

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
- Whereas "PL only represent that the world contains facts"

Like : natural language - more than PL  
ex - some human are intelligent or scholar like cricket  
Assomer :-

→ Object :- people, houses, numbers, theories, colour  
Baseball, volley ball, cricket, all games  
war's - 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000

→ Relation :- Red, Round, is adjacent or n-ary  
relation such as :- the sister of, Brother of, has color  
comes b/w.

→ Function :- Father of best friend.  
Third inning of, End inning of, end of  
① son of, ② best friend of  
③ father of



PL is not sufficient to represent the complex sentences (natural language) p. 9 - 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-handled) notation in FOL

## Following the Basic elements:-

1. constant	1, 2, A, John, Rama, Mumbai, ... categories
2. variables	x, y, z, a, b, c, d, ...
3. predicate	Brother, sister, friend, father, ...
4. Function	son, Left Leg of, Best friend of, daughter of
5. connectives	$\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
6. Equality	$=$
7. Quantifier	$\forall, \exists$

Atomic sentences:- It can be represented as, "relations b/w predicate (term 1), term 2, ... term n)

Ex:- Ravi and Ajay are brothers  $\Rightarrow$  Brothers (Ravi, Ajay)  
 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).  
 John owns gold  $\Rightarrow$  owns (John, gold)

### Program:-

classes

Likes (ram, mango).

girl (seema).

red (rose).

### Output:-

queries



Quantification: - It is a language element which generates "Quantification" : Quantification specifies - Quantity of specimen in the universe.

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

→ every Instant of a Particular thing (everything, everywhere, etc)

→  $\forall$  → Inverted A  
 $x \rightarrow$  for all  
 $\Rightarrow \forall x$  } [for all represent]

$\Rightarrow$  every man Respects his parent.

$\Rightarrow$  In this Question, the predicator is "Respect (x,y)", where  $x = \text{man}$  ;  $y = \text{parent}$

$\Rightarrow$  every man. will use  $\forall$  and it will be represented as  
 $\forall x \text{ man}(x) \rightarrow \text{Respect}(x, \text{parent})$

$\Rightarrow$  Existential Quantifier:- which expresses the statement within its scope is true for at least one instance of some things

$\exists$  :- It is a construction symbol.

Ex: Some Boys plays cricket  
"Play (x,y)" :  $x = \text{boys}$  and  $y = \text{game}$   
 some boys so we will use  $\exists$  and it will be

Represented as  $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$

$\Rightarrow$  Ex: Not all student like both Maths and science  
 $x = \text{student}$   $y = \text{subject}$

like (x,y)  $\Rightarrow \neg \forall x \Rightarrow \exists x \neg \forall y (x) [\text{student}(x) \rightarrow \text{like}(x, \text{maths}), y(\text{science})]$

③

using Free variable, it occurs outside the p.g. 16  
when the scope of quantifier

Ex:  $\forall x [A(x) \vee B(y)]$ , here  $x, y$  are bound  
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) -  $\neg$

Table:

P	$\neg P$
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 Disjunction ( $\vee$ ) : Pronunciation : OR

Sentence has  $\vee$  Connection such as  $P \vee Q$  called Disjunction where  $P$  &  $Q$  are propositions.

Eg: Ritika is a doctor **OR** Engineer.

(TT)

P is Ritika is doctor  
Q is Ritika is Engineer  
=  $P \vee Q$

P	Q	$P \vee Q$
T	T	T
F	T	T
T	F	T
F	F	F

### ④ Implication ( $\rightarrow$ )

Pronunciation : Implies

Sentence such as  $P \rightarrow Q$  is Implication. Implication are also known as **if-then** rule.

Eg: If **it is raining**, then the **street is wet**

P

Q

$\therefore P \rightarrow Q$

P	Q	$P \rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

### ⑤ Biconditional ( $\leftrightarrow$ )

Pronunciation : **if and only if**

$P \leftrightarrow Q$  is Biconditional Sentence.

Eg: If **I am breathing**, then **I am alive**

P

Q

$P \leftrightarrow Q$

P	Q	$P \leftrightarrow Q$
T	T	T
T	F	F
F	T	F
F	F	T

the  
symbols



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.

