Logic Concepts and logic Programming

Symbolic logic is divided into two.

1. Propositional logic
2. Predicate logic.

Propositional Calculus

→ A proposition refers to a declarative statement i.e., True or false.

→ A proposition Calculus refers to a language of propositions in which a set of rules are used to combine simple propositions to form compound propositions with help of certain logical operators.

→ logical operators are often called connectives:
i.e., not (~), and (∧), or (∨), implies (→) equivalence (↔).

→ A well-formed formula is defined as a symbol or a string of symbols generated by formal grammar of a formal language.

Some important properties of a well-formed formula.

- Smallest unit is considered to be a well-formed formula
- If α is a well-formed formula, then ~α is also a well-formed formula
- If α & β are well-formed formulae, then (α ∧ β), (α ∨ β), (α → β), (α ↔ β) are also well-formed formulae.

Truth Table

A	B	~A	A∧B	A∨B	A→B	A↔B
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

→ Compute the truth value of α: $(A \vee B) \wedge (\sim B \rightarrow A)$ using truth table approach.

A	B	A∨B	~B	~B→A	(A∨B)∧(~B→A)
T	T	T	F	T	T
T	F	T	T	T	T
F	T	T	F	T	T
F	F	F	T	F	F

Equivalence laws

Commutative law

$A \vee B \cong B \vee A$.

$A \wedge B \cong B \wedge A$

Associative law

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

$A \wedge (B \wedge C) \cong (A \wedge B) \wedge C$

Double Negation

$\sim(\sim A) \equiv A$.

Distributive laws

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

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

De Morgan's law

$\sim(A \vee B) \cong \sim A \wedge \sim B$

$\sim(A \wedge B) \cong \sim A \vee \sim B$.

Absorption laws	$A \vee (A \wedge B) \cong A$.
	$A \wedge (A \vee B) \cong A$
	$A \vee (\sim A \wedge B) \cong A \vee B$
	$A \wedge (\sim A \vee B) \cong A \wedge B$.
Idempotence	$A \vee A \cong A$
	$A \wedge A \cong A$.
Excluded Middle law	$A \vee \sim A \cong T$ (True)
Contradiction law	$A \wedge \sim A \cong F$
Commonly used equivalence relations	$A \vee F \cong A$
	$A \vee T \cong T$
	$A \wedge T \cong A$
	$A \wedge F \cong F$
	$A \rightarrow B \cong \sim A \vee B$
	$A \leftrightarrow B \cong (A \rightarrow B) \wedge (B \rightarrow A)$.
	$\cong (A \wedge B) \vee (\sim A \wedge \sim B)$.

Propositional logic

deals with validity, Satisfiability & unsatisfiability of a formula and derivation of a new formula using equivalence laws.

→ A formula is said to be valid if and only if it is a tautology

→ A formula is said to be satisfiable if there exists atleast one interpretation for which α is true

→ A formula is said to be unsatisfiable if value of α is false under all interpretations

Show that following is a valid argument.

If it is humid then it will rain and Since it is humid today it will rain.

Sol:- A : It is humid

B : It will rain

$\alpha : [(A \to B) \wedge A] \to B$.

A	B	$A \to B = (X)$	$X \wedge A = (Y)$	$Y \to B$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

→ Use of truthtable in some situations proves to be a waste of time. So, some other methods are

- Natural deduction System
- Axiomatic System
- Semantic tableau & method
- Resolution refutation method.

Natural Deduction System

→ it mimics the pattern of natural Reasoning.

→ It is based on a set of deductive inference rules.

Assuming that $A_1, \ldots A_k$ where $1 \le k \le n$ are a set of atoms and α_j. where $1 \le j \le m$ & β are well-formed formulae.

Inference rules — NDS rules Table

Rule Name	Symbol	Rule	Description
Introducing $\wedge$	$(I:\wedge)$	If $A_1 \ldots A_n$ then $A_1 \wedge \ldots \wedge A_n$.	If $A_1 \ldots A_n$ are true, then their conjunction $A_1 \wedge \ldots \wedge A_n$ is also true.
Eliminating $\wedge$	$(E:\wedge)$.	If $A_1 \wedge \ldots \wedge A_n$ then $A_i (1 \le i \le n)$	If $A_1 \wedge \ldots \wedge A_n$ is true, then any A_i is also true.
Introducing V	$(I:V)$	If any $A_i (1 \le i \le n)$ then $A_i \vee \ldots \vee A_n$	If any A_i $(1 \le i \le n)$ is true then $A_1 \vee \ldots \vee A_n$ is also true.
Eliminating V	$(E:V)$	If $A_1 \vee \ldots \vee A_n$, $A_1 \to A, \ldots A_n \to A$ then A	If $A_1 \vee \ldots \vee A_n$, $A_1 \to A$, $A_2 \to A \ldots$ and $A_n \to A$ are true then A is true.
Introducing $\to$	$(I:\to)$	If from $\alpha_1 \ldots \alpha_n$ infer β is proved then $\alpha_1 \wedge \ldots \wedge \alpha_n \to \beta$ is proved	If given that $\alpha_1, \alpha_2 \ldots \alpha_n$ are true and from these we deduce β then $\alpha_1 \wedge \ldots \wedge \alpha_n \to \beta$ is also true.

Rule Name	Symbol	Rule	Description
Eliminating →	(E:→)	If $A_1 \to A, A_1$, then A.	If $A_1 \to A$ & A_1 are true then A is also true. This is called Modus Ponen Rule
Introducing ↔	(I:↔)	If $A_1 \to A_2, A_2 \to A_1$ then $A_1 \leftrightarrow A_2$	If $A_1 \to A_2$ & $A_2 \to A_1$ are true then $A_1 \leftrightarrow A_2$ is also true.
Elimination ↔	(E:↔)	If $A_1 \leftrightarrow A_2$ then $A_1 \to A_2, A_2 \to A_1$	If $A_1 \leftrightarrow A_2$ is T. then $A_1 \to A_2$ & $A_2 \to A_1$ are true.
Introducing ~	(I:~)	If from A infer $A_1 \wedge \sim A_1$ is proved then $\sim A$ is proved	If from A (which is true). a Contradiction is proved then truth of ~A's also proved
Eliminating ~	(E:~)	If from ~A infer $A_1 \wedge \sim A_1$ is proved then A is proved	If from ~A, a Contradiction is proved then truth of A is also proved.

A theorem in NDS written as from $\alpha_1 \ldots \alpha_n$ infer leads to interpretation that β deduced from a set of hypotheses $\{\alpha_1 \ldots \alpha_n\}$

All hypotheses are assumed to be true in a given context and therefore theorem β is also true in same context.
Thus we can conclude that β is Consistent.

→ Description column consists of rules applied on a subexpression in proof line

→ Second column consists the subexpression obtained after applying an appropriate rule.

→ The final column consists the line number of subexpressions in the proof.

Ex. Prove that $A \wedge (B \vee C)$ is deduced from $A \wedge B$.

Sol. The theorem in NDS can be written as from $A \wedge B$ infer $A \wedge (B \vee C)$ in NDS. We can prove

Description	Formula	Comments
Theorem	from $A \wedge B$ infer $A \wedge (B \vee C)$	To be proved
Hypothesis (given)	$A \wedge B$.	1
$E: \wedge (1)$	A	2
$E: \wedge (1)$	B	3
$I: \vee (3)$	$B \vee C$	4
$I: \wedge (2,4)$	$A \wedge (B \vee C)$	proved.

hypothesis. number.

