Unit-3 First order Logic Artificial Intelligence

Introduction:-

→ In the topic of propositional logic
→ how to represent the statements using propositional logic
→ But the PL represents only "facts"
→ which are either true(or) false
⇒ But PL is not represent the "complex sentences" or "natural language statements."
→ The PL is has very limited expressive power
→ which we cannot represent using PL Logic
→ Where as "PL only represent that the world contain facts like: natural language - more than PL
ex: some human are intellegent or sharp like Ravi
Let Assume:-
→ **Objects**:- people, houses, numbers, theories, colours Baseball, volly ball, cricket, all games wars - I, II, III, centuries 19th, 17th, ...
← **Relations**:- A, B, people, numbers, theories, squares. Red, Round, is adjecent or n-ary relation such as:- the sister of, Brother of, has colour comes blw.
→ **Functions**:- Father of best friend.
third joining of, End joining of, end of
Ex: ① son of, ② best friend of
③ father of

PL is not sufficient to represent the complex sentences (natural language)

Pg - 2

First order logic:- is another way of knowledge representation in "artificial intelligence"

→ It is extension to propositional logic

→ FOL is sufficiently expressive to represent the natural language.

→ FOL is known as "predicate logic"

→ FOL logic is powerful language, that develops Information about objects in a more easy way and can also express the relationship b/w those objects

→ FOL logic assumes the following things

⟹ Natural language (FOL) also has two main parts

a → * Syntax

b → * Semantics

→ **SYNTAX :-** syntax of FOL :- it determines which collection of symbols is a logical expression in FOL so we can write the (short-hand) notation in FOL

Following the Basic elements:-

1.	constant	1, 2, A, Joh, Rama, Mumbai, ... categies
2.	Variables	x, y, z, a, b, c, d ..... " "
(3)	predicates	Brother, sister, Friend, Father, ... "
4.	Function	sqrt, LeftLegof, BestFriendof, daughterof
5.	connectives	$\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$
6.	Equality	$=$ $=$
7.	Quantifier	$\forall$, $\exists$

Atomic sentences:- It can represented as, "relations blw

Predicate (term 1, term 2, ... term n)

Ex:- Ravi and Ajay are brothers $\Rightarrow$ Brothers (Ravi, Aja
Radha likes krishna $\Rightarrow$ Likes (Radha, Krishna)
suma and Rama are sister $\Rightarrow$ sisters (suma, Rama
Ram likes mango $\Rightarrow$ Likes (ram, mango).
seema is a girl $\Rightarrow$ girl (seema).
Rose is red $\Rightarrow$ red (Rose).
Jhon owns gold $\Rightarrow$ owns (Jhon, gold)

Program:-

clauses
likes (ram, mango).
girl (seema).
red (rose).

Output:-

queries

"Quantification" : Quantification specifies — quantity of specimen in the universe.

① Universal Quantifier :- It is a logical representation. which specify that statement "within its range is True. for every thing(or

→ every Instant of a Particular thing (everything, eve, whe

→ ∀ → Inverted A
X → for all
⇒ ∀x
⇒ every man Respect his parent.

⇒ In this Question, the predicate is "Respect (x,y)".
where x = man ; y = parent

⇒ "every man. will use ∀ and it will be represented as ∀x man(x) → Respect (x, parent

⇒ Existential Quantifier :- which expresses the statement within the scope is true for atleast one instant of some things

→ ∃ :- It is a conjunction symbol.

Ex: Some Boys Plays Cricket
"paly (x,y)", : x = boys and y = game
some boys so we will use ∃ and it will be
Represented as ∃x boys(x) → play (x, Cricket)

→ Ex : Not all student like both Maths and Scien
Like (x,y) x = student y = subject
so all student repre ⇒ ∀ ⇒ ∀(x) [studen (x) → like (x, maths, y (scien)

(3)

Pg :6

using free variable, it occurs outside the within the scope of quantifier

Ex : $\forall x [A(x) B(y)]$, here x, y are bond variable

Connectives

logical connectives are used to connect two simpler propositions / representing a sentence logically. There are 5 connectives.

