Q(x))$ which use the same variable name twice, change the name of one of the variables.
    - Thus, we have
    - $\forall x [\exists y \text{ Animal } (y) \wedge \neg \text{Loves}(x, y)] \vee [\exists z \text{ Loves}(z, x)]$.

Contd..

- Step 5 : Move quantifiers to the left
  - $(\forall x P(x)) \vee Q$
    - Can be written as: $\forall x(P(x) \vee Q)$ and
  - $(\exists x P(x)) \vee (\exists y Q(y))$
    - Can be written as: $\exists x \exists y (P(x) \vee Q(y))$
  - Thus, we have
  - $\forall x \exists y \exists z[[Animal (y) \wedge \neg Loves(x, y)] \vee Loves(z, x)]]$.

Contd..

- Step 6: Skolemize to eliminate existential quantifiers
  - Skolemization is the process of removing existential quantifiers by elimination.
  - If ∃is not in the scope of ∀ then eliminate ∃ and replace existentially quantified variable by a constant not in the knowledge base.
  - E.g., a translate ∃x P(x) into P(A), where A is a new constant.
  - If ∃is in the scope of ∀ then eliminate ∃ and replace existentially quantified variable by a function with argument universally quantified variables in whose scope the existential quantifier appears.
    - ∀ x [Animal (F(x)) ∧¬Loves(x, F(x))] ∨ Loves(G(x), x) .
    - Here F and G are Skolem functions.

Contd..

- Step7: Drop universal quantifiers
  - At this point, all remaining variables must be universally quantified. Moreover, the sentence is equivalent to one in which all the universal quantifiers have been moved to the left. We can therefore drop the universal quantifiers:
  - [Animal (F(x)) ∧ ¬Loves(x, F(x))] ∨ Loves(G(x), x) .

Contd..

- Step8: Distribute ∨ over ∧
  - [Animal (F(x)) ∨ Loves(G(x), x)] ∧ [¬Loves(x, F(x)) ∨ Loves(G(x), x)] .
  - This step may also require flattening out nested conjunctions and disjunctions.
  - The sentence is now in CNF and consists of two clauses.
    - [Animal (F(x)) ∨ Loves(G(x), x)]
    - [¬Loves(x, F(x)) ∨ Loves(G(x), x)

Contd..

- **CNF Conversion summarized algorithm:**
  - Above descriptive steps of CNF conversion can be summarized in the following steps

1. Eliminate implications and bi-implications as in propositional case
2. Move negations inward using De Morgan's laws

   plus rewriting $\neg \forall x P$ as $\exists x \neg P$ and $\neg \exists x P$ as $\forall x \neg P$
3. Eliminate double negations
4. Rename bound variables if necessary so each only occurs once

   e.g. $\forall x P(x) \vee \exists x Q(x)$ becomes $\forall x P(x) \vee \exists y Q(y)$
5. Use equivalences to move quantifiers to the left

   e.g. $\forall x P(x) \wedge Q$ becomes $\forall x\ (P(x) \wedge Q)$ where x is not in Q

   e.g. $\forall x P(x) \wedge \exists y Q(y)$ becomes $\forall x \exists y (P(x) \wedge Q(y))$
6. Skolemise (replace each existentially quantified variable by a **new** term)

   $\exists x P(x)$ becomes $P(a0)$ using a Skolem constant $a0$ since $\exists x$ occurs at the outermost level

   $\forall x \exists y P(x, y)$ becomes $P(x, f0(x))$ using a Skolem function $f0$ since $\exists y$ occurs within $\forall x$
7. The formula now has only universal quantifiers and all are at the left of the formula: drop them
8. Use distribution laws to get CNF and then clausal form

The resolution inference rule:

- Two clauses, which are assumed to be standardized apart so that they share no variables, can be resolved if they contain complementary literals.
- Propositional literals are complementary if one is the negation of the other; first-order literals are complementary if one unifies with the negation of the other. Thus, we have

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\text{SUBST}(\theta, \ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}$$

where $\text{UNIFY}(\ell_i, \neg m_j) = \theta$

Contd..

- For example, we can resolve the two clauses

$$[Animal(F(x)) \vee Loves(G(x), x)] \quad \text{and} \quad [\neg Loves(u, v) \vee \neg Kills(u, v)]$$

by eliminating the complementary literals $Loves(G(x), x)$ and $\neg Loves(u, v)$, with unifier $\theta = \{u/G(x), v/x\}$, to produce the **resolvent** clause

$$[Animal(F(x)) \vee \neg Kills(G(x), x)] .$$

Resolution Algorithm

- Algorithm:
  - Convert KB into first order logic expressions.
  - Convert knowledge base (FOPL logic expressions) into CNF
  - convert the negation of query into CNF and then add them into KB.
  - Repeatedly apply resolution to clauses or copies of clauses(a copy of a clause is the clause with all variables renamed) until either the empty clause is derived or no more clauses can be derived.
    - If the empty clause is derived , answer = Yes (query follows form knowledge base).
    - Otherwise answer = No (query does not follow from knowledge base)

Example of resolution refutation

- Example: Consider the following statements:
  - Everyone who loves all animal is loved by some one.
  - Jack is loves all animal
  - Query: Jack is loved by someone.

Contd..