Deduction Theorem:

To prove a formula $\alpha_1 \wedge \cdots \wedge \alpha_n \to \beta$ it is sufficient to prove a theorem from $\alpha_1, \cdots \alpha_n$ infer β.

if $\alpha_1 \wedge \cdots \wedge \alpha_n \to \beta$ is proved then theorem from $\alpha_1, \cdots \alpha_n$ infer β is assumed to be proved.

Prove the theorem infer $[(A \to B) \wedge (B \to C)] \to (A \to C)$.

Sol:

Description	Formula	comments
Theorem	from $A \to B, B \to C$ infer $A \to C$	To be proved
Hypothesis 1	$A \to B$	1
Hypothesis 2	$B \to C$	2
Sub-theorem	from A infer C	3
Hypothesis	A	3.1
E: →(1, 3.1)	B	3.2
E: →(2, 3.2)	C	3.3
I: →(3)	$A \to C$	proved.

Axiomatic System

→ The axiomatic system is based on a set of three axioms and one rule of deduction.

→ Only two logical operators not (~) & implies (→) are allowed to form a formula.

→ ∧, ∨, ↔ can be easily expressed in terms of ~ & → using equivalence rules laws.

Ex:- $A \wedge B \cong \sim(\sim A \vee \sim B) \cong (\sim(A \to \sim B))$ $= (\sim A \overset{\vee}{\to} \sim B)$

$\sim(\sim A) \wedge \sim(\sim B)$

$A \wedge B$

$A \vee B \cong \sim A \to B$ $\sim(\sim A) \vee B$

$A \vee B$

$A \leftrightarrow B \cong (A \to B) \wedge (B \to A) \cong \sim[(A \to B) \to \sim(B \to A)]$.

→ Three axioms which are always True (valid) & one rule called Modus ponen (MP).

Axiom 1 $\alpha \to (\beta \to \alpha)$ $\alpha \to (\beta \to \alpha)$

Axiom 2 $[\alpha \to (\beta \to \gamma)] \to [(\alpha \to \beta) \to (\alpha \to \gamma)]$

$[\alpha \to (\beta \to \gamma)] \to [(\alpha \to \beta) \to (\alpha \to \gamma)]$

Axiom 3 $(\sim\alpha \to \sim\beta) \to (\beta \to \alpha)$.

$(\sim\alpha \to \sim\beta) \to \beta \to \alpha$

Modus Ponen rule Hypothesis : $\alpha \to \beta$ & α ; Consequent : β.

$\alpha \to \beta$ & α consequent : β.

Interpretation of Modus Ponen Rule :

Given that $\alpha \to \beta$ and & α are hypotheses (assumed to true) β is inferred (i.e., true) as a consequent.

Ex:- Establish that $A \to C$ is a deductive consequence of $\{A \to B, B \to C\}$ i.e., $\{A \to B, B \to C\} \vdash (A \to C)$

Sol:

Desc	Formula	Comments
Theorem	$\{A \to B, B \to C\} \vdash (A \to C)$	Prove
Hypothesis 1	$A \to B$	1
Hypothesis 2	$B \to C$	2
Instance of Axiom 1	$(B \to C) \to [A \to (B \to C)]$	3
MP(2, 3)	$[A \to (B \to C)]$	4
Instance of Axiom 2	$[A \to (B \to C)] \to [(A \to B) \to (A \to C)]$	5
MP(4, 5)	$(A \to B) \to (A \to C)$	6
MP(1, 6)	$(A \to C)$	Proved.

$A \to C$ is a deductive consequence of $\{A \to B, B \to C\}$.

Deduction Theorem

Given that Σ is a set of hypotheses and α & β are well-formed formulae.

If β is proved from $\{\Sigma \cup \alpha\}$, then according to deduction theorem, $(\alpha \to \beta)$ is proved from Σ.

We can write $\{\Sigma \cup \alpha\} \vdash \beta$ implies $\Sigma \vdash (\alpha \to \beta)$.

Converse of Deduction Theorem:-

The Converse of deduction theorem can be stated as:

Given $\Sigma \vdash (\alpha \to \beta)$, then $\{\Sigma \cup \alpha\} \vdash \beta$ is proved.

- If α is given, that we can easily prove $\beta \to \alpha$ for any well-formed formulae α & β.
- If $\alpha \to \beta$ is to be proved, then include α in set of hypotheses Σ and derive β from the set $\{\Sigma \cup \alpha\}$. Then, by using deduction theorem, we can conclude that $\alpha \to \beta$.

Ex: Prove $\vdash \sim A \to (A \to B)$ by using deduction theorem

Sol If we can prove $\{\sim A\} \vdash (A \to B)$ then using deduction theorem, we have proved $\vdash \sim A \to (A \to B)$

Proof of $\{\sim A\} \vdash (A \to B)$.

Description	Formula	Comments. prove
Theorem	$\{\sim A\} \vdash (A \to B)$	1
Hypothesis 1	$\sim A$	2
Instance of Axiom 1	$\sim A \to (\sim B \to \sim A)$	
MP(1,2)	$(\sim B \to \sim A)$	3
Instance of Axiom 2	$(\sim B \to \sim A) \to (A \to B)$	4
MP(3,4)	$(A \to B)$	Proved.

Semantic Tableau System in Propositional logic.

→ In NDS & axiomatic systems, forward chaining approach is used for constructing proofs & derivations.

→ In this, we start proofs or derivations from a given set of hypotheses & axioms.

→ In axiomatic, we require a guess for selection of appropriate axiom(s) in order to prove a theorem.

→ Forward chaining is good for theoritical purposes, its implementation in derivations & proofs is difficult

Semantic tableau is a binary tree which is constructed by using Semantic tableau rules with a formula as a root.

Semantic Tableau Rules Table 4.11.

Ex: Construct a Semantic tableau for a formula

$(A \wedge \sim B) \wedge (\sim B \rightarrow C)$

Sol: Construction of Semantic tableau for

$(A \wedge \sim B) \wedge (\sim B \rightarrow C)$.

Description	Formula	Line numbers
Tableau root	$(A \wedge \sim B) \wedge (\sim B \rightarrow C)$	1
Rule 1 (1)	$A \wedge \sim B$	2
	$\sim B \rightarrow C$	3
Rule 1 (2)	A	4
	$\sim B$	5
Rule 6 (3)	$\sim(\sim B)$ | C	6
Rule 3 (6)	B | ✓(open)	
	X closed $\{B, \sim B\}$.	

