AFLL

UNIT - 5

CLASS NOTES

feedback/corrections: vibha@pesu.pes.edu

Vibha Masti

Programming Language

- Declarative Language
  - logic programming
  - prolog, lisp, SQL
- Procedural Language
  - imperative programming
  - C, C++, Java, Python, C#

Prolog

- Basic concepts of logic (less powerful)
- First Order Predicate Logic (FOPL) (more powerful)
- Prolog : Programming Logic
- Based on principles of FOPL

Prolog Basic Constructs

ends in fullstop

comma is AND

semicolon is OR

knowledge base	fact	A.	
knowledge base	rule	A:- B_1, B_2, B_3, ... B_n.	if B_1, B_2, ... B_n then A or $B_1 \land B_2 \land \dots \land B_n \rightarrow A$
knowledge base	rule	A :- B_1; B_2; B_3; ... B_n.	if B_1 or B_2 or ... or B_n then A or $B_1 \lor B_2 \lor \dots \lor B_n \rightarrow A$
	query	?- B_1, B_2, ... B_n	

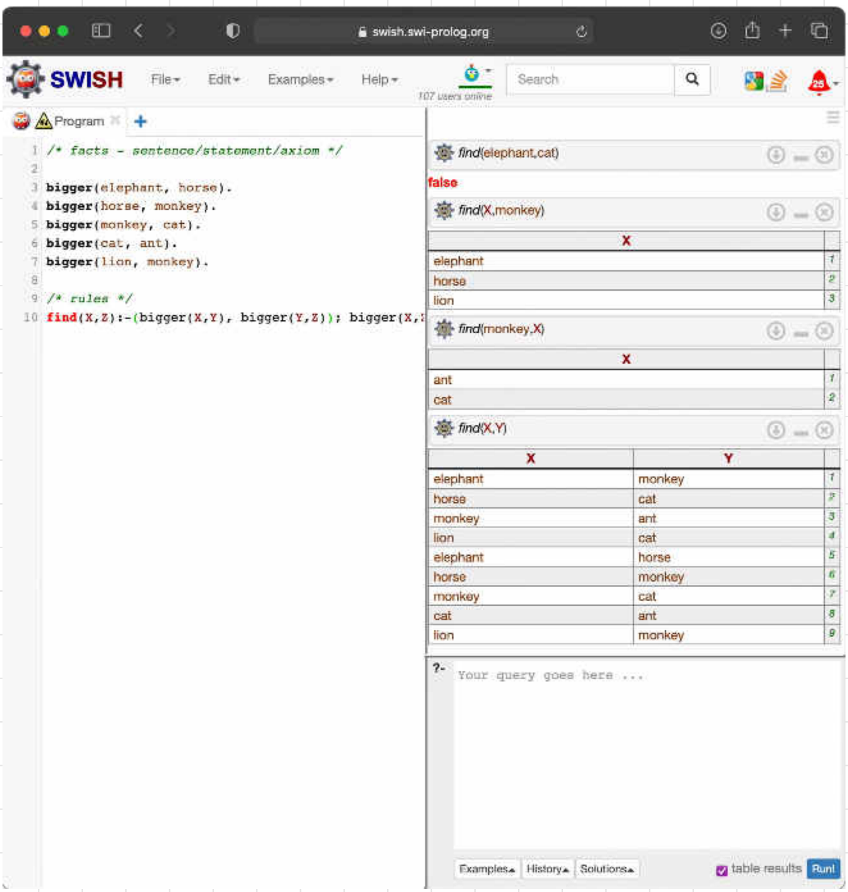
Example – simple logical program

animal size

```
/* facts - sentence/statement/axiom */

bigger(elephant, horse).
bigger(horse, monkey).
bigger(monkey, cat).
bigger(cat, ant).
bigger(lion, monkey).

/* rules */
find(X,Z):-(bigger(X,Y), bigger(Y,Z)); bigger(X,Z).
```