$\psi_1$  &  $\psi_2$  be two atomic sentences &  $\sigma$  be a unifier such that

$$\psi_1 \sigma = \psi_2 \sigma$$

then expressed as,  $\text{UNIFY}(\psi_1, \psi_2)$

Eg:  $\text{Unify}(\text{king}(x), \text{king}(\text{John}))$

Let =  $\psi_1$ ,  $\psi_2$

Substitution -  $\Theta = \{ \text{John}/x \}$  is unifier for these atoms.

### UNIFY ALGORITHM:

- Used for unification which takes two atomic sentences & returns a unifier for those sentences.
- Unification is key component of all first order inference algorithms.
- It returns 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.

$$P(x, y) \rightarrow \textcircled{1}$$

$$P(a, f(z)) \rightarrow \textcircled{2}$$

→ substitute  $x$  with  $a$ ,  $y$  with  $f(z)$  in first expression, represented as  $a/x$ , &  $f(z)/y$

→ with both substitution, first expression will be identical to second expression,

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

### Conditions 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:

~~knows(john, jane)~~

John knows  $x$ , John knows Jane

UNIFY(knows(john,  $x$ ), knows(john, Jane)) =

$\{x/\text{jane}\}$

$\psi_1 = \text{knows}(\text{Richard}, x)$

$\psi_2 = \text{knows}(\text{Richard}, \text{John})$

so  $\Rightarrow \frac{1}{2} \text{ knows}(\text{Richard}, x) ; \text{ knows}(\text{Richard}(\text{John}))$

$\theta = \{ \text{John} / x \}$

Code eg:

unification

①  $\text{parent}(A, B, C)$   
 $\text{parent}(\text{Kevin}, \text{Henry}, 30)$   
                    unify                      } unity succeed  
  not unified

②  $\text{parent}(\text{Kevin}, y, 25)$                       25 not unified  
 $\text{parent}(\text{Kevin}, \text{Henry}, 30)$                       to 25, failed

③ notepad  
women(mia).  
women(fody).

prolog  
?- women(X).  
X = mia

prolog unified women(x) with women(mia)

$\therefore$  instantiating variable x to mia

④ ①  $P(x, a, b)$     ②  $P(y, z, b)$

① check predicates, 'P' same in both expressions

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) (4) (3)

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 & Apply Inference rules (Modus ponens) in forward direction to extract more data until a goal is reached.

FC algorithm 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

$A \rightarrow B$

fact

B

He is running.

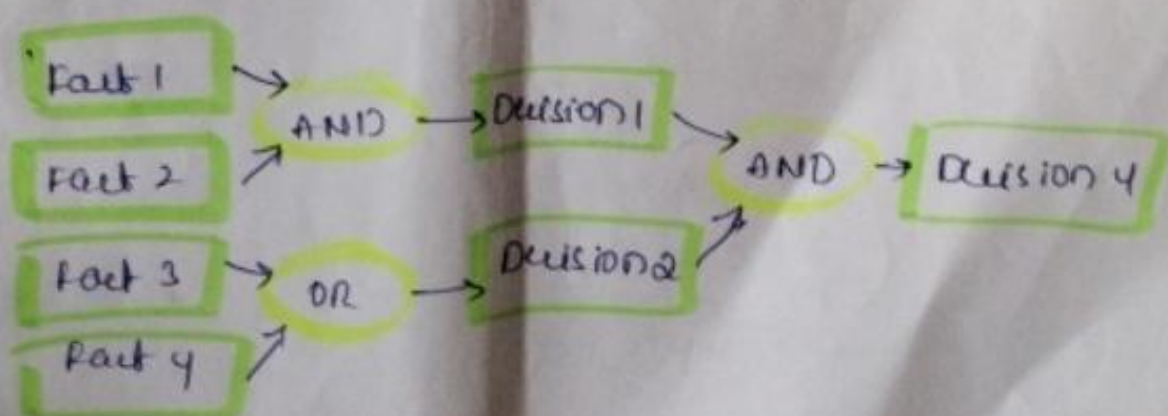
if he is running, he sweats.

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 Goals from generated clauses.
- ③ Make note of what we need to Establish at the end.