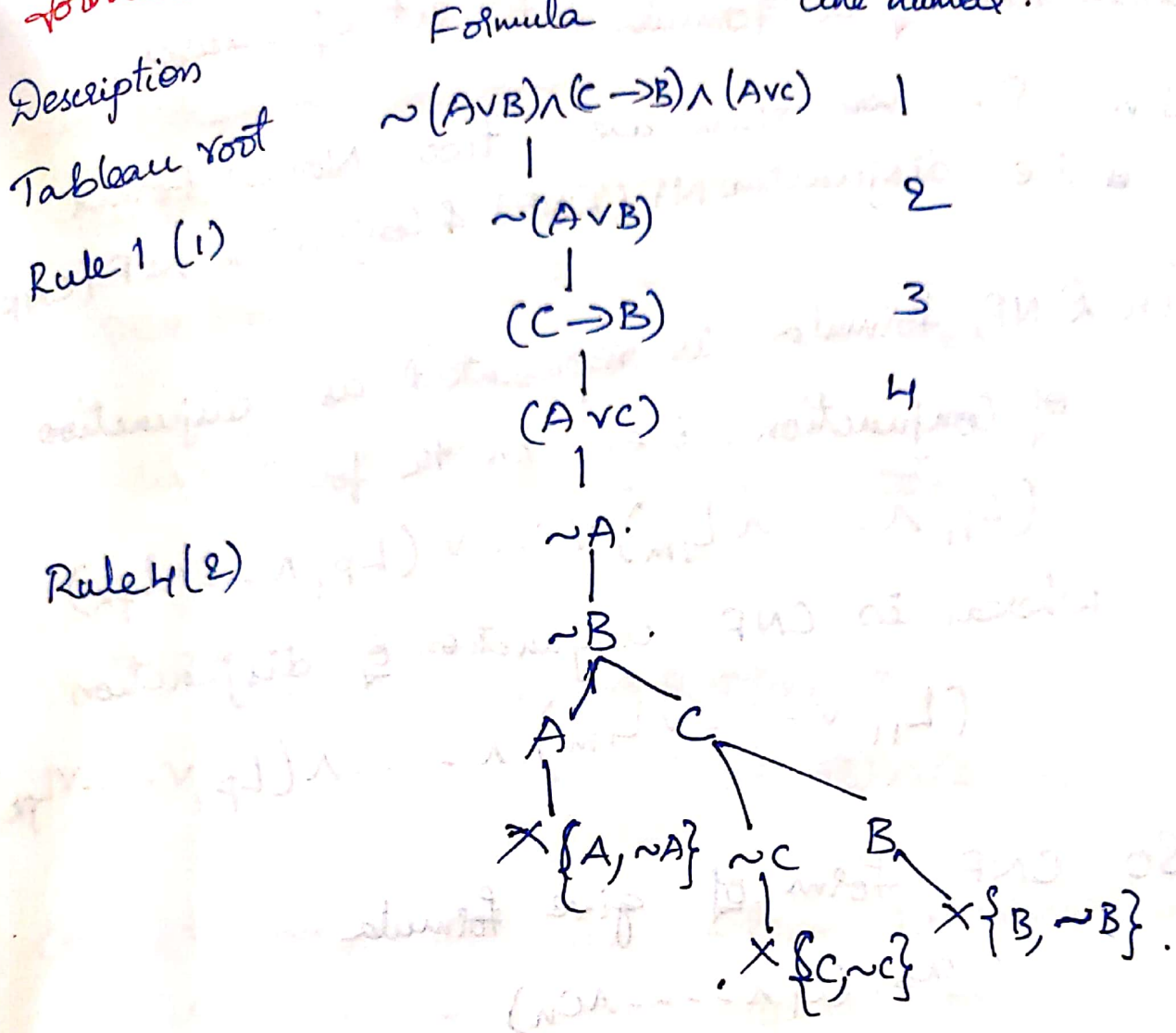
Satisfiability & unsatisfiability

Consider α to be any formula.

- A path is said to be Contradictory or closed (finished) whenever Complementary atoms appear on the Same path of a Semantic tableau. This denotes inconsistency.

- If all paths of a tableau for a given formula α are found to be closed, it is called a Contradictory tableau. This indicates that there is no interpretation or model that satisfies α.
- A formula α is said to be satisfiable if a tableau with root α is not a contradictory tableau, i.e, it has at least one open path. We can obtain a model of an interpretation under which the formula α is evaluated to be true by assigning T (true) to all atomic formulae appearing on open path of Semantic tableau of α.
- A formula α is said to be unsatisfiable if a tableau with root α is a Contradictory tableau.
- If we obtain a contradictory tableau with root $\sim\alpha$, we say that formula α is tableau provable. Alternatively, a formula α is said to be tableau provable if a tableau with root $\sim\alpha$ is a Contradictory tableau.
- A set of formulae $S = \{\alpha_1, \alpha_2 \ldots \alpha_n\}$ is said to be unsatisfiable if a tableau with root $(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n)$ is a Contradictory tableau.
- A set of formulae $S = \{\alpha_1, \alpha_2 \ldots \alpha_n\}$ is said to be satisfiable if formulae in a set are simultaneously true, that is, if a tableau for $\alpha_1 \wedge \alpha_2 \ldots \wedge \alpha_n$ has atleast one open path.

Consider a set $S = \{\sim(A \vee B), (C \to B), (A \vee C)\}$ of formulae. Show that S is unsatisfiable.



- Let S be a set of formulae. The formula α is said to be tableau provable from S (denoted by $S \vdash \alpha$) if there is a contradictory tableau from S with $\sim\alpha$ as a root.
- A formula α is said to be a logical consequence of a set S if & only if α is tableau provable from S
- If α is a tableau provable $(\vdash \alpha)$ then it is also valid $(\models \alpha)$ & vice versa.

Resolution Refutation in PL.

Conversion of a formula into set of clauses

In PL there are two Normal Forms.
i.e. disjunctive NF (DNF) & Conjunctive NF (CNF)

In DNF, formula is represented as disjunction of Conjunction, i.e., in the form

$(L_{11} \wedge \dots \wedge L_{1m}) \vee \dots \vee (L_{p1} \wedge \dots \wedge L_{pk})$

whereas in CNF conjunction of disjunction.

$(L_{11} \vee \dots \vee L_{1m}) \wedge \dots \wedge (L_{p1} \vee \dots \vee L_{pk})$.

So CNF form of give formula as $(C_1 \wedge \dots \wedge C_n)$

where C_K $(1 \leq K \leq n)$ is a disjunction of literals & is called clause.

Clause is defined as formula of form $(L_1 \vee \dots \vee L_m)$

Ex:- Set $\{A \vee B, \sim A \vee D, C \vee \sim B\}$.

represents set of clauses $A \vee B$, $\sim A \vee D$, $C \vee \sim B$.

Conversion of formula to its CNF.

double negation $\sim(\sim A) \equiv A$.

Demorgan's laws. $\sim(A \wedge B) \equiv \sim A \vee \sim B$

$\sim(A \vee B) \equiv \sim A \wedge \sim B$.

distributive law to get CNF.

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

Eliminate $\rightarrow$ & $\leftrightarrow$:

$$A \leftrightarrow B \equiv \sim A \vee B$$

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A).$$

Q. Convert the formula $(\sim[A \rightarrow B) \wedge (C \wedge \sim A)$ into its equivalent CNF representation.

$$(\sim A \rightarrow B) \wedge (C \wedge \sim A) \equiv (\sim(\sim A) \vee B) \wedge (C \wedge \sim A)$$

$$\equiv (A \vee B) \wedge (C \wedge \sim A)$$

$$\equiv (A \vee B) \wedge C \wedge \sim A.$$

Set of clauses are $\{(A \vee B), C, \sim A\}$.

Resolution of clauses

→ Two clauses can be resolved by eliminating complementary pair of literals. if any, from both, a new clause is constructed by disjunction of remaining literals in both clauses.