- run on https://swish.swi-prolog.org

Queries

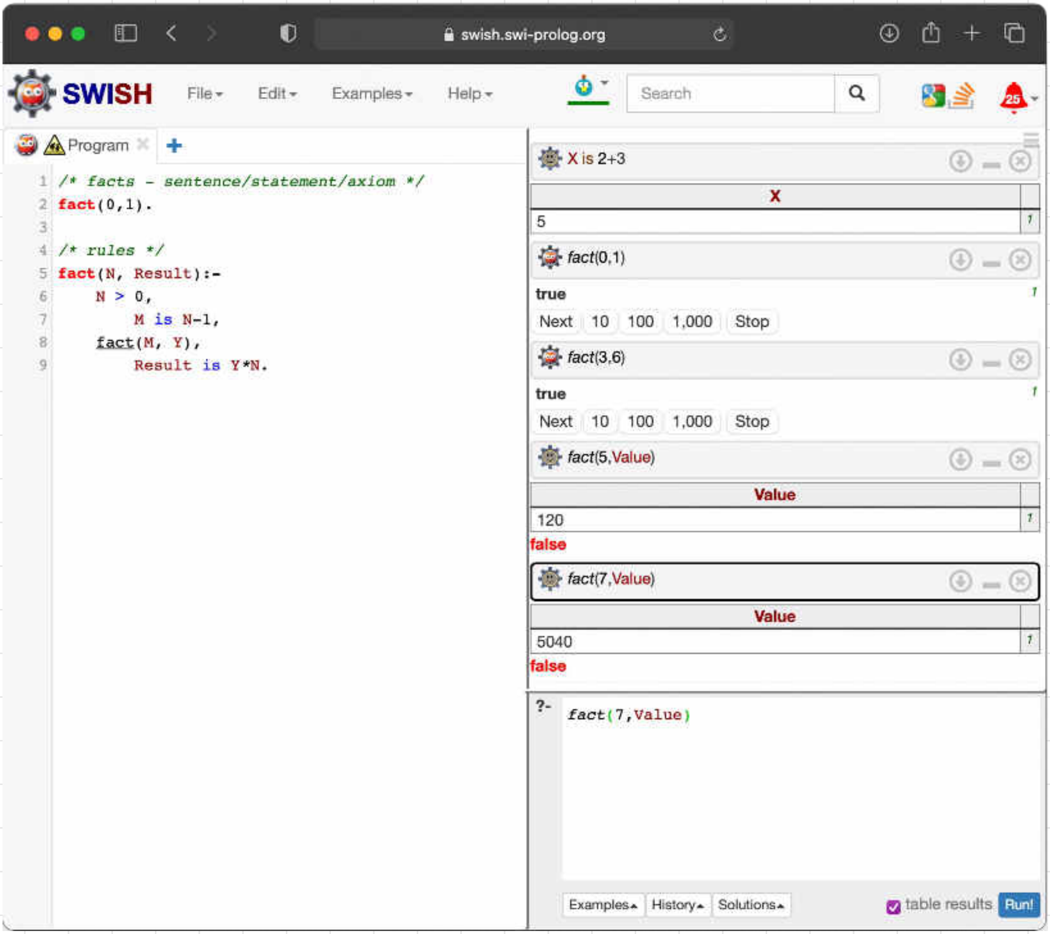


Factorial Program

```
fact(0,1).
fact(N, Result):-
    N > 0,
        M is N-1,
    fact(M, Y),
        Result is Y*N.
```