① **Negation**: A sentence such as $\neg P$ is negation of P. A literal can be either positive or negative literal.

pronunciation (NOT) • ¬

Table:

P	¬P
True	False
False	True

② **Conjunction ($\wedge$)**: A sentence which has $\wedge$ connective such as $P \wedge Q$ is called Conjunction.

Eg: Rohan is Intelligent & hardworking.

P = Rohan is intelligent
Q = Rohan is hardworking

$P \wedge Q$

Pronunciation (AND) - $\wedge$

TT:

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

③ **Disjonction (V)** : Pronounciation : OR

Sentence has V Connection such as P V Q called Disjonction where P & Q are Propositions

Eg: Rithika is a doctor or Engineer ; (TT)

P is Rithika is doctor
Q is Rithika is Engineer
= P V Q

P	Q	P V Q
T	T	T
F	T	T
T	F	T
F	F	F

④ **Implication (→)** Pronounciation : Implies

Sentence such as P → Q is Implication.
Implication are also known as if-then rules.

Eg: If it is Raining, then the street is wet
P Q

∴ P → Q

P	Q	P → Q
T	T	T
T	F	F
F	T	T
F	F	T

⑤ **Biconditional (⟺)**

Pronounciation ⟺ if and only if

P ⟺ Q is Biconditional Sentence.

Eg: If I am breathing ; then I am alive
P Q

P ⟺ Q

P	Q	
T	T	T
T	F	F
F	T	F
F	F	T

Unification:- # To find common solution for expressions containing variables

It is a process of making two different logical Atomic expressions identical by finding a substitution.

Unification depends on substitution process.

→ It takes two literals as input & makes them identical using substitution.

Ψ_1 & Ψ_2 be two Atomic sentences & σ be a unifier such that

$$\Psi_1\sigma = \Psi_2\sigma$$

then expressed as, UNIFY (Ψ_1, Ψ_2)

Eg: Unify {king(x), king(John)}

Let - Ψ_1, Ψ_2

Substitution - θ = {John/x} is unified for these atoms,

UNIFY ALGORITHM:

→ Used for unification which takes two Atomic sentences & returns a unified for those sentences.

→ Unification is key component of all First Order Inference Algorithms.

→ It return fail, if expressions don't match each other.

→ The substitution variables are called Most General unifier / MGU.

Eg. let's say, Two different expressions,
P(x, y) and P(a, f(z))

we need to make both statements identical to each other, for this we perform substitution.

P(x, y) → ①

P(a, f(z)) → ②

→ Substitute x with a, y with f(z) in first Expression, represented as a/x, & f(z)/y

→ with both substitutions, first Expression will be Identical to second expression,

substitution set: [a/x, f(z)/y].

Condition for unification

→ Predicate symbol must be same, Atoms & Expressions with different predicate symbol can never be unified.

→ No. of arguments in both Expressions must be identical.

→ Unification will fail if there are two similar variables present in same Expression

---

Eg Code: ~~match(...), match(...)~~
John knows x, John knows Jane
UNIFY(knows(John, x), knows(John, Jane)) =
{x / Jane}

φ_1 : knows (Richard, x)

φ_2 : knows (Richard, John)

so ⇒ { knows (Richard, x) ; knows (Richard (John)}

s subst θ = { John/x }

Unification

Code eg:

(1) parent (A, B, C) — unify succeed

parent (Kevin, Kenny, 30) — not unified

(2) parent (Kevin, y, 25) — 25 not unified to 30, failed

parent (x, Henry, 30)

(3)

notepad

```
woman(mia).
woman(jody).
```

prolog

```
?- women(X).
X = mia
```

prolog unifies woman(X) with woman(mia)

∴ Instantiating variable x to mia

(4) ① P(x, a, b) ② P(y, z, b)

① check predicates, 'p' same in both expression

② x with y (x/y), a with z (a/z), b with b (not needed)

P(y, a, b) P(y, z, b)

Forward Chaining:- (Data-driven technique)

Forward chaining is also known as a forward deduction (or) Forward Reasoning method when using a Inference Engine.

→ Forward chaining is a form of reasoning which start with atomic sentences in the knowledge Base & Applies Inference rules (Modus ponens) in forward direction to extract More Data until a goal is reached.