- KB in FOPL:
  - $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{ Loves}(y, x)]$.
  - $\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(\text{jack}, y)$

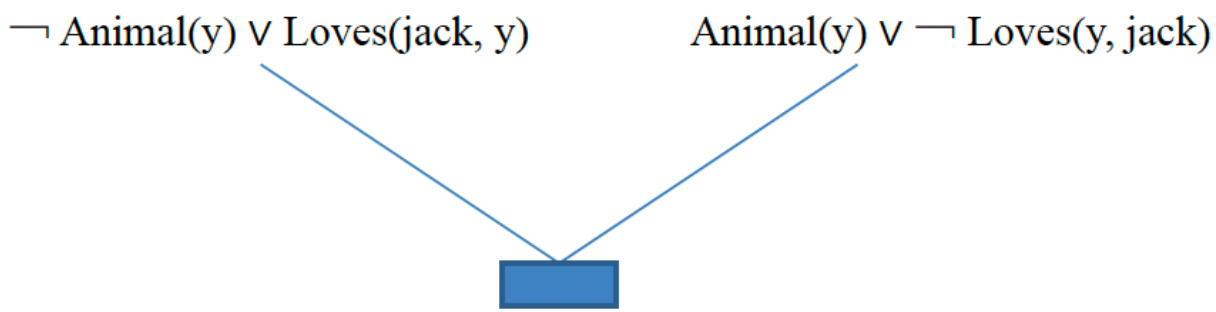
- Query in FOPL:
  - $\exists y \text{ Animal}(y) \Rightarrow \text{Loves}(y, \text{jack})$

- Negatón of query:
  - $\forall y \neg \text{Animal}(y) \Rightarrow \neg \text{Loves}(y, \text{jack})$

Contd..

- KB and Query in CNF:
    - [Animal (F(x)) ∨ Loves(G(x), x)]
    - [¬Loves(x, F(x)) ∨ Loves(G(x), x)
    - ¬ Animal(y) ∨ Loves(jack, y)
    - Animal(y) ∨ ¬ Loves(y, jack)

Contd..



Hence jack is loved by someone

Example of resolution refutation

KB:

1. Anyone passing his history exams and winning the lottery is happy.

2. Anyone who studies or is lucky can pass all his exams.

3. John did not study but John is lucky.

4. Anyone who is lucky wins the lottery.

query: “Is John happy?”.
Use the resolution refutation algorithm to answer the given query.

Contd..

(a) Translate the following four English sentences to first order logic (FOL).

1. Anyone passing his history exams and winning the lottery is happy.

2. Anyone who studies or is lucky can pass all his exams.

3. John did not study but John is lucky.

4. Anyone who is lucky wins the lottery.

(b) Convert them to conjunctive normal form (CNF).

(c) Answer the query "Is John happy?". Use the resolution refutation algorithm.

Contd..

1. Anyone passing his history exams and winning the lottery is happy.

$$\forall x\ Pass(x, HistoryExam) \wedge Win(x, Lottery) \Rightarrow Happy(x)$$

2. Anyone who studies or is lucky can pass all his exams.

$$\forall x \forall y\ Study(x) \vee Lucky(x) \Rightarrow Pass(x, y)$$

3. John did not study but John is lucky.

$$\neg Study(John) \wedge Lucky(John)$$

4. Anyone who is lucky wins the lottery.

$$\forall x\ Lucky(x) \Rightarrow Win(x, Lottery)$$

(b) Convert them to conjunctive normal form (CNF).

1 $\forall x\ Pass(x, HistoryExam) \wedge Win(x, Lottery) \Rightarrow Happy(x)$

2 $\forall x \forall y\ Study(x) \vee Lucky(x) \Rightarrow Pass(x, y)$

3 $\neg Study(John) \wedge Lucky(John)$

4 $\forall x\ Lucky(x) \Rightarrow Win(x, Lottery)$

First: Implication elimination

Contd..

1 $\neg[Pass(x, HistoryExam) \wedge Win(x, Lottery)] \vee Happy(x)$

2 $\neg[Study(x) \vee Lucky(x)] \vee Pass(x, y)$

3 $\neg Study(John) \wedge Lucky(John)$

4 $\neg Lucky(x) \vee Win(x, Lottery)$

Then: drop the ”for all” quantifiers

1 $[\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery)] \vee Happy(x)$

2 $[\neg Study(x) \wedge \neg Lucky(x)] \vee Pass(x, y)$

3 $\neg Study(John) \wedge Lucky(John)$

4 $\neg Lucky(x) \vee Win(x, Lottery)$

1 $[\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery)] \vee Happy(x)$

2 $[\neg Study(x) \wedge \neg Lucky(x)] \vee Pass(x, y)$

3 $\neg Study(John) \wedge Lucky(John)$

4 $\neg Lucky(x) \vee Win(x, Lottery)$

1 $\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$

2 $[\neg Study(x) \vee Pass(x, y)] \wedge [\neg Lucky(x) \vee Pass(x, y)]$

3 $\neg Study(John) \wedge Lucky(John)$

4 $\neg Lucky(x) \vee Win(x, Lottery)$

1 $\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$

2a $\neg Study(x) \vee Pass(x, y)$

2b $\neg Lucky(x) \vee Pass(x, y)$

3a $\neg Study(John)$

3b $Lucky(John)$

4 $\neg Lucky(x) \vee Win(x, Lottery)$

(c) Query and resolution refutation

1	$\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$
2a	$\neg Study(x) \vee Pass(x, y)$
2b	$\neg Lucky(x) \vee Pass(x, y)$
3a	$\neg Study(John)$
3b	$Lucky(John)$
4	$\neg Lucky(x) \vee Win(x, Lottery)$

Knowledge Base (KB)

Contd..