- run on https://swish.swi-prolog.org

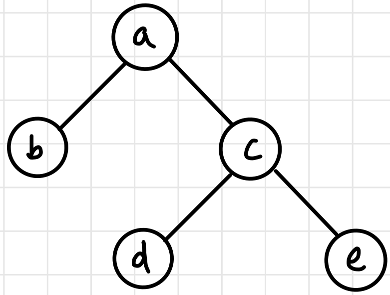
Queries



Binary Tree

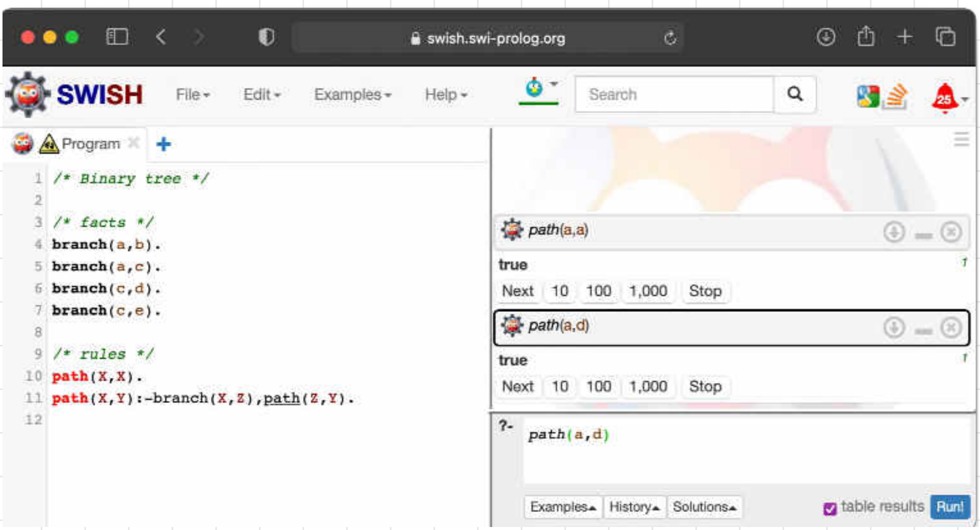
```
/* Binary tree */

/* facts */
branch(a,b).
branch(a,c).
branch(c,d).
branch(c,e).

/* rules */
path(X,X).
path(X,Y):-branch(X,Z),path(Z,Y).
```

recursive call



- run on https://swish.swi-prolog.org

Queries



Prolog Interpreter

- examines rules in the order in which they have been specified in the database

- database created — Knowledge Base : list of facts (predicates) and rules about the domain

Logic

- syntax : design (legal expressions)
- semantics: working (truth values of legal expressions)
- proof system: way of manipulating syntactic expressions to get others

Propositional Logic

Proposition

- declarative sentence , either true or false
- 'Joe is happy'
- merge atomic sentences to form complex ones

Question 1

Proposition or not?	proposition	value
1) Bangalore is the capital of India	yes	false
2) New Delhi is the capital of India	yes	true
3) Do you like coffee?	no	—
4) Get me a cup of coffee	no	—
5) $1+2=3$	yes	true
6) $2+3=4$	yes	false
7) $x+3=10$	no	—
8) $x+y=z$	no	—

Compound Propositions / Complex Sentence

- Propositions formed from existing propositions using logical connectives/ operators

- Logical operators
  * Negation ($\neg$) (not)
  * Conjunction ($\wedge$) (and)
  * Disjunction ($\vee$) (or)
  * Conditional statement ($\rightarrow$) (implies)
  * Biconditional statement ($\leftrightarrow$)

Truth Table

hypothesis/ premise/ claim

p	q	conjunction: $p \wedge q$ and	disjunction: $p \vee q$ or	if p then q: $p \rightarrow q$ implies	q iff p: $p \leftrightarrow q$ biconditional
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

q whenever p

Question 2

Let p, q, r be the propositions

p: You have the flu
q: You miss the final examination
r: You pass the course

Express the propositions as English sentences

- $p \rightarrow q$

  if you have the flu, then you miss the final examination

- $\neg q \leftrightarrow r$

  you pass the course if and only if you do not miss the final examination

- $q \rightarrow \neg r$

  if you miss the final examination, you do not pass the course

- $(p \rightarrow \neg r) \vee (q \rightarrow \neg r)$

  if you have the flue then you do not pass the course or if you miss the final examination then you do not pass the course

- $(p \wedge q) \vee (\neg q \wedge r)$

  you have the flu and you miss the final examination or you do not miss the final examination and you pass the course

Question 3

Let p and q be propositions

p: You drive over 60 kmph
q: You get a speeding ticket

Write the following propositions using p, q and logical connectives

(a) You do not drive over 60 kmph
$\neg p$

(b) You drive over 60 kmph, but you do not get a speeding ticket
$p \wedge \neg q$