Note

- If C is a resolvent of 2 clauses C_1 & C_2 then C is called a logical consequence of set of clauses $\{C_1, C_2\}$. This is known as resolution principle.

- If a contradiction is derived from a set S of clauses using resolution then S is said to be unsatisfiable. Derivation of contradiction for a set S by resolution method is called a resolution refutation of S

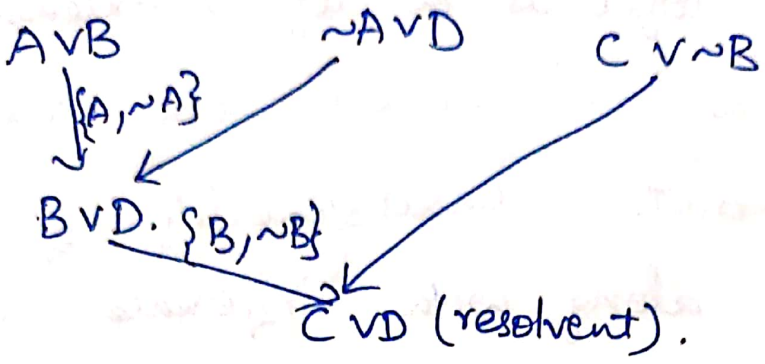
- A clause C is said to be a logical consequence of S if C is derived from S.

- Alternatively, using resolution refutation concept, a clause C is defined to be a logical consequence of S if & only if set $S' = S \cup \{\sim C\}$ is unsatisfiable, i.e., a contradiction is deduced

from set S', assuming that initially the set S is satisfiable.

Q find resolvent of clauses in set $\{A \vee B, \sim A \vee D, C \vee \sim B\}$

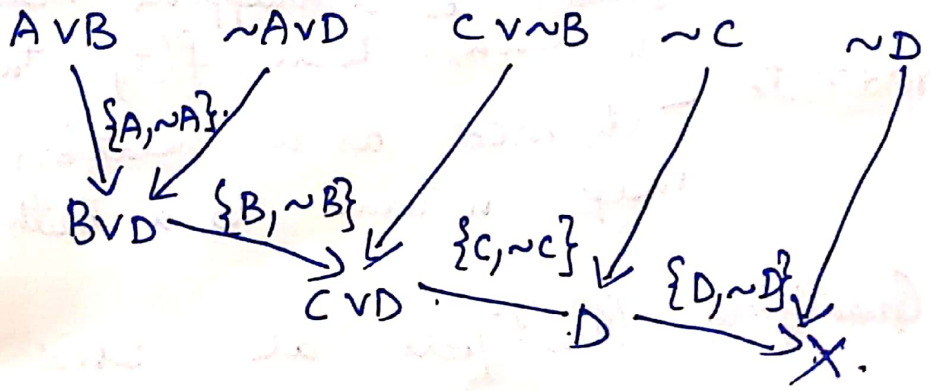
Sol.



Q. Using resolution refutation principle show that $C \vee D$ is a logical consequence of $S = \{A \vee B, \sim A \vee D, C \vee \sim B\}$.

Sol To prove that first we will add negation of logical consequence, i.e., $\sim(C \vee D) \cong \sim C \wedge \sim D$. to set S to get $S' = \{A \vee B, \sim A \vee D, C \vee \sim B, \sim C, \sim D\}$

Now we can show that S' is unsatisfiable by deriving contradiction using resolution principle.



we get contradiction from S' we can conclude that $(C \vee D)$ is a logical consequence of $S = \{A \vee B, \sim A \vee D$

Predicate logic.

→ Statements like All birds fly can't be represented in propositional logic.

→ Predicate logic is a logical extension of propositional logic. deals with validity, Satisfiability & unsatisfiability of a formula along with inference rules for derivation of a new formula.

→ Predicate calculus is study of predicate systems, when inference rules are added to predicate calculus, it becomes predicate logic.

Predicate Calculus

→ Term — A term is defined as either a variable or constant. & n-place f^n. A f^n is defined as a mapping that maps n terms to a single term $f(t_1, \ldots t_n)$; $t_1 \ldots t_n$ are terms

→ Predicate — defined as a relation that maps n terms to a truth value (T, F)

→ Quantifiers. Quantifiers are used with variables, "universal ($\forall$) Existential ($\exists$) there exists forall.

Well formed formula

- Atomic formula $P(t_1 \ldots t_n)$ (also called an atom) is a well-formed formula, where P is a predicate symbol and $t_1 \ldots t_n$ are terms.

- If α & β are well-formed formulae, then $\sim(\alpha)$, $(\alpha \vee \beta)$, $(\alpha \wedge \beta)$, and $(\alpha \leftrightarrow \beta)$ are well formed formulae.

- If α is a well-formed formula & x is a free variable in α, then $(\forall x)\,\alpha$ and $(\exists x)\,\alpha$ are both well formed formulae.

- Well formed formulae may be generated by applying the rules.

Ex:- A Stmt x is brother of y can be easily represented in predicate Calculus as brother (x,y) which maps it to true or false. when x & y gets instantiated to actual values.

- A Stmt Peter loves his son is represented as love ("Peter", Son("Peter")).

  Here Son is a fn that maps peter to his Son & love is a predicate name which takes two terms & maps them to 'T' or 'F' depending on values of its terms.

be numbers.

First-Order Predicate Calculus (FOPC)

If Quantification in predicate formula is only on simple variables & not on predicates of functions then it is called (FOPC).

If quantification is over first-order predicates & functions, then it becomes second-order predicate calculus.

Ex:- $\forall p(P(x)) \leftrightarrow P(y))$ is second-order predicate statement.

$\forall x \forall y (P(x) \leftrightarrow P(y))$ is 1st-order predicate stmt.

→ Adding inference rules added to first-order predicate calculus, it becomes first-order predicate logic

Interpretations of formulae in FOL

→ In propositional logic, an interpretation refers to an assignment of truth values to atoms.

→ Consists of non-empty domain D and involves assignment of values to each constant, function symbol & also an assignment of

truthvalues to each predicate atom.

The following results hold true for any Interpretation I over a domain D.

- $(\forall X)P(X)$ = true if & only if $P(x)$ = true, $\forall x \in D$ otherwise it is false.
- $(\exists X)P(X)]$ = true if & only if $\exists c \in D$ such that $P(c)$ = true, else false.

Q. Evaluate the truth value of an FOL formula $\alpha: (\forall X)(\exists Y)P(X,Y)$ under following interpretation I:

- D = {1,2}
- P(1,1) = F, P(1,2) = T, P(2,1) = T, P(2,2) = F.

Sol:- Let us denote true by T & false by F.

For X = 1, then $\exists 2 \in D$ such that P(1,2) = T

~~for X=2, then ∃1~~

for X = 2, then $\exists 1 \in D$ such that P(2,1) = T. Hence, α is true under interpretation I.

(ii) Evaluate $\alpha = (\forall X)[P(X) \rightarrow Q(f(X), c)]$ under following interpretation.

- D = {1,2}
- c = 1 (c is a constant from the domain D)
- f(1) = 2, f(2) = 1.

- $P(1) = F$, $P(2) = T$
- $q(1,1) = T$, $q(1,2) = T$, $q(2,1) = F$, $q(2,2) = T$

Sol: For $x = 1$.