1	$\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$
2a	$\neg Study(x) \vee Pass(x, y)$
2b	$\neg Lucky(x) \vee Pass(x, y)$
3a	$\neg Study(John)$
3b	$Lucky(John)$
4	$\neg Lucky(x) \vee Win(x, Lottery)$
Q	$\neg Happy(John)$

{x/John}

Contd..

1 $\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$

2a $\neg Study(x) \vee Pass(x, y)$

2b $\neg Lucky(x) \vee Pass(x, y)$

{x/John}

3a $\neg Study(John)$

3b $Lucky(John)$

4 $\neg Lucky(x) \vee Win(x, Lottery)$

$\neg Pass(John, HistoryExam) \vee \neg Win(John, Lottery)$

Q $\neg Happy(John)$

Contd..

1 $\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$

2a $\neg Study(x) \vee Pass(x, y)$

2b $\neg Lucky(x) \vee Pass(x, y)$

3a $\neg Study(John)$

{x/John; y/HistoryExam}

3b $Lucky(John)$

4 $\neg Lucky(x) \vee Win(x, Lottery)$

$\neg Pass(John, HistoryExam) \vee \neg Win(John, Lottery)$ 5

Q $\neg Happy(John)$

Contd..

1	$\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$	
2a	$\neg Study(x) \vee Pass(x, y)$	
2b	$\neg Lucky(x) \vee Pass(x, y)$	
3a	$\neg Study(John)$	
3b	$Lucky(John)$	
4	$\neg Lucky(x) \vee Win(x, Lottery)$	
	$\neg Pass(John, HistoryExam) \vee \neg Win(John, Lottery)$	5
	$\neg Lucky(John) \vee \neg Win(John, Lottery)$	6

{x/John; y/HistoryExam}

Contd..

1 $\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$

2a $\neg Study(x) \vee Pass(x, y)$

2b $\neg Lucky(x) \vee Pass(x, y)$

3a $\neg Study(John)$

3b $Lucky(John)$

4 $\neg Lucky(x) \vee Win(x, Lottery)$ {x/John}

$\neg Pass(John, HistoryExam) \vee \neg Win(John, Lottery)$ 5

$\neg Lucky(John) \vee \neg Win(John, Lottery)$ 6

Contd..

1 $\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$

2a $\neg Study(x) \vee Pass(x, y)$

$\neg Lucky(John)$ 7

3a $\neg Study(John)$

3b $Lucky(John)$

4 $\neg Lucky(x) \vee Win(x, Lottery)$ {x/John}

$\neg Pass(John, HistoryExam) \vee \neg Win(John, Lottery)$ 5

$\neg Lucky(John) \vee \neg Win(John, Lottery)$ 6

Contd..

1 $\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$

2a $\neg Study(x) \vee Pass(x, y)$

$\neg Lucky(John)$ 7

3a $\neg Study(John)$

3b $Lucky(John)$

4 $\neg Lucky(x) \vee Win(x, Lottery)$

$\neg Pass(John, HistoryExam) \vee \neg Win(John, Lottery)$ 5

$\neg Lucky(John) \vee \neg Win(John, Lottery)$ 6

Contd..

1 $\neg Pass(x, HistoryExam) \vee \neg Win(x, Lottery) \vee Happy(x)$

2a $\neg Study(x) \vee Pass(x, y)$

$\neg Lucky(John)$ 7

3a $\neg Study(John)$

3b $Lucky(John)$

$\varnothing$

4 $\neg Lucky(x) \vee Win(x, Lottery)$

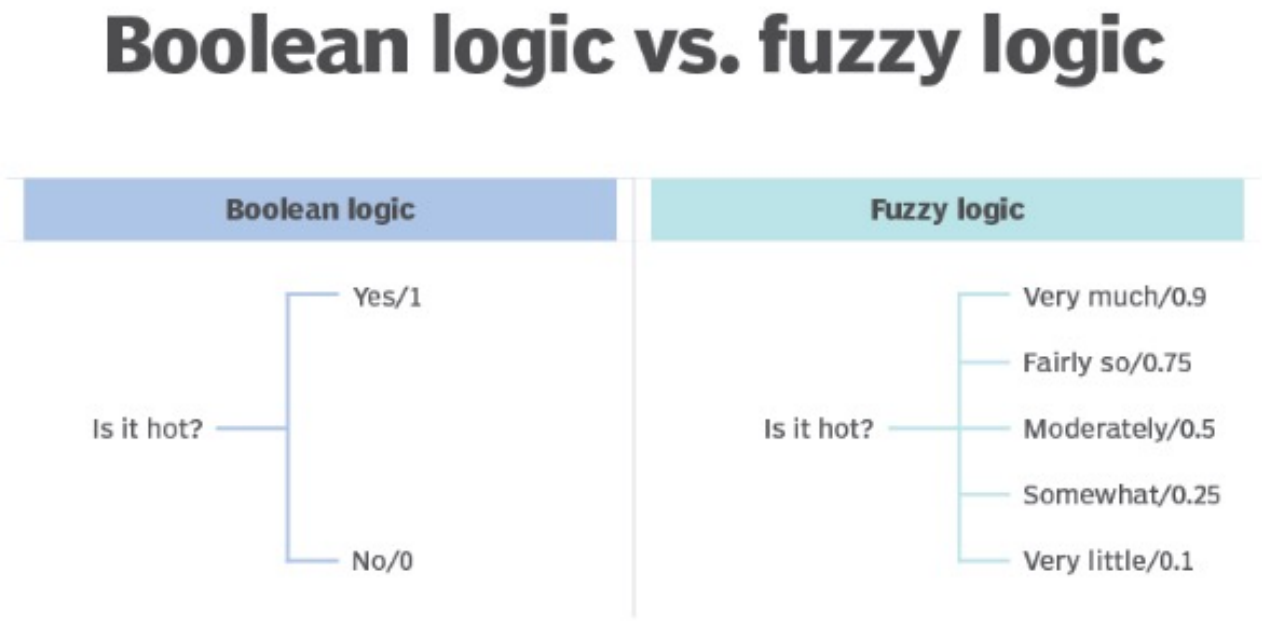
$\neg Pass(John, HistoryExam) \vee \neg Win(John, Lottery)$ 5