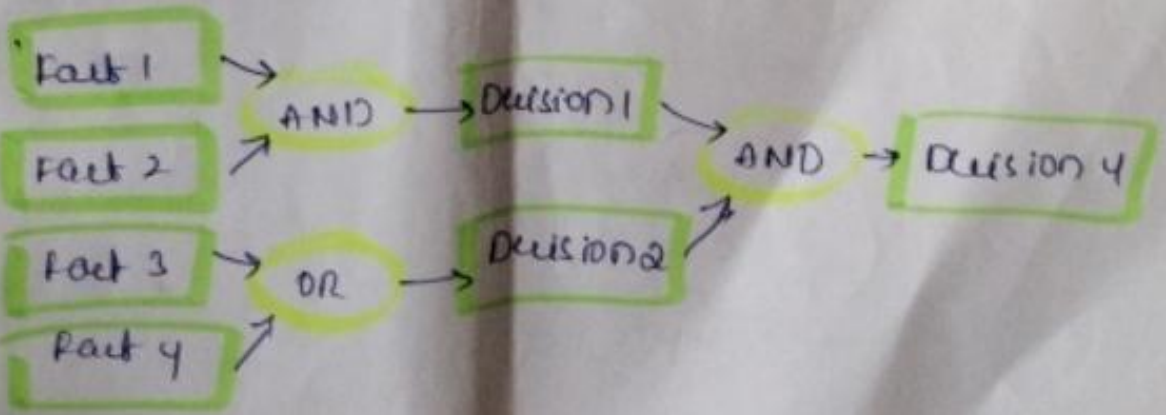
FC algorithms starts from known facts, triggers all rules whose premises are satisfied and add their conclusion to the known facts. This process repeats until the problem is solved.

Eg: A
A → B fact
B

he is running.
If he is running, he sweats.
he is sweating.

→ A is starting point
→ This fact is used to achieve Decision B.

① Change the clauses to First order Predicate Logic
② Distinguish facts from generated clauses.
③ Make note of what we need to establish at the end.



Forward chaining & Backward chaining in AI

In AI, Forward & Backward chaining is one of Important topics.

Inference Engine:

The Inference Engine is the Component of the Intelligent System in AI, which applies logic Rules to knowledge Base to Infer new Information from known facts.

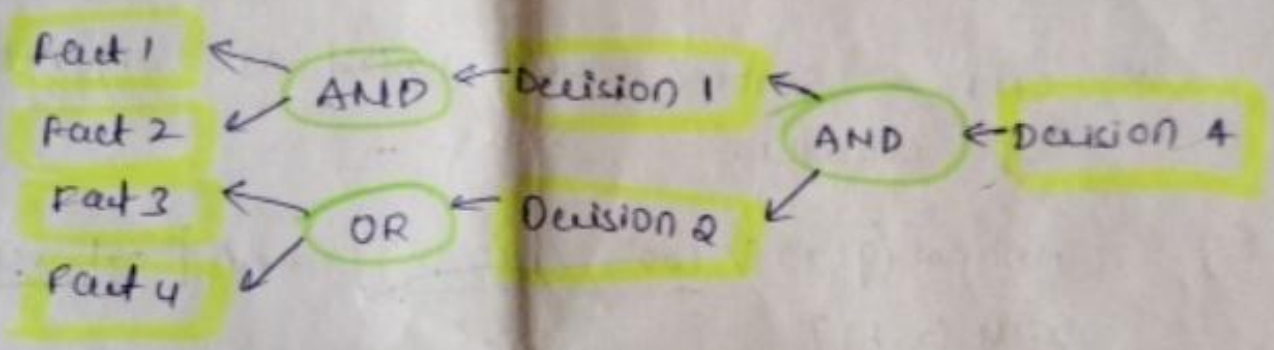
→ First Inference Engine was part of Expert System. Inference Engine Commonly proceeds in two modes. which are:

(a) Forward chaining
(b) Backward chaining

Backward chaining (Goal-driven technique) (5)

→ It is a concept on AI which involves Backtracking from the endpoint / goal to step that lead to endpoint

→ This type of chaining starts from the goal and moves Backward to comprehend the steps that were taken to attain this goal.



