## First - Order Logic:

FOL / PC

↳ Predicate Calculus

* More expressive
* More succinct

} than propositional logic.

Higher complexity!

eg: $\left[\forall r \ \underbrace{Pit(r)}_{\text{Property}} \Rightarrow [\forall s \ \underbrace{Adjacent(r, s)}_{\text{Relation}} \Rightarrow Breezy(s)]\right]$

↳ Quantification

## World:

In propositional logic, world is some assignment of all variables.

Here it is more complicated.

* Objects → eg: people, houses, wumpus, numbers, colors ...
* Properties → breezy, large
* Relations → inside, adjacent
* Functions → fatherOf, bestfriend

eg: one plus one equals two

objects : one, two

retations : equals

functions : plus

properties : —

* squares adjacent to the wumpus are smelly

o : squares, wumpus

r : adjacent

p : smelly

f : ~

Syntax:

* Constants : 2, Jack, UCLA [Objects]
* Predicates : adjacent [Relations, properties]
* Functions : left of [Functions]
* Variables : $x, y, z, \ldots$ [Objects]
* Connectors : $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$

* Equality : $=$ (type of predicates)

* Quantifiers : $\forall$, $\exists$

for all there exists

## ATOMIC SENTENCES:

TERM : Costant or variable or function $(term_1, \ldots, term_n)$

ATOMIC SENTENCE:

Predicate $(term_1, \ldots, term_n)$

eg: Brother (Jack, Tom)

Brother (Father of (Jack), Tom)

## SYNTAX:

$S, S_1, S_2$ : sentences

$\neg S$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_2$

* $>(1,2) \vee \leq(1,2)$

* $>(1,2) \wedge \neg(>(1,2))$

* [ >(age(sally), age(layla)) $\vee$ ... ] $\Rightarrow$ ...

function : maps objects to objects

eg: age(sally)

relation: holds between objects

$>(\cdot, \cdot)$

returns true / false.

property : Applies to only one object. Returns T/F.

UNIVERSAL QUANTIFICATION $\forall$ for all :

$\forall$ <variables> <sentence>

$\forall x \quad At(x, UCLA) \Rightarrow Smart(x)$

Relation

Predicates : At

Smart

Property

Constant : UCLA

equivalent to the conjunction of all instantiations of P

$[At(John, UCLA) \Rightarrow Smart(John)] \wedge$

$[At(Ed, UCLA) \Rightarrow Smart(Ed)] \wedge$

$[At(Fatherof(Ed), UCLA) \Rightarrow Smart(Fatherof(Ed))] \wedge \ldots$

## EXISTENTIAL QUANTIFICATION $\exists$

there exists

$\exists$ <variable> <sentence>

$\exists x \; At(x, UCLA) \wedge Tall(x)$

DISJUNCTION

$[At(Ed, UCLA) \wedge Tall(Ed)] \vee$

$[At(Sandy, UCLA) \wedge Tall(Sandy)] \vee$

$\vdots$

note:

$\forall$ has $\Rightarrow$

$\exists$ has $\wedge$.

## PROPERTIES OF QUANTIFIERS:

* $\forall x \forall y$ same as $\forall y \forall x$
* $\exists x \exists y$ same as $\exists y \exists x$
* $\exists x \forall y$ is not the same as $\forall y \exists x$

eg: $\forall y \exists x \; Loves(x, y)$ : Everyone in the world is loved by at least one person.

$\exists x \forall y$ Loves $(x, y)$: There is a person who loves everyone in the world.

- $\forall x$ likes $(x,$ Ice Cream$)$

$\neg \exists x \neg$likes$(x,$ Ice, Cream$)$

$\neg(\alpha \wedge \beta) \rightarrow \neg\alpha \vee \neg\beta$

$\neg(\alpha \vee \beta) \rightarrow \neg\alpha \wedge \neg\beta$

$\neg \exists x \neg$likes$(x,$ Ice cream$) \rightarrow \forall x \neg\neg$likes$(x,$ Ice Cream$)$

$\forall x$ likes $(x,$ Ice Cream$)$.

EQUALITY:

1. Spot has two sisters

Predicate: sister $(\_ , \_)$

Constants: Spot

$\exists x \exists y$ sister$(x,$ spot$) \wedge$ sister$(y,$ spot$) \wedge \neg(x = y)$

2. Spot has exactly two sisters

$\exists x\, \exists y\ sister(x, spot) \wedge sister(y, spot) \wedge \neg(x = y) \wedge$

$[\forall z\ \ sister(z, spot) \Rightarrow (z = x \ \vee\ z = y)]$

$[\neg(\exists z\ sister(z, spot)\ \neg(z = x)\ \vee\ \neg(z = y))]$

$\exists!$ UNIQUENESS QUANTIFIER:

$\exists!\ x\ \ king(x)$

there is exactly one king.

$\exists x\ \ king(x) \wedge [\forall y\ \ king(y) \Rightarrow (x = y)]$

Is this okay?

$\forall x\ [cat(x) \vee (\exists x\ \ Brother(Rich, x))]$

its fine but not advised.

Avoid having free variables → variables with no quantifiers.

WELL-FORMED FORMULA: expression with no free variables.