$\neg Lucky(John) \vee \neg Win(John, Lottery)$ 6

Fuzzy Logic

Fuzzy Logic

- ✓ The term **fuzzy** refers to things that are not clear or are vague.
- ✓ In the real world many times we encounter a situation when we can't determine whether the state is true or false, their fuzzy logic provides very valuable flexibility for reasoning.
- ✓ Compared to traditional binary sets (where variables may take on true or false values) fuzzy logic variables may have a truth value that ranges in degree between 0 and 1.



Fuzzy Logic

- ✓ It deals with reasoning that is approximate rather than fixed and exact.
- ✓ Fuzzy logic is based on the observation that people make decisions based on non-numerical information.

Example: If temperature is high , set speed to high.

Crisp – Clear, either 0 or 1

Fuzzy – not clear

Linguistic Variables:

- ✓ By a linguistic variable we mean a variable whose values are words or sentences in a natural or artificial language.
- ✓ **A numerical variable takes numerical values: Age = 65**. While, a linguistic variables takes linguistic values: Age is old

T(age) = {young, not young, very young, ...middle aged, not middle aged, ...old, not old, very old, more or less old, ...not very young and not very old, ...}

Fuzzy Logic

- ✓ **Fuzzy Rule-based** are rule-based systems, where fuzzy sets and fuzzy logic are used as tools for representing different forms of knowledge about the problem at hand.