Steps:

Firstly, goal state & Rules are selected where the goal state is reside in Then part as conclusion.

Eg: B # B is Endpoint, used be starting point for Backward tracking

A → B Fact

A # Initial State

Tom is sweating (B)

If a person is ~~sweats~~ Running, he will sweat (A → B)

Tom is Running (A)

Resolution in FOL:

Resolution is a theorem proving technique that proceeds by

Resolution is used if there are various statements are given, & we need to prove a conclusion of those statements.

⇒ Resolution is a single Inference rule which can efficiently operate CNF (Conjuctive normal form).

* CNF → sentence represented as conjunction of clauses.

Eg: [Animal(g(x) V Loves(f(x), x)] and [¬Loves(a, b) V ¬ Kills(a, b)]

Two complimentary literals: Loves(f(x), x) and ¬ Loves(a, b)

two Literals unified with unifier θ = [a/f(x) and b/x]

⇒ [Animal(g(x) V ¬ Kills(f(x), x]

eg:

a. John likes all kind of food.
b. Apple and vegetable are food.
c. Anything Anyone eats and not killed is food.
d. Anil eats peanuts and still alive.
e. Harry eats everything that Anil eats.
f. John likes peanuts.

→ Statements in FOL

(1) $\forall x : \neg food(x) \rightarrow likes(John, x)$

$food(Apple) \wedge food(vegetables)$

$\forall x \forall y : eats(x,y) \wedge \neg killed(x) \rightarrow food(y)$

$eats(Anil, peanuts) \wedge Alive(Anil)$.

$\forall x : eats(Anil, x) \rightarrow eats(Harry, x)$

$\forall x : \neg killed(x) \rightarrow alive(x)$ } Added Predicates

$\forall x : alive(x) \rightarrow \neg killed(x)$ } Added Predicates

$likes(John, peanuts)$.

(2) Conversion of FOL into CNF

* Eliminate all implication → & also move negation (¬) inwards & rename variables

$\forall x \neg food(x) \vee likes(John, x)$

$food(APPle) \wedge food(vegetables)$

$\forall x \forall y \neg eats(x,y) \vee killed(x) \vee food(y)$

$eats(Anil, Peanuts) \wedge alive(Anil)$

$\forall x \neg eats(Anil, x) \vee eats(Harry, x)$

$\forall x \neg killed(x) \vee alive(x)$

$\forall x \neg alive(x) \vee \neg killed(x)$

$likes(John, Peanuts)$

→ Rename variables,

$\forall x \rightarrow food(x) \vee likes(John, x)$

$food(APPLE) \wedge food(vegetables)$

$\forall y \forall z \rightarrow eats(y, z) \vee killed(y) \vee food(z)$

$eats(Anil, peanuts) \wedge alive(Anil)$

$\forall w \rightarrow eats(Anil, w) \vee eats(Harry, w)$

$\forall q \rightarrow killed(q) \vee alive(q)$

$\forall k \rightarrow alive(k) \vee \rightarrow killed(k)$

$likes(John, peanuts)$.

→ Eliminate existential instanciation quantifier by Elimination. $(\exists)$

* Here, no existential quantifier so all statements will remain same.

→ Drop universal quantifiers $(\forall)$

a. $\rightarrow food(x) \vee likes(John, x)$

b. $food(APPLE)$

c. $food(vegetables)$

d. $\rightarrow eats(y, z) \vee killed(y) \vee food(z)$

e. $eats(Anil, peanuts)$

f. $alive(Anil)$

g. $\rightarrow eats(Anil, w) \vee eats(Harry, w)$

h. $killed(g) \vee alive(g)$

i. $\rightarrow alive(k) \vee \rightarrow killed(k)$

j. $likes(John, Peanuts)$

John likes peanuts

John likes anything

(i) likes (John, peanuts)
likes(John, x)

unify: [x = peanuts].

(2). Resolution : FOL → (CNF) Conjuctive normal form.

likes (John, peanuts)
likes (John, x)

John, John;
~~unified~~ &
resolved
cancelled

left,
food (peanuts)

* unification is key concept of resolution

FOL Inference Rules for quantifier:

As propositional logic, we also have inference rules in First-order logic.