(c) You will get a speeding ticket if you drive over 60 kmph
$p \rightarrow q$

(d) If you do not drive over 60 kmph, then you will not get a speeding ticket
$\neg p \rightarrow \neg q$

(e) You get a speeding ticket but you do not drive over 60 kmph
$p \wedge \neg q$

Question 4

Determine the truth values of the following propositions

a. $2+2=4$ iff $1+1=2$ true (both true, $\leftrightarrow$)
b. $1+1=2$ iff $2+3=4$ false
c. $0>1$ iff $2>1$ false
d. if $1+1=2$ then $2+2=5$ false (p is true, q is false)
e. if $1+1=3$ then $2+2=4$ true (p is false, q is X)
f. if $1+1=3$ then $2+2=5$ true
g. if $2+2=4$ then $1+2=3$ true
h. $1+1=3$ iff monkeys can fly true
i. if monkeys can fly then $1+1=3$ true
j. if $1+1=3$ then unicorns exist true
k. if $1+1=3$ then dogs can fly true
l. if $1+1=2$ then dogs can fly false

Syntax & Semantics of PL

syntax

ambiguous grammar

Sentence $\longrightarrow$ AtomicSentence | ComplexSentence
AtomicSentence $\longrightarrow$ True | False | P | Q | R | . . .
ComplexSentence $\longrightarrow$ (Sentence) | [Sentence] | $\neg$ Sentence
| Sentence $\wedge$ Sentence
| Sentence $\vee$ Sentence
| Sentence $\Rightarrow$ Sentence
| Sentence $\Leftrightarrow$ Sentence

Operator Precedence : $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

takes care of ambiguity

semantics

Based on truth value of the sentences. Meaning of the sentence is a function of the meaning of its parts.

Tautology

- Compound proposition that is always true no matter what the truth values of the propositions that occur in it
- Example : $(p \rightarrow q) \wedge (q \rightarrow p) \leftrightarrow (p \leftrightarrow q)$

Contradiction

- Compound proposition that is always false no matter what the truth values of the propositions that occur in it
- Example: $\neg(p \to q) \wedge \neg(p \wedge \neg q)$

$\underbrace{\neg(p \to q)}_{10}$; $\neg(p \wedge \neg q)$: 00, 11

Logical Equivalence

- Compound propositions p & q are logically equivalent if $p \leftrightarrow q$ is a tautology
- The notation $p \equiv q$ denotes that p and q are logically equivalent
- Compound propositions that have the same truth values in all possible cases are called logically equivalent
- Example: $(p \to q) \wedge (q \to p) \equiv (p \leftrightarrow q)$
- The symbol '≡' is not a logical connective. Hence $p \equiv q$ is not a compound proposition and it simply means $p \leftrightarrow q$ is a tautology.

Question 5

Show that $p \to q \equiv (\neg p \vee q)$

p	q	$p \to q$	$\neg p$	$\neg p \vee q$	$(p \to q) \leftrightarrow (\neg p \vee q)$
0	0	1	1	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	1	1	0	1	1

Laws of Logic

Equivalence	Name of Identity
$p \wedge T \equiv p$ $p \vee F \equiv p$	Identity Laws
$p \wedge F \equiv F$ $p \vee T \equiv T$	Domination Laws
$p \wedge p \equiv p$ $p \vee p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative Laws
$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ $(p \vee q) \vee r \equiv p \vee (q \vee r)$	Associative Laws
$p \vee (q \wedge r) \equiv (p \wedge q) \vee (p \wedge r)$ $p \wedge (q \vee r) \equiv (p \vee q) \wedge (p \vee r)$	Distribution Laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	DeMorgan's Laws
$p \wedge (p \vee q) \equiv p$ $p \vee (p \wedge q) \equiv p$	Absorption Laws
$p \wedge \neg p \equiv F$ $p \vee \neg p \equiv T$	Negation Laws