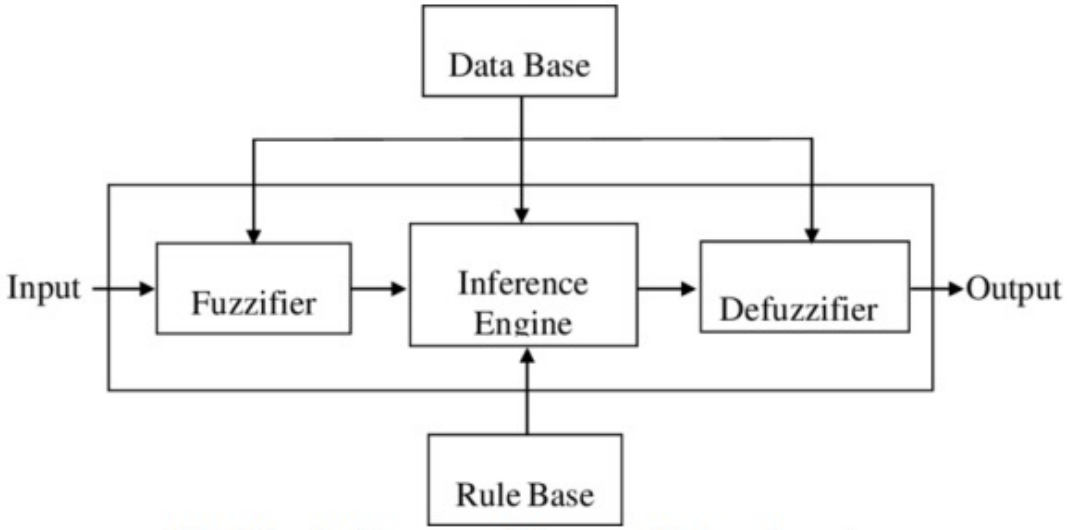
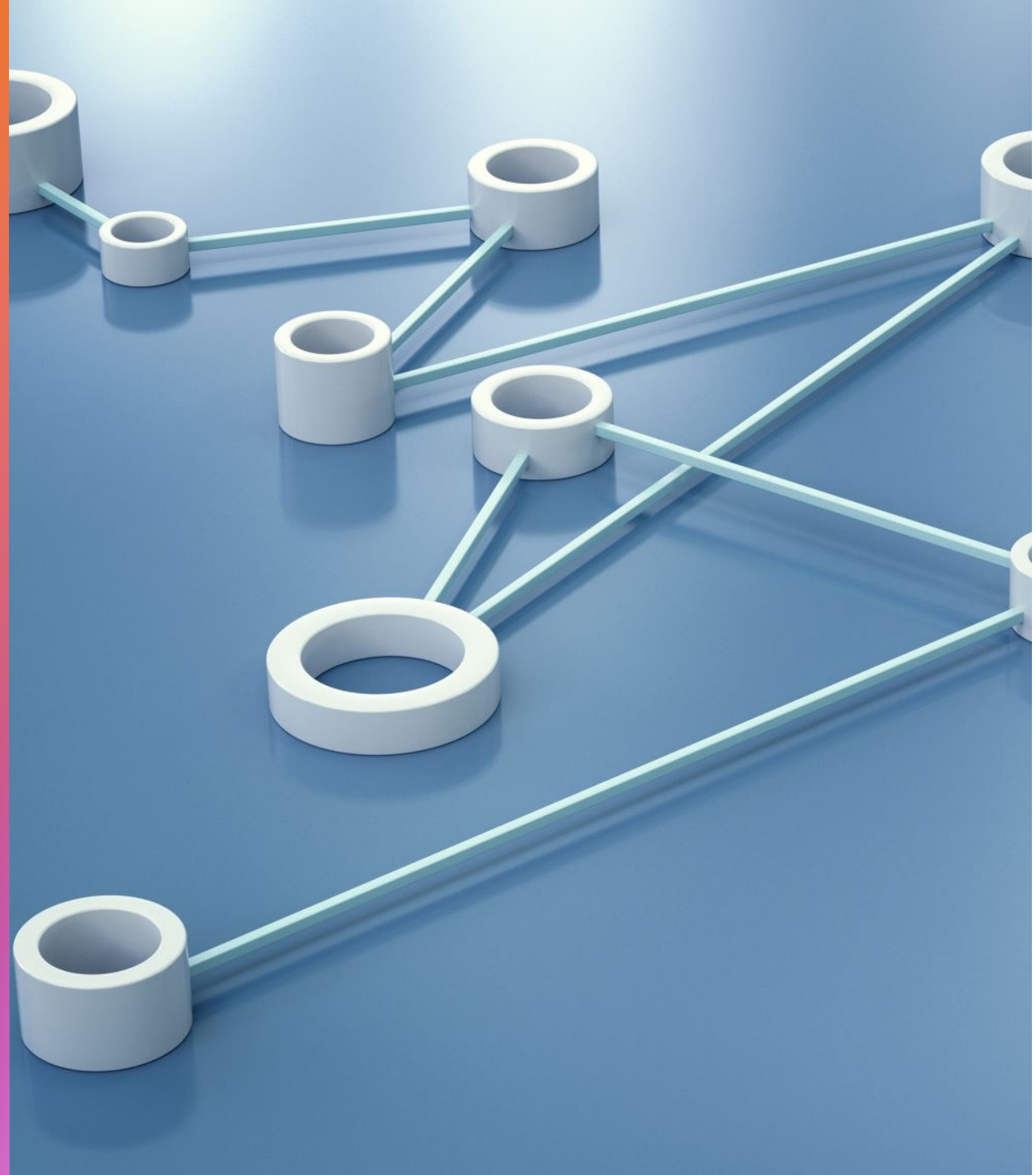


Fig: Block diagram of fuzzy rule based system

- ✓ The **fuzzifier** converts real numbers of inputs into fuzzy sets.
- ✓ The **knowledge base** includes a fuzzy rule-base and a database. ***Membership functions*** of the linguistic terms are contained in the database. The rule base consists of if then rules, which represents the relationship between input and output variables.
- ✓ The **defuzzifier** converts fuzzy value back to real world crisp values.



Semantic Network

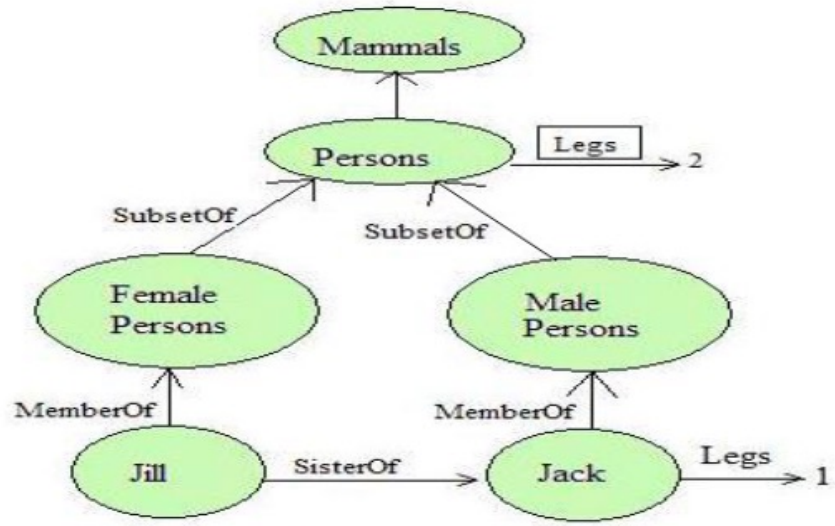
Semantic Network

- A semantic Net or semantic network is a knowledge representation technique used for propositional information. So it is also called a propositional net.
- Mathematically a semantic net can be defined as a labeled directed graph.
- Consist of:
  - Nodes
  - Links(edges)
  - Link labels

Semantic Network

- Nodes
  - In the semantic network diagram nodes appera as circles or ellipses or rectangles to represent objects such as physical objects, concepts or situations
- Links:
  - Appear as arrows to express the relationships between objects
- Link Labels:
  - Specify particular relations, relationship provide the basic structure for organizing knowledge.
  - The object and relations involved need not be so concrete.
- As nodes are associated with other nodes semantic nets are also reffered to as associative nets.