Basic Inference rules in FOL:-

1. Universal Generalization
2. Universal Instantiation
3. Existential Instantiation
4. Existential introduction.

① Universal generalization : Is a valid Inference rule which stated if premise P(c) is true for any arbitary element c in universe of discourse,

$\therefore \forall x P(x)$.

$\therefore$ Represented → $\dfrac{P(c)}{\forall x P(x)}$ ∀ ↑

P(c): "A Byte Contain 8 bits" so all bytes Contain 8 bits for $\forall x P(x)$

② Universal Instantiation: also Called as universal Elimination (UI) is valid reference rule.

As per UI rule state, that Can refer any sentence P(c) by substituting a ground term c from $\forall x P(x)$

represented as ; $\dfrac{\forall x P(x)}{P(c)}$

Eg: If Every person like ice-cream" ⇒ $\forall x P(x)$

So we infer, "John likes ice-cream" ⇒ P(c)

(3) Existential Instantiation: Also called as Existential Elimination, which is valid Inference rule in FOL

Represented, $\frac{\exists x P(x)}{P(c)}$ for some element c

(4) Existential generalisation Introduction: Also known as Existential Introduction

Represented, $\frac{P(c) \text{ for some element } c}{\exists x P(x)}$

Eg: "Priyanka got good marks in English"

"Therefore, someone got good marks in English"

Generalized MODUS PONENS RULE:

For Inference Proofs in FOL, we have a single Inference rule which is called Generalized modus ponens rule.

Summarized as,

"P Implies Q & P is Inserted to be true, therefore Q must be True"

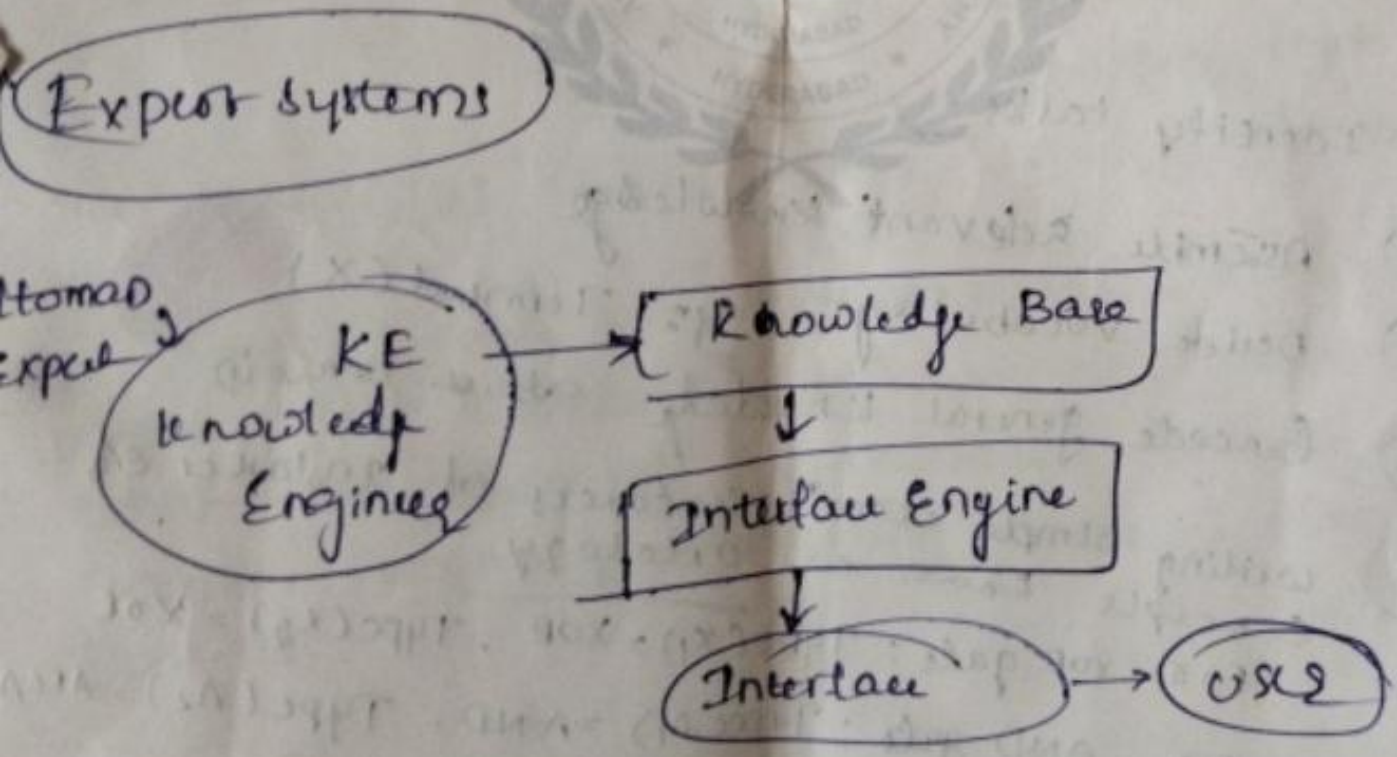
$$\frac{P(x) \Rightarrow Q(x), P(A)}{Q(A)}$$