$P(1) \rightarrow q(f(1), 1) \equiv P(1) \rightarrow q(2,1) \equiv F \rightarrow q(2,1) \equiv T$

For $x = 2$

$P(2) \rightarrow q(f(2), 1) \equiv P(2) \rightarrow q(1,1) \equiv T \rightarrow q(1,1) \equiv T \rightarrow T \equiv T$

we can say that α is true for all values of $x \in D$ under interpretation I.

Logic.

→ Logic is the study of what follows from from a set of premises. Conclusions.

→ Study of principles of correct reasoning.

→ we study principles Governing the validity of arguments & check what whether certain Conclusion follows from some given assumption.

Ex:- Alice likes everyone who likes logic

Alice like logic.

Alice likes himself.

Is argument valid ? How?

→ logic process takes in some information called premises & produces some o/ps called Conclusions.

First-order Predicate logic

1) FOPL was developed to extend the expressiveness of Proposition logic
2) It is generalization of propositional logic that permits reasoning about world entities as well as classes & subclasses of objects.
3) Predicate logic uses variables & quantifiers

Knowledge Representation

→ Predicate logic used for storing knowledge & interfacing new knowledge while solving problems requiring concepts of AI.

→ Main focus is on storing knowledge of information in such a manner that programs can process it and achieve human intelligence.

→ The ways with which knowledge is represented may solve a problem easier.

→ Goal of knowledge representation is to represent knowledge in a manner that facilitates the process of inferencing (concluding) from it.

→ Any knowledge representation system should possess properties such as learning, efficiency in acquisition, representational adequacy & inferential adequacy.

→ An expert system that incorporates knowledge in the form of rules & facts is called a rule-based expert system.

Approaches to Knowledge Representation

→ AI programs uses structures known as knowledge structures to represent objects, facts, relationships & procedures.

→ Main function of these knowledge structures is to povide expertise & information so that a program can operate in an intelligent way.

→ Knowledge structures are composed of both traditional & complex structures. such as semantic n/w, frames, Scripts, Conceptual dependency.

→ Knowledge bases share some f^n of dbs such as storing, updating, retrieving inf.

Relational Knowledge

It comprises objects consisting of attributes & associated values.

Relational table

Name	Age	Sex	Qual	Salary.
John	38	M	Graduate	20,000
Mike	25	M	U.G	15,000
Mary	30	F	Ph.D	25000
James.	29.	M	Grad	18000

Knowledge Represented in logic

All humans are mortal.

$(\forall x)$ human $(x) \leftarrow$ mortal (x).

We can represent set of rules, derive more facts, truths, & verify correctness.

Procedural Knowledge

encoded form of procedures which carry out specific ~~task~~ tasks based on relevant knowledge.

Relational Knowledge

It comprises objects consisting of attributes & associated values.

Relational table

Name	Age	Sex	Qual	Salary.
John	38	M	Graduate	20,000
Mike	25	M	U.G	15,000
Mary	30	F	Ph.D	25000
James.	29.	M	Grad	18000

Knowledge Represented in logic

All humans are mortal.

$(\forall x)$ human $(x) \leftarrow$ mortal (x).

We can represent set of rules, derive more facts, truths, & verify correctness.

Procedural Knowledge

encoded form of procedures which carry out specific ~~task~~ tasks based on relevant knowledge.

Knowledge Representation using Semantic N/w

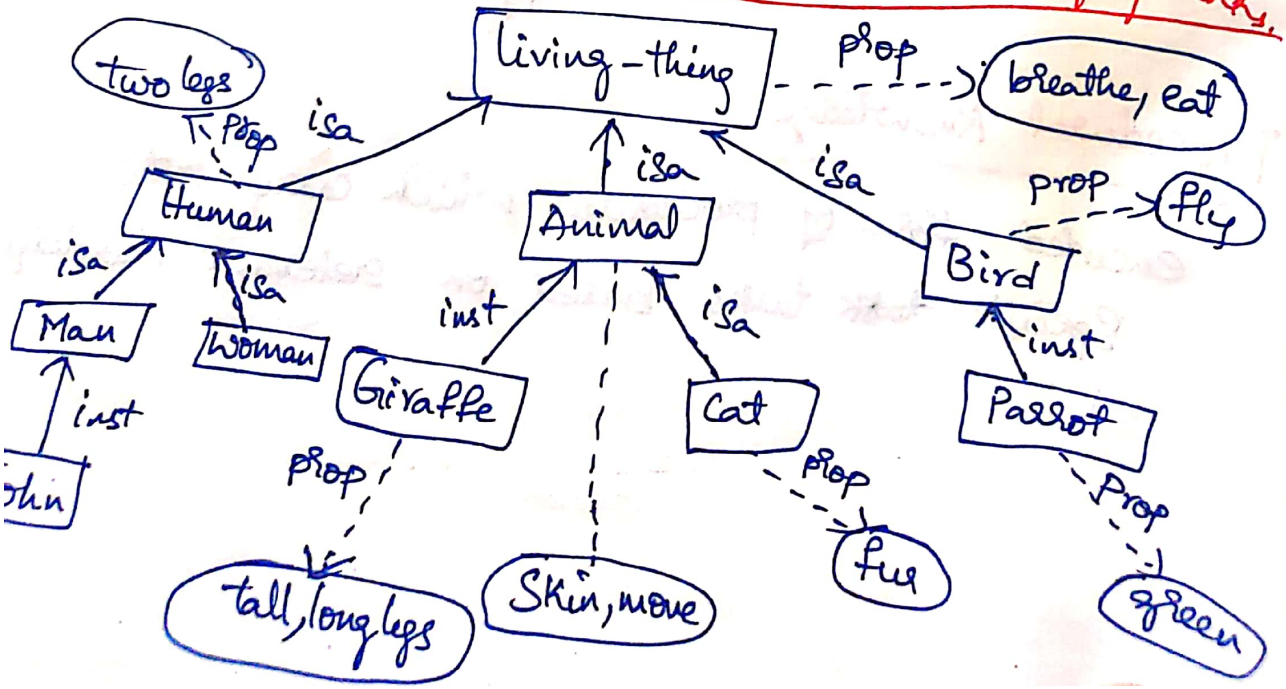
Meaning of a Concept is derived from its relationship with other Concepts.

KR using Semantic Net

→ isa – relation connects 2 classes.

→ inst – relation relates Specific members of a class.

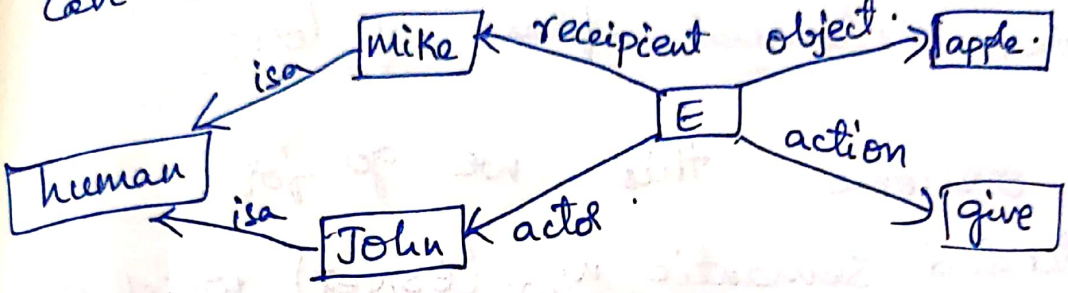
fig. Concepts Connected with prop links.