## Forward chaining & Backward chaining in AI

(3)

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:

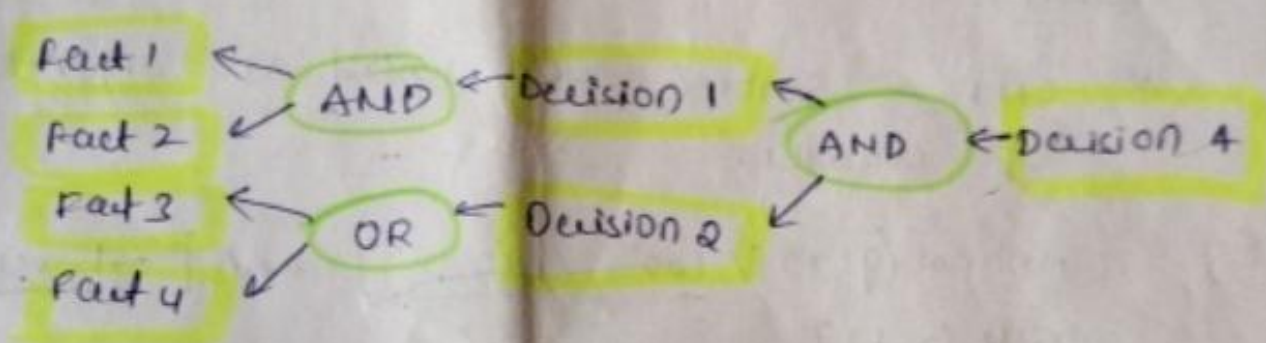
- (a) Forward chaining
- (b) Backward chaining



## Backward chaining (Goal-driven technique) (5)

→ It is a concept in 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 beside in Then part as conclusion.

Eg:  $A \rightarrow B$  Fact  $\#$  B is Endpoint, used as starting point for Backward tracking  
A  $\#$  Initial state

Tom is Sweating (B)

If a person is ~~sweat~~ Running, he will sweat ( $A \rightarrow B$ )

Tom is Running (A)

## Resolution in FOL:

④③

~~There 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 (Conjunctive normal form).

\* CNF ⇒ sentence represented as conjunction of clauses.

Eg:  $[Animal(x) \vee \text{Loves}(x, x)]$  and  $[\neg \text{Loves}(a, b) \vee \neg \text{Kills}(a, b)]$

two complementary literals:  $\text{Loves}(x, x)$  and  $\neg \text{Loves}(a, b)$

two literals unified with unifier  $\theta = [a/x \text{ and } b/x]$

⇒  $[Animal(x) \vee \neg \text{Kills}(x, x)]$ .



eg:

- John likes all kind of food.
- Apple and vegetable are food.
- Anything Anyone eats and not killed is food.
- Anil eats peanuts and still alive.
- Harry eats everything that Anil eats.
- John likes peanuts.

(7)

Statements in FOL

$$\forall x: \text{food}(x) \rightarrow \text{likes}(\text{John}, x)$$

$$\text{food}(\text{Apple}) \wedge \neg \text{food}(\text{vegetable})$$

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

$$\text{eats}(\text{Anil}, \text{peanuts}) \wedge \text{Alive}(\text{Anil})$$

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

$$\forall x: \neg \text{killed}(x) \rightarrow \text{alive}(x)$$

$$\forall x: \text{alive}(x) \rightarrow \neg \text{killed}(x)$$

$$\text{likes}(\text{John}, \text{peanuts})$$

} Added predicates

Conversion of FOL into CNF

\* Eliminate all implications  $\rightarrow$  & also move negation ( $\neg$ ) inwards & rename variables

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

$$\neg \text{food}(\text{Apple}) \wedge \neg \text{food}(\text{vegetables})$$

$$\forall x. \forall y \rightarrow \text{eats}(x, y) \vee \text{killed}(x) \vee \neg \text{food}(y)$$

$$\text{eats}(\text{Anil}, \text{peanuts}) \wedge \text{Alive}(\text{Anil})$$

$$\forall x \rightarrow \text{eats}(\text{Anil}, x) \vee \text{eats}(\text{Harry}, x)$$