Knowledge Representation:

Ontological Engineering: Ontological Engineering refers to the representation of abstract ~~Ideas~~ Ideas. These ideas include actions, items, physical objects and Beliefs.

→ Process of ontological Engineering corresponds to process of knowledge engineering.

upper ontology: Refers to the general framework which is established based on the concepts. This is because, depending on general concepts, graphs are constructed at upper level and the highly specific concepts under lower level.



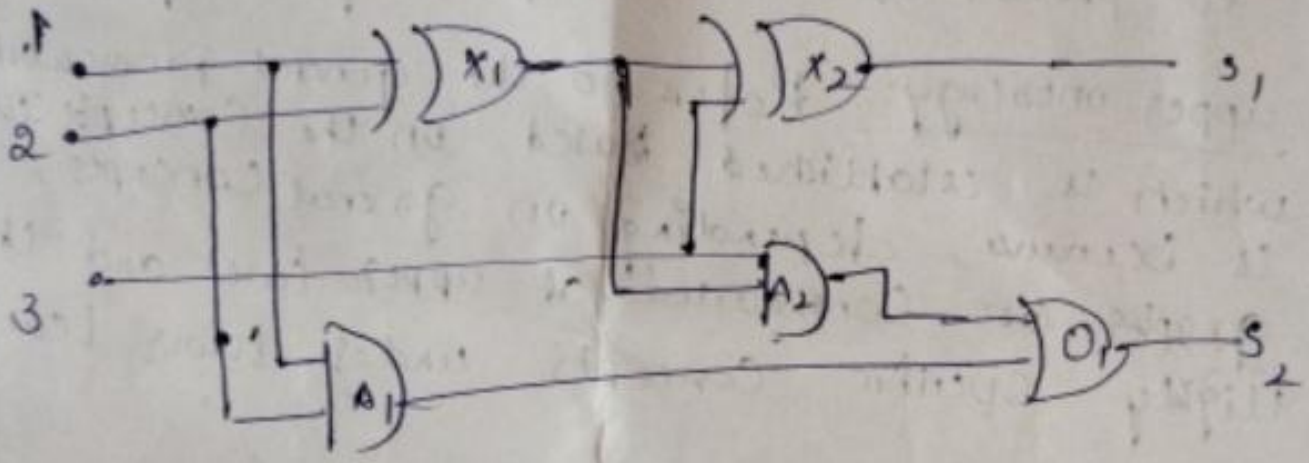
KE: knowledge engineering: process of constructing knowledge Base in FOL, who investigates ~~Concept~~ level Domain, generate representation of ob

Called knowledge Engineer.

→ Knowledge Engineering process is an Electronic ckt Domain, which is already familiar.

Knowledge Engineering process:

One-Bit Full Adder ckt:



① Identify tasks

② Assemble relevant knowledge

③ Decide vocabulary eg: Terminal(X)

④ Encode general knowledge about domain

⑤ writing simple atomic sentences of instances of concepts known as, Ontology.

For XOR gate: TYPE(X_1) = XOR, TYPE(X_2) = XOR

For AND gate: TYPE(A_1) = AND, TYPE(A_2) = AND

For OR gate: TYPE(O_1) = OR

⑥ Debug knowledge Base.