Question 6

Prove using laws of logic

1. $\neg(p \rightarrow q) \equiv p \wedge \neg q$

$p \rightarrow q \equiv \neg p \vee q$ * important

$\neg(p \rightarrow q) \equiv \neg(\neg p \vee q)$

$\neg(p \rightarrow q) \equiv p \wedge \neg q$ De Morgan's

2. $p \wedge (p \to q) \equiv p \wedge q$

$p \wedge (\neg p \vee q) \equiv p \wedge q$

$(p \wedge \neg p) \vee p \wedge q \equiv p \wedge q$

$p \wedge q \equiv p \wedge q$

3. $\neg (p \vee (\neg p \wedge q)) \equiv \neg (p \vee q)$

$\neg ((p \vee \neg p) \wedge (p \vee q)) \equiv \neg (p \vee q)$

$\neg (p \vee q) \equiv \neg (p \vee q)$

Question 7

Prove the following expressions are tautologies

1. $(p \wedge q) \to (p \vee q)$

$((p \wedge q) \to (p \vee q)) \longleftrightarrow T$

$\neg (p \wedge q) \vee (p \vee q) \equiv T$

$\neg p \vee \neg q \vee p \vee q \equiv T$

$(\neg p \vee p) \vee (\neg q \vee q) \equiv T$

$T \vee T \equiv T$

$T \equiv T$

2. $(p \wedge q) \to (p \to q)$

$(p \wedge q) \to (p \to q) \equiv T$

$(p \wedge q) \to (\neg p \vee q) \equiv T$

$\neg (p \wedge q) \vee (\neg p \vee q) \equiv T$

$\neg p \vee \neg q \vee \neg p \vee q \equiv T$

$\neg q \vee q \vee \neg p \equiv T$

$T \vee \neg p \equiv T$

$T \equiv T$

Logical Entailment

- Describes the relationship between statements that hold true when one statement logically follows from one or more statements

- $\alpha \models \beta$

- premises conclusion
  $p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q$ is called a logical entailment or a valid argument.

- It must be a tautology

- The sentence q logically follows from $(p_1 \wedge p_2 \wedge \ldots \wedge p_n)$ which is another sentence

- Conclusion is entailed by the premises

Mathematically

$$\alpha \equiv \beta \text{ iff } \alpha \models \beta \text{ and } \beta \models \alpha$$

- Model Checking (truth table)

- Rules of Inference (Theorem Proving)

Rules of Inference

- Well-known argument forms to simplify a complex argument form at hand

- Well-known argument forms called Rules of Inference

Rules

no need to memorise (geeksforgeeks)

also imp; derived

Rule of Inference	Tautology	Name
p $p \rightarrow q$ $\therefore q$	$(p \wedge (p \rightarrow q)) \rightarrow q$	Modus Ponens
$\neg q$ $p \rightarrow q$ $\therefore \neg p$	$(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$	Modus Tollens
$p \rightarrow q$ $q \rightarrow r$ $\therefore p \rightarrow r$	$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$	Hypothetical syllogism
$\neg p$ $p \vee q$ $\therefore q$	$(\neg p \wedge (p \vee q)) \rightarrow q$	Disjunctive Syllogism
p $\therefore (p \vee q)$	$p \rightarrow (p \vee q)$	Addition
$(p \wedge q) \rightarrow r$ $\therefore p \rightarrow (q \rightarrow r)$	$((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$	Exportation
$p \vee q$ $\neg p \vee r$ $\therefore q \vee r$	$((p \vee q) \wedge (\neg p \vee r)) \rightarrow q \vee r$	Magical rule Resolution

only imp

can derive all others

Proof by Resolution

- Most modern automated rule provers use proof by resolution technique
- It is an $\wedge$ of $\vee$s
- Each premise in the form of disjunction (should be in CNF or Conjunctive Normal Form)

$$p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q$$

p_i is a disjunction ($\vee$)

Question 8

ors

$(p \vee q) \wedge (\neg p \vee r) \leftrightarrow (q \vee r)$ is a tautology

and

nots get cancelled

$$\begin{array}{c} \cancel{p} \vee q \\ \cancel{\neg p} \vee r \\ \hline q \vee r \end{array}$$