$$\forall x \rightarrow \text{killed}(x) \vee \text{alive}(x)$$

$$\forall x \rightarrow \text{alive}(x) \vee \neg \text{killed}(x)$$

$$\text{likes}(\text{John}, \text{peanuts})$$

→ Rename variables,

$\forall x \rightarrow \text{food}(x) \vee \text{likes}(\text{John}, x)$   
 $\neg \text{food}(\text{Apple}) \wedge \neg \text{food}(\text{vegetables})$   
 $\forall y \forall z \rightarrow \text{eats}(y, z) \vee \text{killed}(y) \vee \neg \text{food}(z)$   
 $\text{eats}(\text{Anil}, \text{peanuts}) \wedge \text{alive}(\text{Anil})$   
 $\forall w \rightarrow \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Harry}, w)$   
 $\forall g \rightarrow \text{killed}(g) \vee \text{alive}(g)$   
 $\forall k \rightarrow \text{alive}(k) \vee \neg \text{killed}(k)$   
 $\text{likes}(\text{John}, \text{Peanuts})$

→ Eliminate existential instantiation quantifier by Elimination.

( $\exists$ )  
\* Here, no existential quantifier so all statements will remain same.

→ Drop universal quantifiers ( $\forall$ )

- a.  $\neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
- b.  $\neg \text{food}(\text{Apple})$
- c.  $\neg \text{food}(\text{vegetables})$
- d.  $\neg \text{eats}(y, z) \vee \text{killed}(y) \vee \underline{\neg \text{food}(z)}$
- e.  $\text{eats}(\text{Anil}, \text{Peanuts})$
- f.  $\text{alive}(\text{Anil})$
- g.  $\neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Harry}, w)$
- h.  $\text{killed}(g) \vee \text{alive}(g)$
- i.  $\neg \text{alive}(k) \vee \neg \text{killed}(k)$
- j.  $\text{likes}(\text{John}, \text{Peanuts})$



John likes peanuts  
John likes anything

- ① likes (John, peanuts)  
likes (John, x)

unify:  $\boxed{x = \text{peanuts}}$

- ② Resolution: FOL  $\rightarrow$  CNF <sup>Conjunctive normal form</sup>

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

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

left,  
food (peanuts)

\* unification is key concept of resolution

## FOL Inference Rules for Quantifiers

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

### Inference Basic rules in FOL:-

- ① universal Generalization
- ② universal Instantiation
- ③ Existential Instantiation
- ④ Existential Introduction

① universal Generalization: Is a valid Inference rule which states if premise  $P(c)$  is true for any arbitrary element  $c$  in universe of discourse,  
 $\therefore \forall x P(x)$ .

$\therefore$  Represented  $\rightarrow \frac{P(c)}{\forall x P(x)}$   $\forall$   
 $\uparrow$

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

② universal Instantiation: also called as universal Elimination (UI) is a valid inference rule.

As per UI rule states, that can infer any sentence  $P(c)$  by substituting a ground term  $c$  from  $\forall x P(x)$

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

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

So we infer, "John likes ice-cream"  $\Rightarrow P(c)$



③ Existential Instantiation, also called as Existential Elimination, which is a valid Inference rule in FOL.  
 Represented,  $\frac{\exists x P(x)}{P(c)}$  for some element  $c$

④ Existential Introduction, also known as Existential Generalisation.  
~~Existential Introduction~~  
 Represented,  $\frac{P(c)}{\exists x P(x)}$  for some element  $c$ .

Eg: "Priyanka got good marks in English"  
 "Therefore, someone got good marks in English."

### Generalised MODUS PONENS RULE:

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

Summarized as,

"P implies Q & P is asserted to be true, therefore Q must be True"