Events:

Event Calculus performs action that are entirely dependent on time Rather than scenarios. The Term 'event Calculus' denotes wider class of actions. It is Implemented inorder to overcome the problems incurred in situation Calculus.

Event Calculus makes use of two relations which is analogous to result relation in situation calculus.

They are: ① Initiates (evt, flt, ti)

This relation specifies that the fluent (flt) is true provided, if the Event is initiated at time ti.

② Terminates (evt, flt, ti)

This relation specifies that fluent (flt) should be true when its terminate

∴ Event Calculus was extended so as to solve Natural issue associated with such as problems of event with duration, simultaneous occuring events, Frequently fluctuating Events & other Barriers.

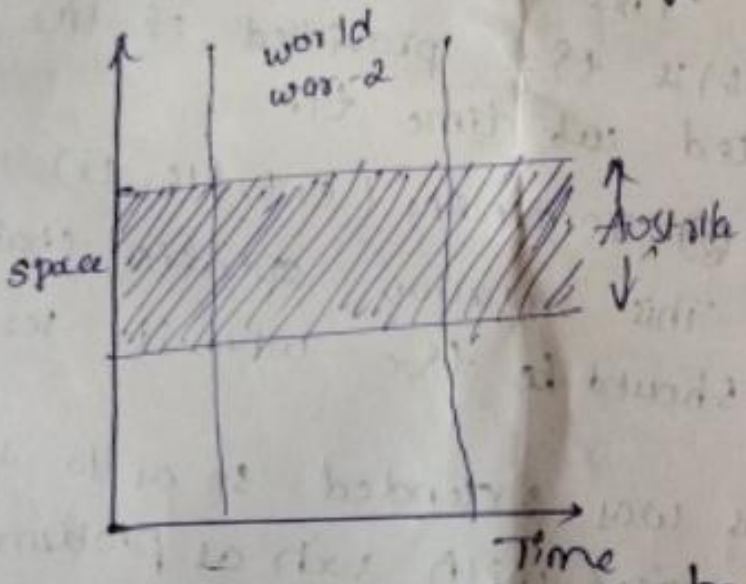
Generalized Events

Generated by considering certain characteristic of 'space-time-chunk'

→ It generalizes Concepts like action, locations times, events & physical objects.

→ A generalized event can also be referred as 'event'

Eg: Considering an event world war-2 occured at different points in space time with Random with time differing borders.



→ Let time Duration of war, be an interval 20th century under fixed limited temporal Extent & maximum spatial Event's location → Australia.

These events are subdivided into various sub events.

→ Subevent (Britain Battle, world war-2)

→ Subevents (world war-2, 20th Century

process Categories / Liquid Events:

- These Events don't have any definite structure.
- that means the process Continously change.

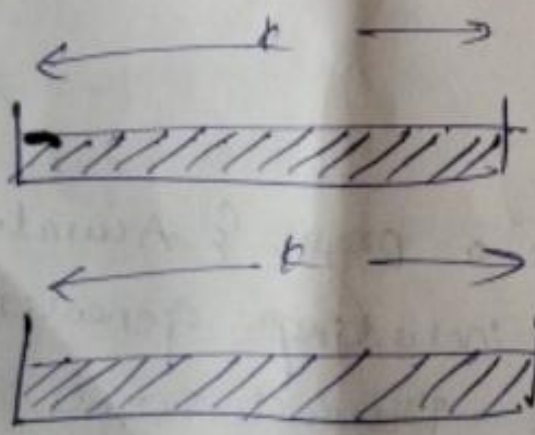
Fluent Calculus:

→ Fluent Calculus generally makes the things real pairs i.e, create a real pairs of Fluents, but it doesn't create pairs of individual Events.

→ The order of representing event of two things occuring simultaneously at same time by a function Both(evt1, evt2).

Eg: A person can perform two tasks simultaneously at same time. He/She can sing & perform dance simultaneously at same time.

$(\forall p \in person) \wedge T(sing(P) \wedge dance(P), t)$



$T(Both(s,d), t)$

∧ is commonicative, associative & similar to Logical Connective.