Question 9

$p \wedge (\neg p \vee r) \leftrightarrow r$ is a tautology

nots get cancelled

$$\begin{array}{c} \cancel{p} \\ \cancel{\neg p} \vee r \\ \hline r \end{array}$$

Conjunctive Normal Form (CNF)

- Open up implications to get ORs ($p \rightarrow q \leftrightarrow \neg p \vee q$)
- Get rid of double negatives
- Convert $A \vee (B \wedge C)$ to $(A \vee B) \wedge (A \vee C)$ (distributivity)

Question 10

Convert $A \rightarrow (B \wedge C)$ to CNF

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

Resolution

$$\frac{\begin{array}{c}\neg A \lor B \\ \neg A \lor C\end{array}}{\neg A \lor B \lor C}$$

Modus Ponens and Modus Tollens

- Special cases of Resolution Rule

* Modus Ponens

$$\left.\begin{array}{l} p \to q \\ p \end{array}\right] \text{ if } p \to q \text{ \& } p, \text{ } q \text{ is true}$$
$$\therefore q$$

$$p \to q \equiv \neg p \lor q$$

$$\frac{\begin{array}{c}\neg p \lor q \\ p\end{array}}{q}$$

* Modus Tollens

$$\frac{\begin{array}{c}p \to q \\ \neg q\end{array}}{\therefore \neg p}$$

$$\frac{\begin{array}{c}\neg p \lor q \\ \neg q\end{array}}{\neg p}$$

Question 11

Prove that conclusion logically entails from the premise

Premises
$P \lor Q$
$P \to R$
$Q \to R$

$$\frac{\begin{array}{c}\cancel{P} \lor \cancel{Q} \\ \cancel{\neg P} \lor R \\ \cancel{\neg Q} \lor R\end{array}}{R}$$

Conclusion
R

Question 12

Prove that conclusion logically entails from the premise

Premises

$P \to Q$ $\qquad\qquad$ $\neg P \lor Q$

$R \to S$ $\qquad\qquad$ $\neg R \lor S$

Conclusion

$(P \lor R) \to (Q \lor S)$

negate the conclusion

$\equiv \neg(\neg(P \lor R) \lor (Q \lor S))$

$\equiv (P \lor R) \land \neg(Q \lor S)$

$\equiv (P \lor R) \land (\neg Q \land \neg S)$

Resolve premise and ¬conclusion

$\cancel{\neg P} \lor \cancel{Q}$

$\cancel{\neg R} \lor \cancel{S}$

$\cancel{P} \lor \cancel{R}$

$\cancel{\neg Q}$

$\cancel{\neg S}$

$0 \equiv$ false

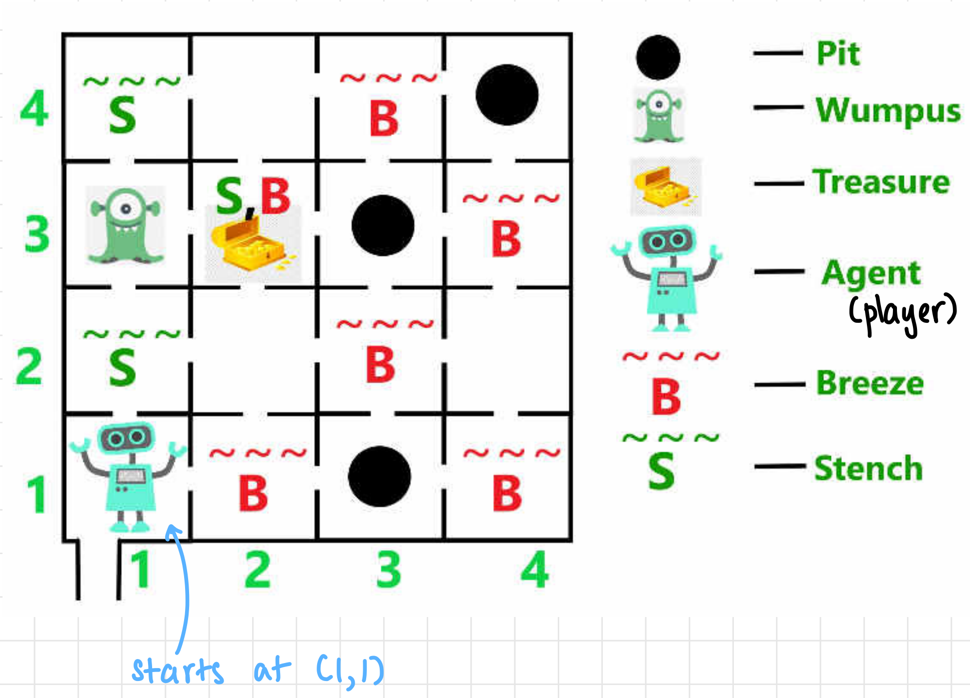
$\therefore$ conclusion follows the premise