Inheritance in Semantic Net.

Extended Semantic Networks for KR.

John gives an apple to mike.
John and mike are human.
Can be represented in Semantic Network.



E represents an event ~~obi~~ i.e., act of giving

clausual form of logic

object (E, apple)
action (E, give)
actor (E, john)
receipient (E, mike)
isa (john, human)
isa (mike, human)

Semantic net can be coded using binary representation, Such representation is easy.
& advantageous when additional information is added to given System.
Ex:. john gave an apple to mike in kitchen.
location (E, kitchen) is added.

john gives an apple to everyone he likes

give (john, X, apple) ← likes (john, X).

Conclusions Conditions.

In Conventional Semantic N/W, we can't express clausal form of logic

To overcome this we go for extended Semantic N/W. (ESNet). which was proposed by R. Kowalski & Colleagues.

ESnet is powerful representation as compared to logic & Semantic N/W.

In ESnet. Constant, variable & f^n al terms are represented by Constant, variable & f^n al nodes resp.

Binary predicate Symbols in clausal logic. are represented by labels on arcs. of ESNet.

love (john, mary).

john —love→ Mary.

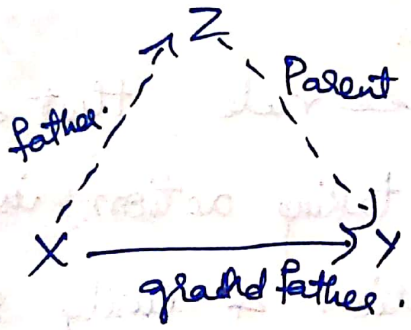
Conclusions & Conditions of clausal form are represented in ESNet.

Conditions (-ve atoms) are with dotted arrow lines. (<----)

Conclusions (+ve atoms) are with continuous lines. (---->).

grandfather(X,Y) ← father(X,Z), parent(Z,Y).

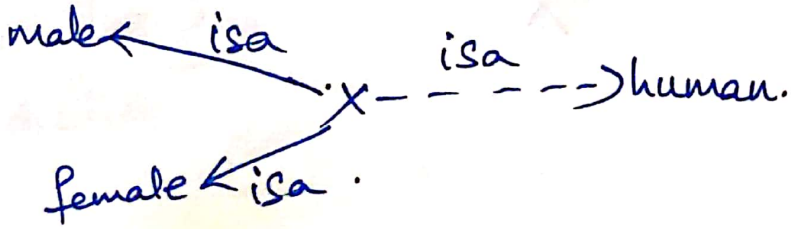
Conclusions. Conditions



ESNet representation

Illy male(X), female(X) ← human(X)

can be represented using binary representation as isa(X, male), isa(X, female) ← isa(X, human).



Inference rules

- representation of inference for every action of giving, there is an action of taking in clausal logic is

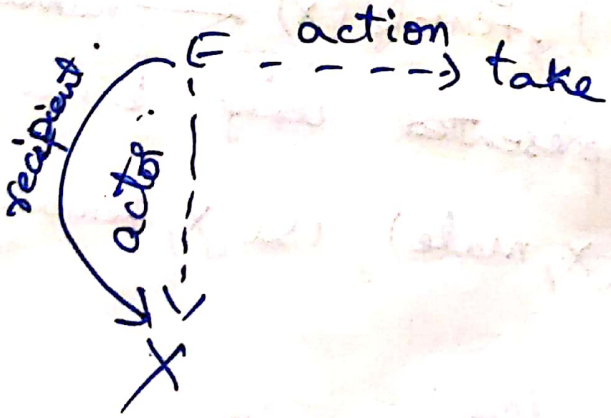
action(f(x), take) ← action(X, give).

$f(x) \xrightarrow{\text{action}}$ take.

$X \xrightarrow{\text{action}}$ give.

- The inference rule that an actor who performs a taking action is also recipient of this action & easily represented in clausal logic.

recipient(E, x) ← action(E, take), actor(E, x).



Where Threshold (

Q. Represent the following clauses in ESNet.

recipient (E, X) ← action (E, take), actor (E, X)

E → some event

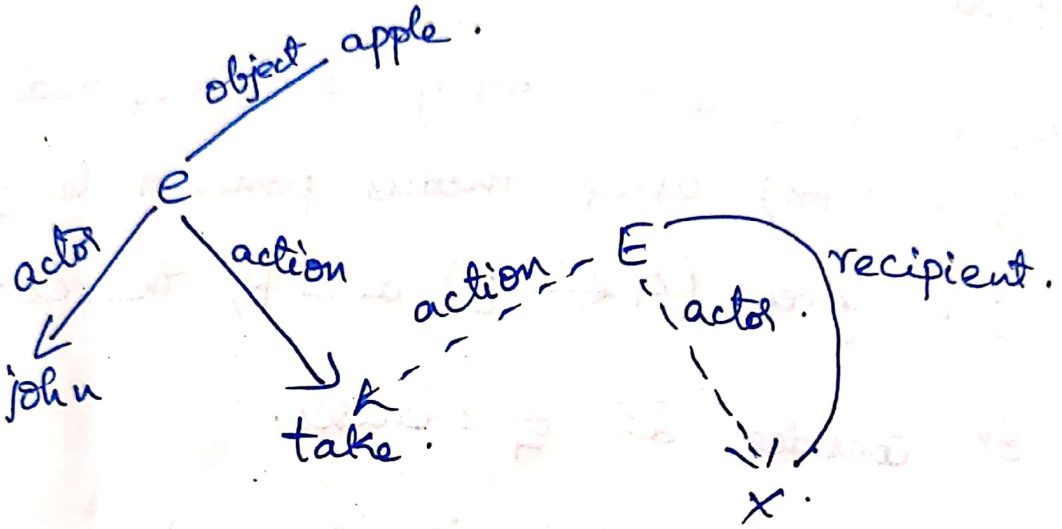
object (e, apple)

e → actual event.

action (e, take)

actor (e, john).

Sol:-



hierarchy links isa, inst, part_of (& prop).

are available in Sem.ESNet also.

part_of link is a hidden existential quantifier.

Ex:- every human has two legs

two_legs —Part_of→ human.

e.g. every human, there exist two legs which are part of that human

Contradiction.

P —Part_of→ X.

P —Part_of→ Y.

X —isa→ Y.

Deduction in Extended Semantic N/ws.

logic are two types of inference mechanisms

- forward reasoning inference mechanism.
- Backward " " "

forward Reasoning Inference.

Given an ESNet. apply following reduction. (resolution) using modus ponen rule of logic i.e., given (A ← B) and B, then conclude A.

Ex:- Consider Set of clauses.

isa (X, human) ← isa (X, man)

isa (john, man).

Using modusponen rule of logic, we can easily derive that isa (john, human) holds true.

new assertion isa (john, human) is inferred by elimination of assertion & their denial links.