$$P(x) \Rightarrow Q(x), P(A)$$

---


$$Q(A)$$

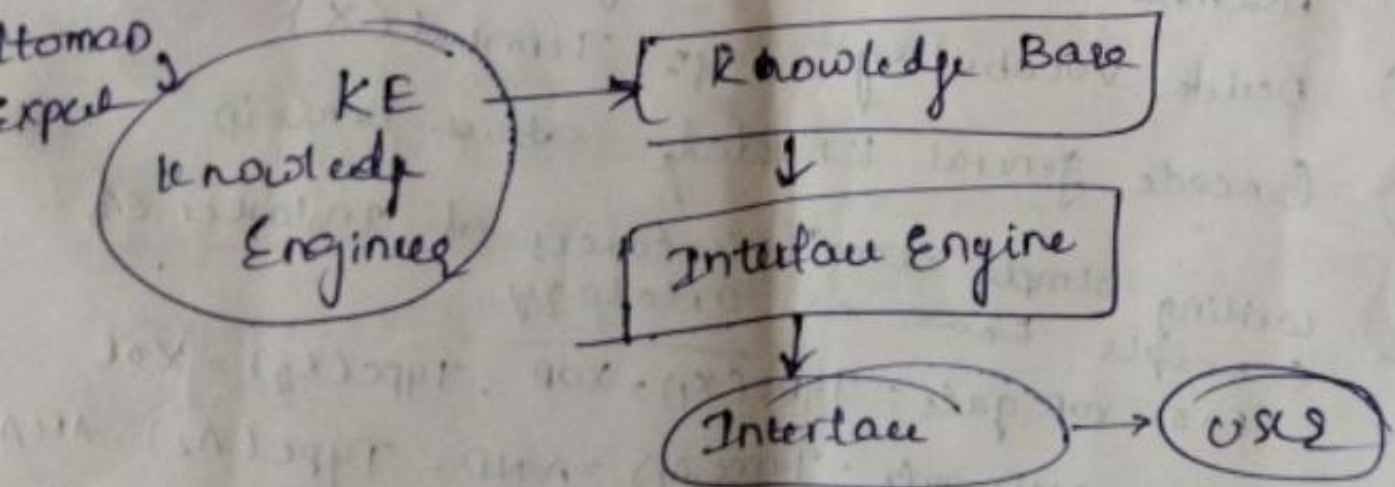
## Knowledge Representation:

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.

## Expert systems



KE: Knowledge Engineering: process of constructing

→ knowledge Base in FOL, who investigates ~~current~~ domain, generates representation of it



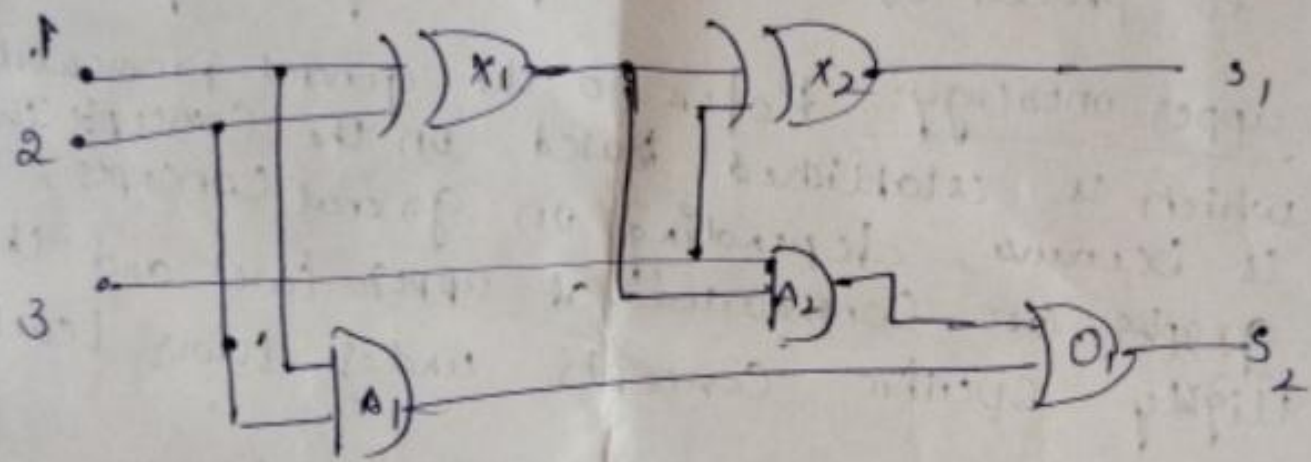
Called Knowledge Engineer.

(2)

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

Knowledge Engineering process;

One-Bit Full Adder ckt:



① Identity tasks

② Assemble relevant knowledge

③ Decide vocabulary e.g. Terminal(X)

④ Encode general knowledge about domain

⑤ writing simple atomic sentences of instance of concepts known as, ontology.

For XOR gate:  $Type(X1) = XOR$ ,  $Type(X2) = XOR$

For AND gate:  $Type(A1) = AND$ ,  $Type(A2) = AND$

For OR gate:  $Type(O1) = OR$

⑥ Debug knowledge Base.