$(A \land \neg A$ should be false$)$

WUMPUS WORLD

- Model of game: certain aspects immutable
- Wumpus : demon
- Artificial Intelligence
- 4x4 grid-world

geeksforgeeks



- As the player progresses in the game, they gain more information about the world (adds to knowledge)
- Breeze adjacent to pits (not diagonal)
- Game ends if player (agent) enters pit or meets Wumpus

- Player cannot see all squares (like minesweeper)
- Player can feel a breeze; tells them that pit is near
- Player can feel a stench; tells them that Wumpus is near
- Goal: find safest path and exit
- Simulator: https://thiagodnf.github.io/wumpus-world-simulator/

press enter to claim gold

shoot arrow (space) to kill Wumpus

Knowledge-Based Agent

- Software that gathers information about an environment and takes actions based on that (player in a game)
- It could be a robot, a web shopping program, traffic control system etc.
- Knowlege-based agent composed of two parts: knowledge base and inference system

  knowledge base → initial facts ((1,1) is safe)

  inference system → percepts (breeze, stench, gold)
- Logical entailment

Knowlege Base

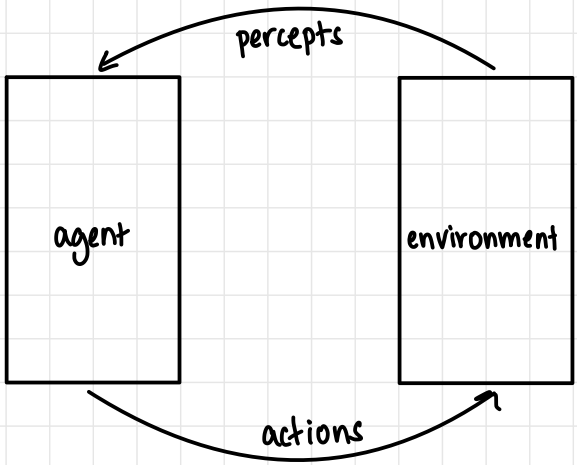
- Knowledge base is a collection of sentences (facts about the world)
- Sentences are expressed in a knowledge representation language/ declarational language (Propositional Logic, First Order Predicate Logic)
- Lots of work to create a sentence for each square wrt breeze, stench, Wumpus, pit (Propositional Logic)
- Cannot generalise using Propositional Logic (cannot represent patterns)

Inference System

- Deriving new sentences from old
- Add sentences to knowledge base
- Sentence: proposition about the world
- Inference system applies logical rules to the knowledge base to deduce new information (logical entailment)
- Inference system generates new facts so that an agent can update the KB.

Objective

- See how knowledge-based agent can represent the world in which it operates (KB) and deduce what actions it should take.
- Use Propositional Logic as representational logic



Agent: player

Environment: Wumpus World Model

Actions: L, R, U, D

Percepts: from sensors (stench, breeze, gold)

Operations Performed by KB-Agent

- Input from environment (to KB)
- Asks KB what action to perform
- Output (performs action)

World Model

- Action space (actuators)
- Percept space (sensors)
- Environment

Action Space (Actuators)

- Move forward
- Turn left/right by 90°
- Grab (gold)
- Shoot (in a straight line)