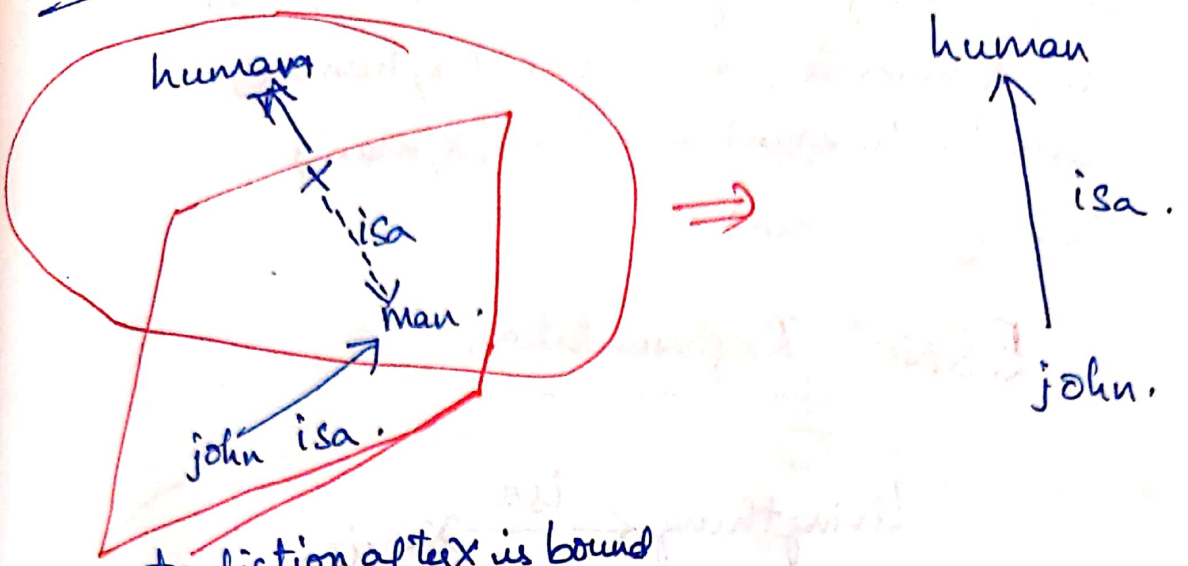
Given Set of clauses

isa (X, human) ← isa (X, man)

isa (john, man)

inferencing

isa (john, human)



Contradiction of X is bound to john.

Backward Reasoning inference.

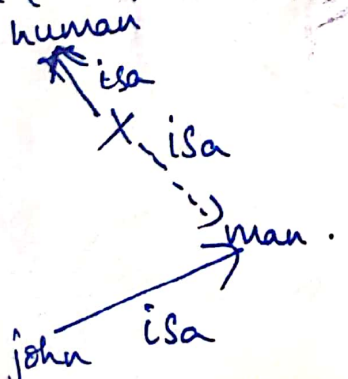
we can prove a conclusion from a given ESNet by adding denial of conclusion to n/w.

prove isa(john, human).

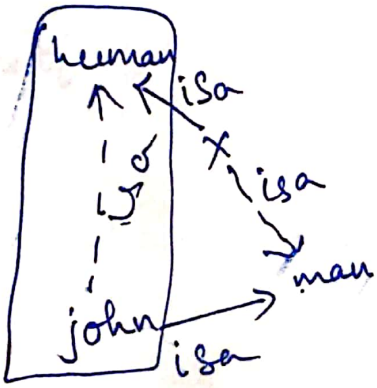
Given Set of clauses

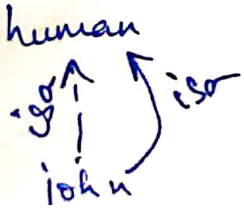
isa(X, human) ← isa (X, man)

isa (john, man).



prove conclusion



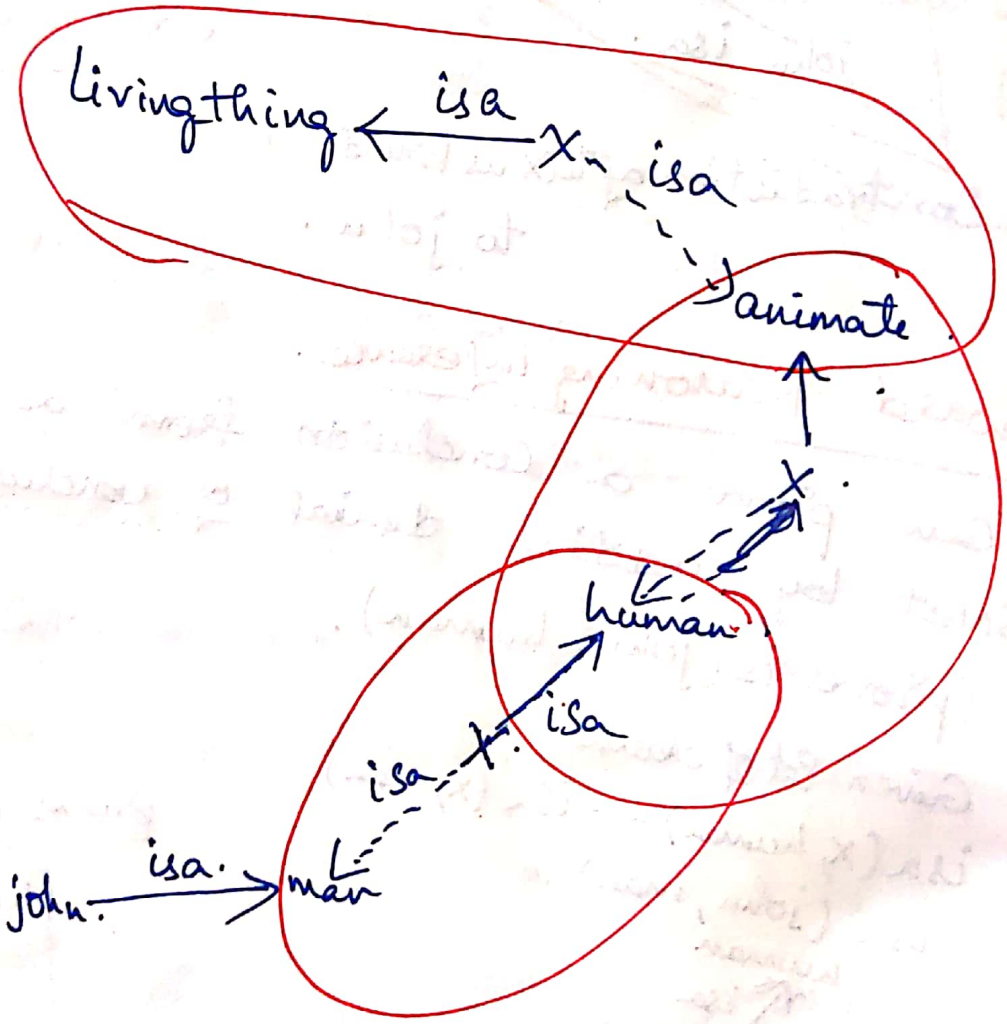


Example for Illustrating Inferencing methods.

Consider the following example.

isa(X, living_thing) ← isa(X, animate)
isa(X, animate) ← isa(X, human)
isa(X, human) ← isa(X, man)
isa(john, man).