Semantic Network



- In the above figure all the objects are within ovals and connected using labelled arcs.
- Note that there is a link between Jill and FemalePersons with label MemberOf. Simlarly there is a MemberOf link between Jack and MalePersons and SisterOf link between Jill and Jack.
- The MemberOf link between Jill and FemalePersons indicates that Jill belongs to the category of female persons.

Semantic Network

- **Inheritance Reasoning**
  - Unless there is a specific evidence to the contrary, it is assumed that all members of a class (category) will inherit all the properties of their super classes.
  - So semantic network allows us to perform inheritance reasoning.
    - For example: Jill inherits the property of having two legs as she belongs to the category of FemalePersons which in turn belongs to the category of Persons which has a boxed Legs link with value 2.
  - Semantic nets allows multiple inheritance. So an object can belong to more than one category and a category can be a subset of more than one another category.

Semantic Network

- Inverse Links
  - Semantic network allows a common form of inference known as inverse links.
    - For example: we can have a *HasSister* link which is the inverse of *SisterOf* link.
    - The inverse links make the job of inference algorithms much easier to answer queries such as who the sister of *Jack* is.
    - On discovering that *HasSister* is the inverse of *SisterOf* the inference algorithm can follow that link HasSister from *Jack* to *Jill* and answer the query.

Semantic Network

- Disadvantage Of Semantic Nets
  - One of the drawbacks of semantic network is that the links between the objects represent only binary relations.
    - For example, the sentence Run(Kirtipur Express, Kirtipur, Ratnapark, Today) cannot be asserted directly.
  - There is no standard definition of link names.

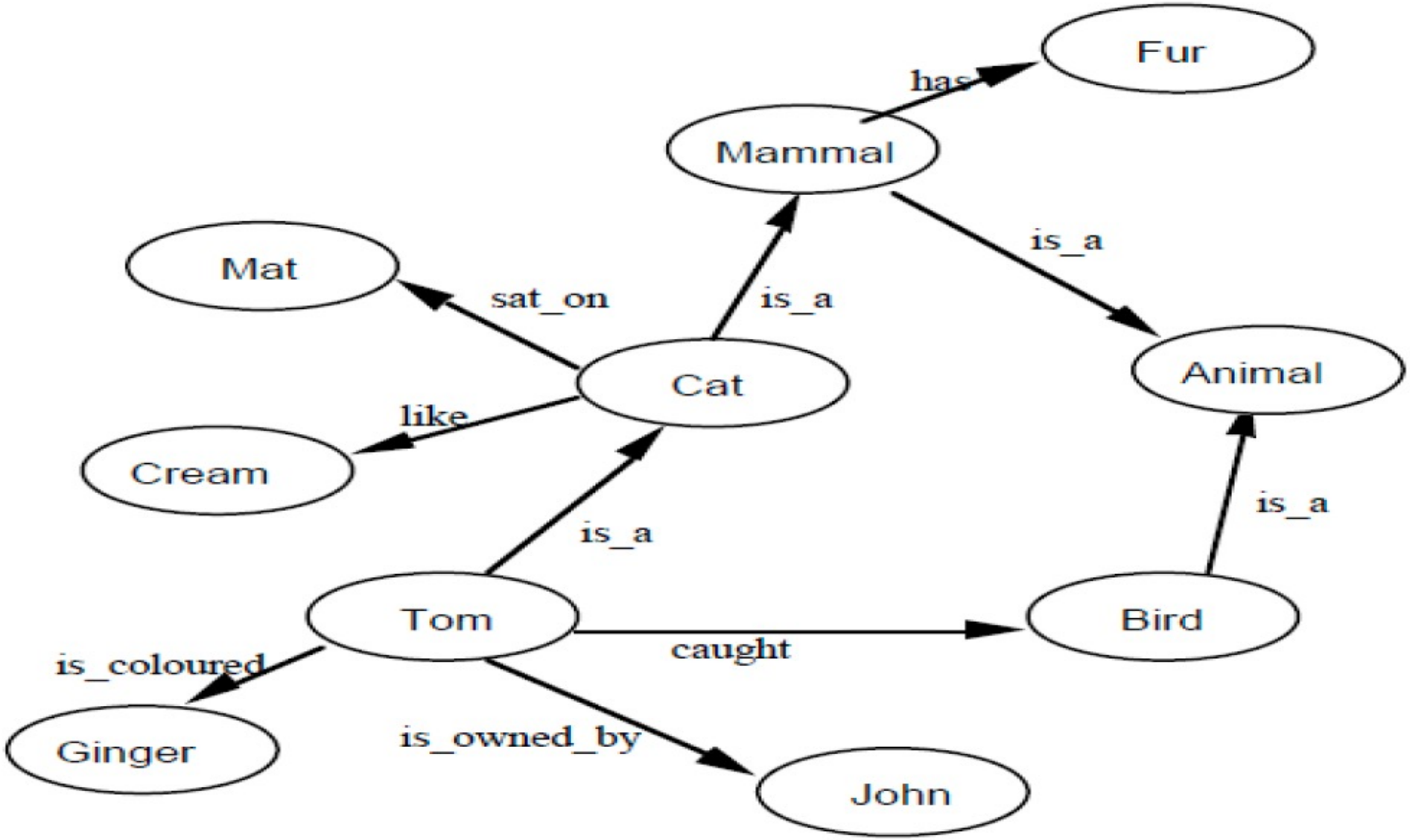
Semantic Network

- Advantages Of Semantic Nets
  - Semantic nets have the ability to represent default values for categories.
    - In the above figure Jack has one leg while he is a person and all persons have two legs. So persons have two legs has only default status which can be overridden by a specific value.
  - They convey some meaning in a transparent manner.
  - Semantic nets are simple and easy to understand.