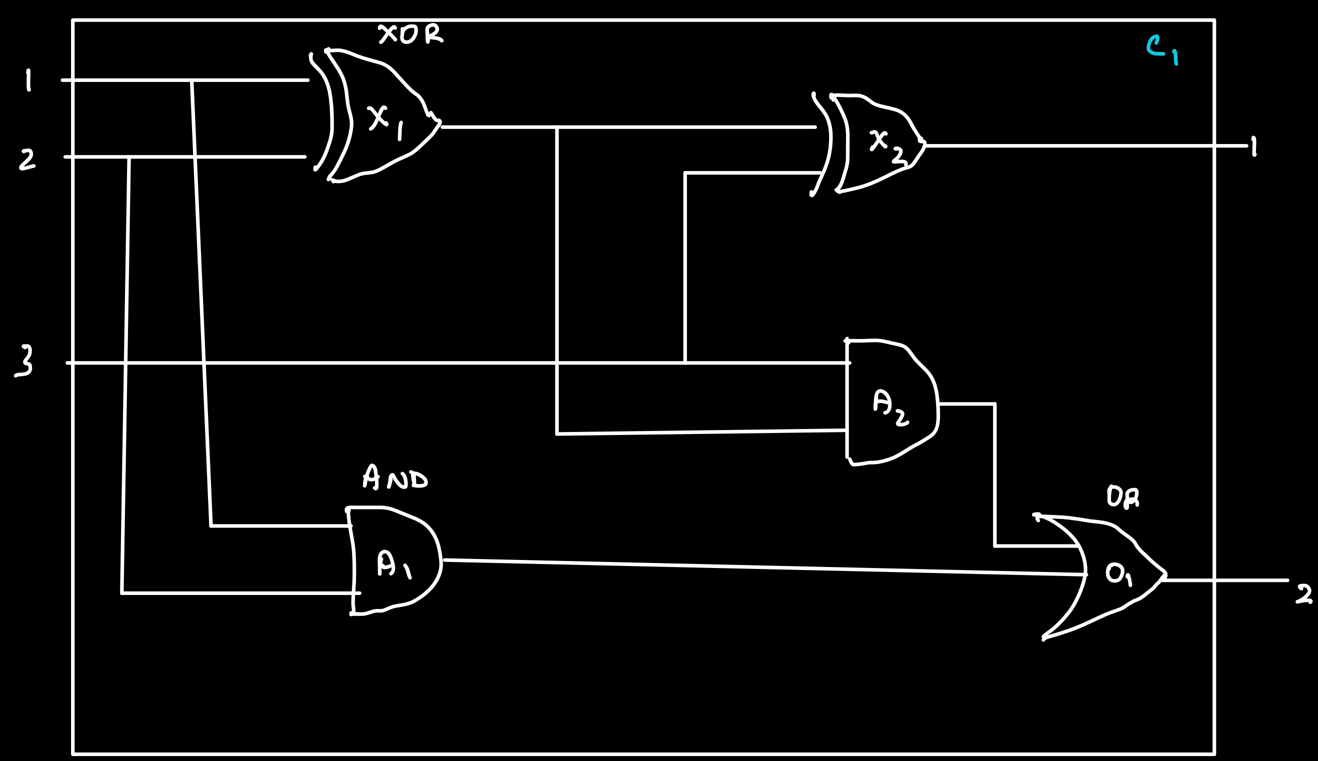
KB: $\begin{cases} \forall x \quad cat(x) \Rightarrow mammal(x) \\ cat(tuna) \\ cat(spot) \end{cases}$

Query : $\exists x \quad mammal(x)$

Returns $\{x / spot, \; x / Tuna\}$ not just T/F like in $\Delta \nvDash \alpha$ or $\Delta \vDash \alpha$.

EXAMPLE:

one-bit full adder



1 Vocabulary

constants, functions, predicates.

KB → 2 independent of this circuit (domain)

→ 3 this circuit (instance)

Queries : say output is 01, what inputs give this output.

$\exists i_1, \exists i_2, \exists i_3 \dots$

1. Vocabulary:

CONSTANTS:

$\left.\begin{array}{l}\text{AND, OR, NOT, XOR} \\ 0, 1\end{array}\right\}$ domain

$\left.X_1, X_2, A_1, A_2, O_1\right\}$ instance

FUNCTIONS:

Type, Signal, In, Out

PREDICATE:

Connected.

2. Domain

* $\forall t_1, t_2 \quad connected(t_1, t_2) \Rightarrow signal(t_1) = Signal(t_2)$

* $\forall t_1, t_2 \quad connected(t_1, t_2) \Leftrightarrow connected(t_2, t_1)$

  ⋮

* $\forall g \quad type(g) = OR \Rightarrow [signal(out(1, g)) = 1 \Leftrightarrow$

  ⋮ $\qquad \exists n \;\; Signal(In(n, g)) = 1]$

  (AND, XOR, NOT)

  ⋮

* $\forall t \quad signal(t) = 1 \lor signal(t) = 0$

* $\neg(1 = 0)$

3. Instance

$Type(x_1) = XOR$

$Type(x_2) = XOR$

⋮

$Type(O_1) = OR$

$Connected(Out(1, x_1), In(2, A_2))$

⋮

QUERY :

$$\exists i_1, i_2\, i_3 \quad \begin{aligned} &Signal(In(1, C_1)) = i_1 \;\wedge \\ &Signal(In(2, C_1)) = i_2 \;\wedge \\ &Signal(In(3, C_1)) = i_3 \;\wedge \\ &Signal(Out(1, C_1)) = 0 \;\wedge \\ &Signal(Out(2, C_1)) = 1 . \end{aligned}$$

Output :

Yes

| $i_1$ | $i_2$ | $i_3$ |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 . |