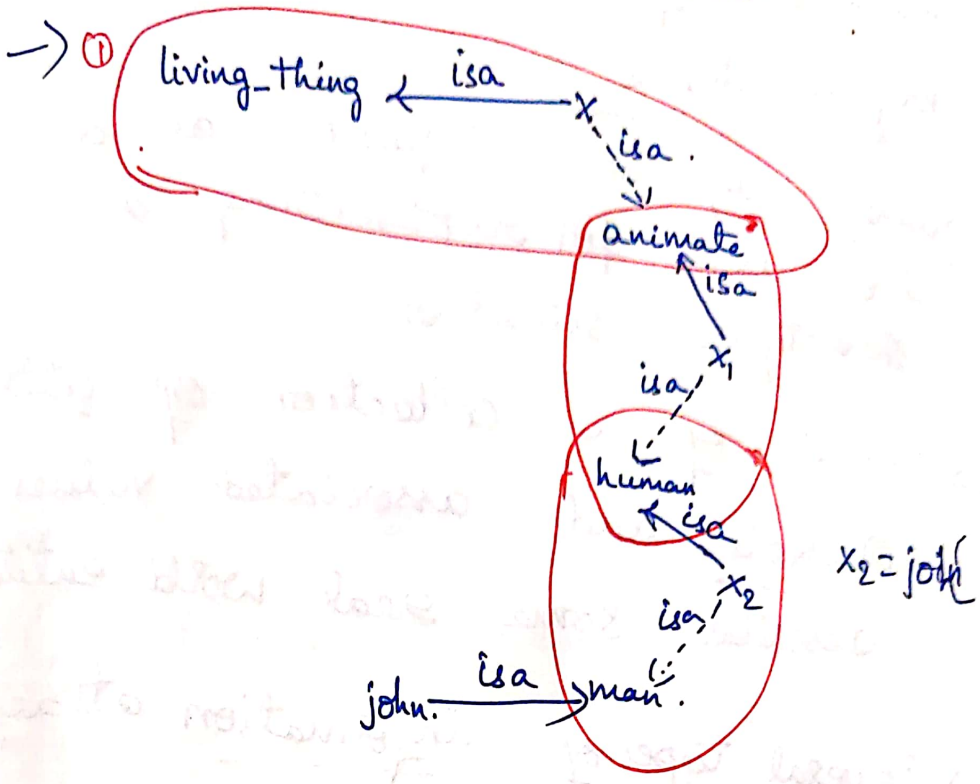
ESNet Representation.

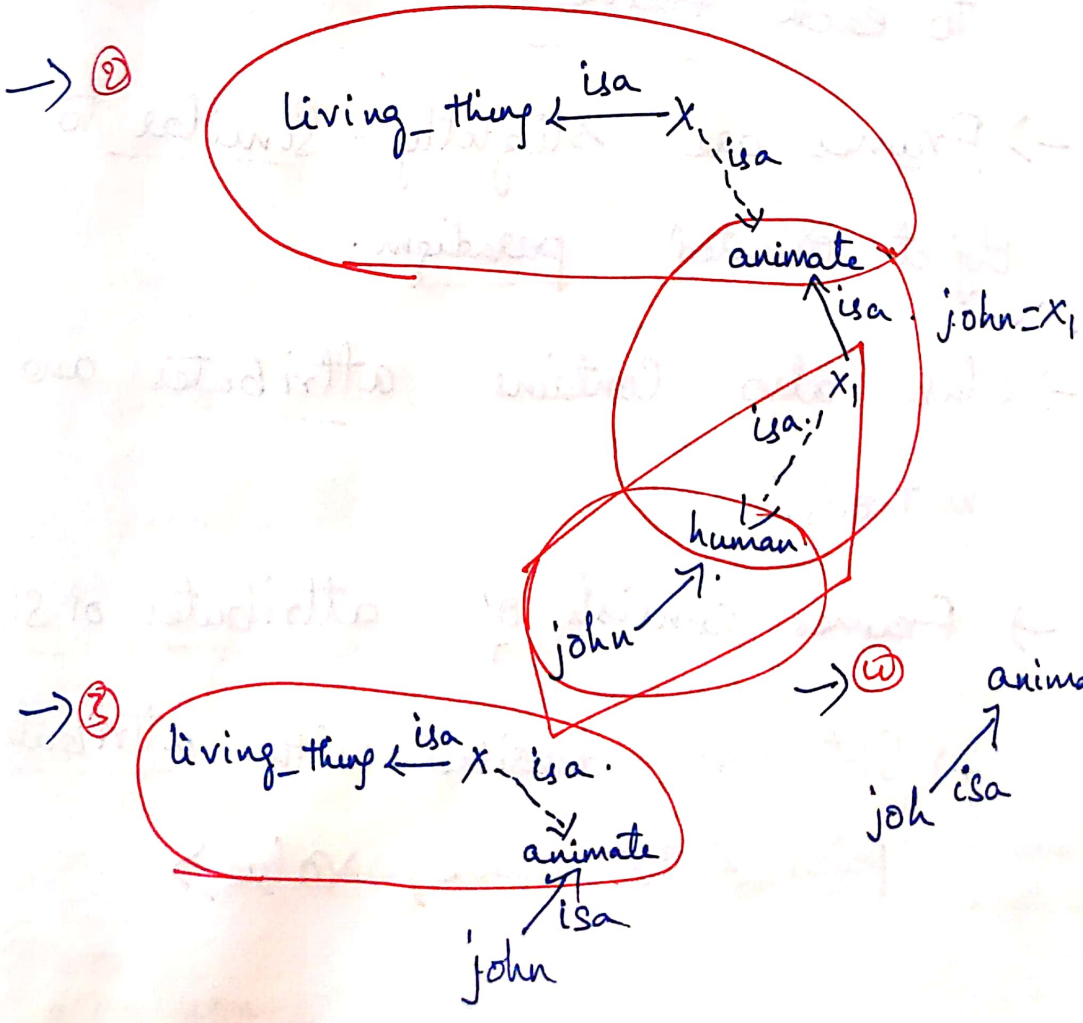


well defined mathematical form,

Forward Reasoning Inference

new assertion john is ~~human~~ animate.





KR using Frames

→ each node of a Semantic net is represented by a frame.

→ Frame may be defined as a data structure i.e. used for representing a stereotyped situation..

→ Consists of a collection of attributes or slots and associated values that describe some real world entity.

→ Several types of information attached to each frame.

→ Frames are slightly similar to object oriented paradigm.

→ class also contains attributes and methods.

→ Frames consists of attributes or slots.

→ Slots are described with attribute-value pairs <Slot_name, value>.

Can

→ Slots have facets describing their properties.

→ value of soft slot may be primitive, such as text string, constant or integer or may be another frame.

→ Slot may contain value, refer to other frames or methods.

→ Frames may contain triggers for checking consistency or obtaining updates of other slots.

→ Frames are basically a m/c usable formalization.

→ Structure of a frame. pg no. 255. in Saroj Kowshik.

→ list of facets in a frame.

→ Hospital frame.

→ Frame Descripts.

links

AKO: link connects two frames.

Inst: Links connects particular instance frame to class frame. Ex: AIIMS is an instance of class frame hospital.

Part of: Connects two class frames one of which is contained in other class.

drawbacks of Contingency plan.

→ logical agent must Consider every logically possible explanation for observation & leads to Complex belief - state representations

→ handles every eventuality can grow large & must Consider arbitrarily unlikely contingencies

→ There might be NO plan that is guaranteed to achieve goal yet agent must act.

It must have some way to compare merits of plans that are not guaranteed.

Ex:- A_{90}.

In some sense A_{90} is fact the right thing to do rational decision - ∴ depends on both relative importance of various goals & likelihood that & degree to, which, they will be achieved.