Example1: semantic network

- Represent the following fact in semantic network
  - Tom is a cat.
  - Tom caught a bird.
  - Tom is owned by John.
  - Tom is ginger in color.
  - Cats like cream.
  - The cat sat on the mat.
  - A cat is a mammal.
  - A bird is an animal.
  - All mammals are animals.
  - Mammals have fur.

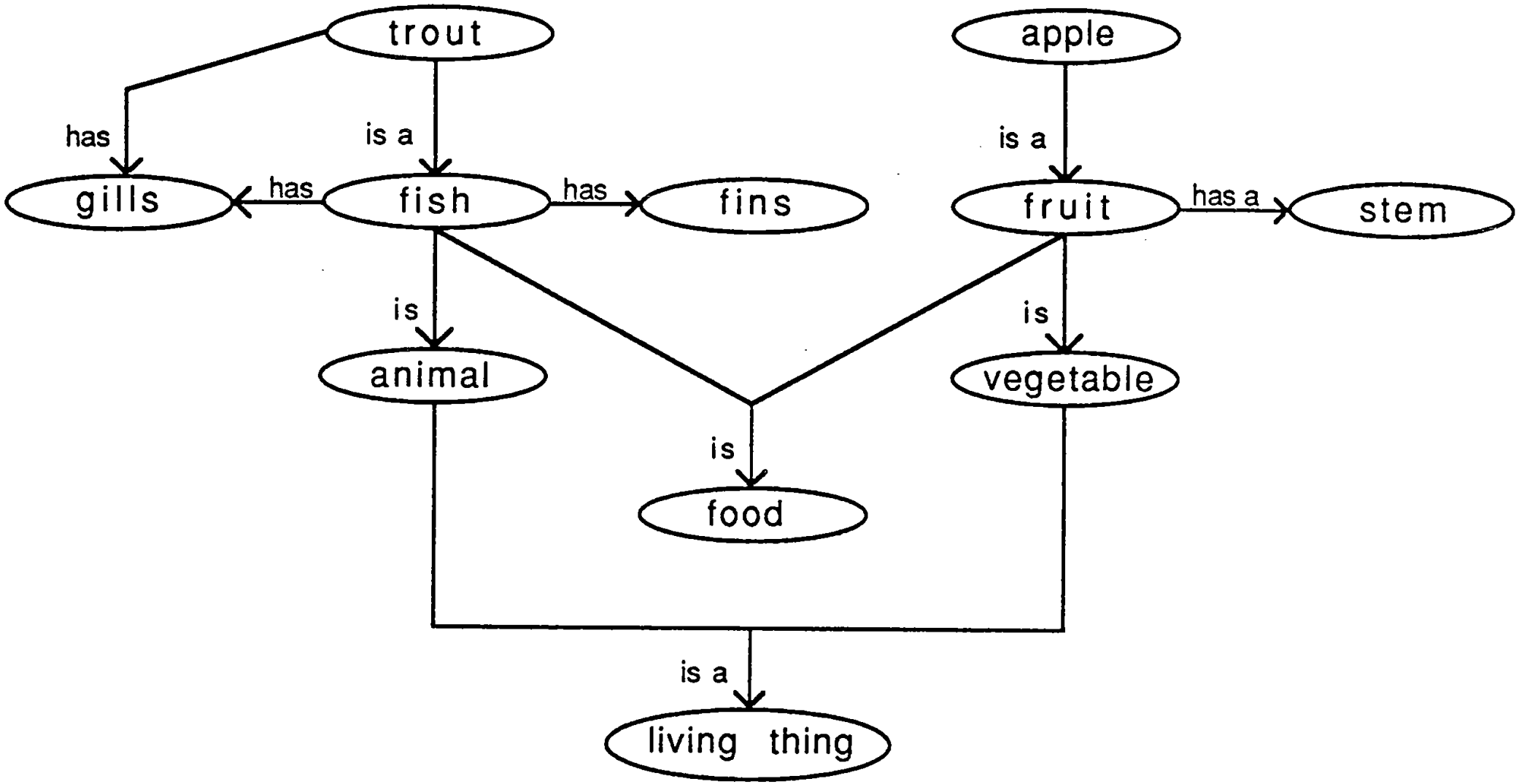
Contd..


Fur
has
Mammal
is_a
Mat
sat_on
is_a
Animal
Cat
like
Cream
is_a
is_a
Tom
Bird
caught
is_coloured
Ginger
is_owned_by
John

Example2: semantic network

- Represent the following fact in semantic network

(1) A trout is a fish.
(2) A fish has gills.
(3) A fish has fins.
(4) Fish is food.
(5) Fish is animal.
(6) An apple is a fruit.
(7) Fruit has a stem.
(8) Fruit is food.
(9) Fruit is vegetable.
(10) An animal is a living thing.
(11) A vegetable is a living thing.