## Events:

Event Calculus performs actions that are entirely dependent on time rather than scenarios. The term 'event calculus' denotes wider class of actions. It is implemented in model 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  $\text{Initiates}(\text{evt}, \text{flt}, t_i)$   
This relation specifies that the fluent ( $\text{flt}$ ) is true provided, if the event is initiated at time  $t_i$ .

- ②  $\text{Terminates}(\text{evt}, \text{flt}, t_i)$

This relation specifies that the fluent ( $\text{flt}$ ) should be true when its terminate.

∴ Event Calculus was extended so as to solve various issue associated with such as problems of event with duration, simultaneous occurring events, frequently fluctuating events, & other barriers.



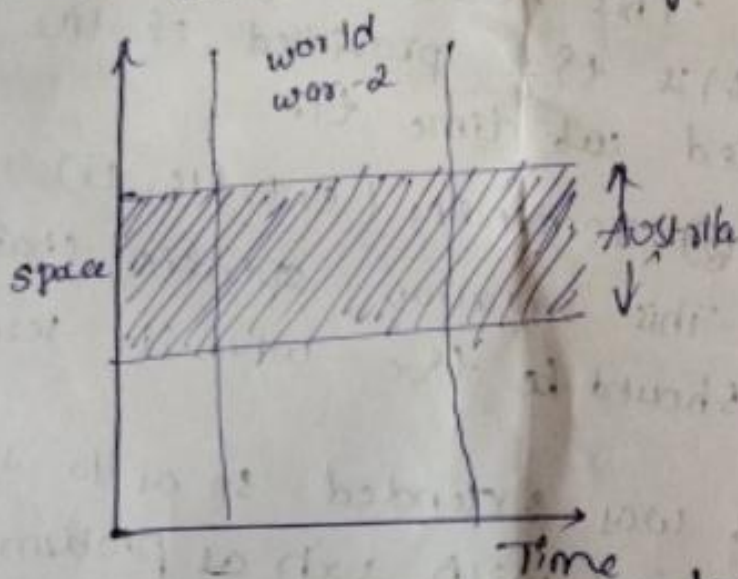
## Generalized Events

Generated by considering certain characteristics of 'space-time-chunk'.

→ It generalizes concepts like actions, locations, times, places & physical objects.

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

eg: Considering an event world war-2 occurred 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, location → Australia.

These events are subdivided into various sub events.

→ Subevent (Britain Battle, world war-2)

→ Subevent (world war-2, 20th Century)

## process categories / Liquid Events:

(6)

These Events don't have any definite structure.  
that means the processes continuously change.

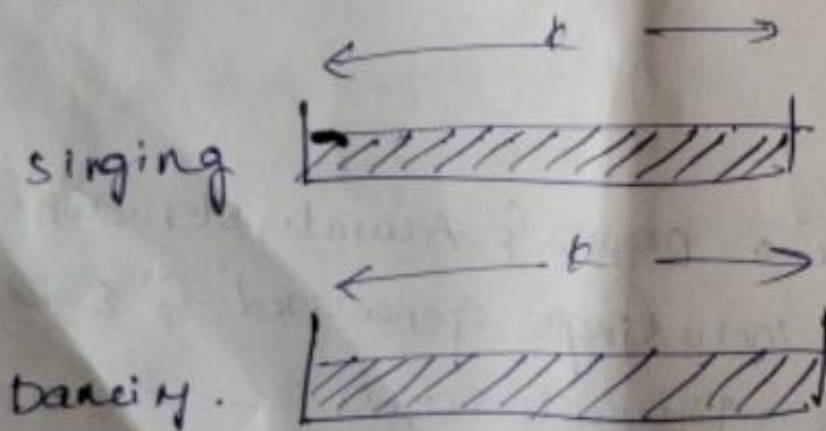
## Fluent Calculus:

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

→ The order of representing event of two things  
occurring simultaneously at same time by a  
function  $\text{Both}(e_1, e_2)$ .

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

$(\exists \text{person}) \wedge T(\text{sing}(p) \wedge \text{dance}(p), t)$



$T(\text{Both}(s, d), t)$

① is commutative, associative & similar  
to logical connective.