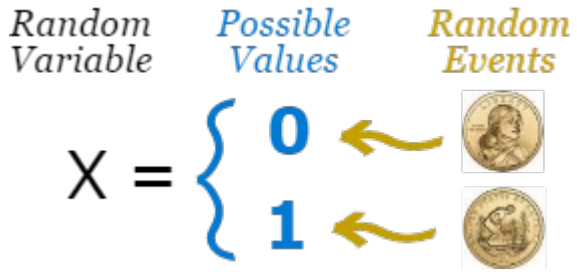

trout
apple
has
is a
is a
gills
has
fish
has
fins
fruit
has a
stem
is
is
animal
vegetable
is
food
is a
living thing

Random Variables

- ✓ In probability theory and statistics, a random variable (or stochastic variable) is a way of assigning a value (often a real number) to each possible outcome of a random event
- ✓ Intuitively, a random variable can be thought of as a quantity whose value is not fixed, but which can take on different values
- ✓ For example:

There are two possible outcomes for a coin toss: heads, or tails.

The possible outcomes for one fair coin toss can be described using the following random variable:



- ✓ and if the coin is equally likely to land on either side then it has a probability mass function given by:

$$\rho_X(x) = \begin{cases} \frac{1}{2}, & \text{if } x = \text{head}, \\ \frac{1}{2}, & \text{if } x = \text{tail}. \end{cases}$$

Random Variables

- ✓ Random Variable domain are: Boolean, Discrete and Continuous

Boolean

Cavity = false, toothache = true

Discrete random variables

e.g., Weather is one of (sunny, rain, cloudy, snow) Weather = rain is a proposition

Continuous random variables

e.g., Temp = 21.6, also allow, e.g., Temp < 22.0.

Random Variables

- ✓ The **prior or unconditional probability** associated with a proposition is the degree of belief accorded to it in the absence of any other information.

- ✓ Example:

  P(Weather= sunny) = 0.72

  P(Weather= rain) = 0.1

  P(Weather= cloudy) = 0.08

  P(Weather= snow) = 0.1

- ✓ **Probability distribution** gives values for all possible assignments: P(Weather) = (0.72, 0.1, 0.08, 0.1)

Random Variables

- ✓ **Joint distribution** is based on joint probability, which can be simply defined as the probability of two events (variables) happening together.
- ✓ These two events are usually coined event A and event B, and can formally be written as: **p**(A and B)
- ✓ P(Weather, Season) = a 4 ×4 matrix of values.

Season \ Weather	Sunny	Rain	Cloud	Snow
Spring	0.07	0.03	0.10	0.06
Summer	0.13	0.01	0.05	0.01
Autumn	0.05	0.05	0.15	0.03
Winter	0.05	0.01	0.10	0.10

Inference using full joint distribution

- ✓ The probability of any proposition event can be computed from the **full joint distribution.**
- ✓ The probabilities in the joint distribution must sum to 1.

We consider the following domain consisting of three Boolean variables: Toothache, Cavity, and Catch (the dentist's nasty steel probe catches in my tooth).

- ✓ The full joint distribution is the following 2x2x2 table:

	toothache		¬toothache	
	Catch	¬catch	catch	¬catch
Cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

- ✓ Simply identify those possible worlds in which the proposition is true and add up their probabilities.
- ✓ For example, there are six possible worlds in which cavity V toothache holds:

P(cavity V toothache) = 0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28

Inference using full joint distribution

Calculating Conditional Probability

$$P(a|b) = \frac{P(a \wedge b)}{P(b)}$$

P(¬cavity | Toothache) = P(¬cavity ^ Toothache)/ P(Toothache)

= (0.016 + 0.064)/(0.108 + 0.012 + 0.016 + 0.064) = 0.4

Again let's calculate

P(cavity | Toothache) = P(cavity ^ Toothache)/ P(Toothache)

= (0.108 + 0.012)/(0.108 + 0.012 + 0.016 + 0.064) = 0.6

P(Toothache) = 0.108 + 0.012 + 0.016 + 0.064

This process is called marginalization.

Bayes' Rule

Case of conditional dependence

$$P(a \wedge b) = P(a \mid b)P(b) \quad \text{and} \quad P(a \wedge b) = P(b \mid a)P(a)$$

Equating the two right-hand sides and dividing by P(a), we get

$$P(b \mid a) = \frac{P(a \mid b)P(b)}{P(a)}$$

Applying Bayes' rule:

$$P(\text{cause} \mid \text{effect}) = \frac{P(\text{effect} \mid \text{cause})P(\text{cause})}{P(\text{effect})}$$

Bayes' rule

- ✓ A doctor knows that the disease meningitis causes the patient to have a stiff neck 70% of the time. The doctor also knows that the probability that a patient has meningitis is 1/50,000, and the probability that any patient has a stiff neck is 1%.

Find the probability that a patient with a stiff neck has meningitis.

- ✓ Here, we are given;

$$p(s|m) = 0.7$$

$$p(m) = 1/50000$$

$$p(s) = 0.01$$

Now using Bayes' rule;

$$P(m|s) = P(s|m)P(m)/P(s) = (0.7*1/50000)/(0.01) = 0.0014$$

Thank You