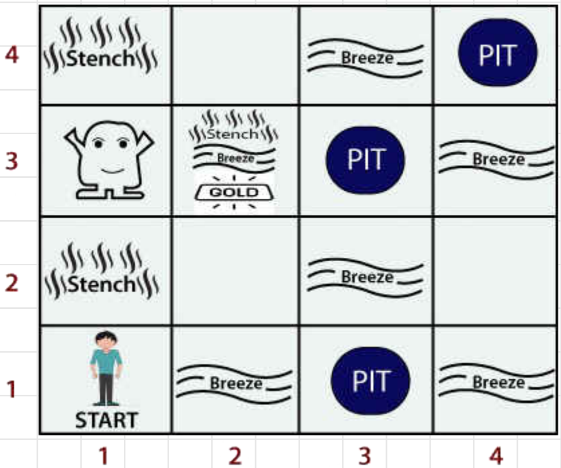
Percept Space (Sensors)

- Cells adjacent to a cell that contains Wumpus (not diagonally) give out a stench that the Agent can perceive
- Cells adjacent to a cell that contains a pit (not diagonally) give out a breeze that the Agent can perceive
- Cells containing gold glitter that the Agent can perceive
- When Agent walks into a wall, it perceives a bump

Environment

- 4x4 Grid of rooms (could be any size)
- Agent always starts in the (1,1) cell while facing to the right
- Locations of gold and Wumpus are chosen randomly with a uniform distribution (except (1,1))
- Any cell can be a pit (except (1,1)) with a probability of 0.2
- Game ends when Agent dies or exicts successfully
- Points are awarded each time gold is obtained

The Wumpus World Knowlege Base



A Typical Wumpus World

1. P_{xy} is true if there is a pit at (x,y) → 16 propositions
2. W_{xy} is true if there is a Wumpus in (x,y) → 16 props
3. B_{xy} is true if agent perceives a breeze in (x,y) → 16 props
4. S_{xy} is true if agent perceives a stench in (x,y) → 16 props

- Totally 64 propositions ⇒ 2^{64} size of truth table

Aspects of the Wumpus world

General Aspects - Immutable (Applies to all models) - facts known

Proposition	Meaning
$R_1 : \neg P_{11}$ ①	No pit in (1,1)
$R_2 : B_{11} \leftrightarrow (P_{12} \vee P_{21})$ ② ③ ④	Breeze in (1,1) iff pit in adjoining cell (1,2) or (2,1)
$R_3 : B_{21} \leftrightarrow (P_{11} \vee P_{31} \vee P_{22})$ ⑤ ① ⑦ ⑥	Breeze in (2,1) iff pit in adjoining cell (1,1), (3,1) or (2,2)

Aspects Related to a Specific Model - facts gained (figure)

Proposition	Meaning
$R_4 : \neg B_{11}$ ②	No breeze in (1,1)
$R_5 : B_{21}$ ⑤	Breeze in (2,1)

- Here, we have written 7 propositions ($2^7 = 128$ entries in truth table)

- 128 rows

premises AND of R_is

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

- If one or more rows are true, KB is said to be satisfiable
- Every row is a model
- If KB is true, what can be entailed?
- KB $\models \alpha$ (α logically follows)
- KB $\rightarrow \alpha$ is a tautology
- Can entail no breeze (1,1), breeze in (2,1), no pit in (1,1), (1,2) or (2,1) (infer)
- Called model checking; inefficient

Proof By Resolution

$KB \models \neg P_{12}$

Premises	Conclusion
(unrelated)	
1. $\neg P_{11}$	$\neg P_{12}$
2. $B_{11} \leftrightarrow (P_{12} \vee P_{21})$	need pair for P_{12}
3. $B_{21} \leftrightarrow (P_{11} \vee P_{31} \vee P_{22})$	
4. $\neg B_{11}$	
5. B_{21}	

Steps

1. CNF
2. Negate conclusion $\longrightarrow P_{12}$
3. Resolution rules until false / stuck

Premises	Conclusion
1. $B_{11} \leftrightarrow (P_{12} \vee P_{21})$	Reduce to CNF $\equiv (B_{11} \rightarrow (P_{12} \vee P_{21})) \wedge ((P_{12} \vee P_{21}) \rightarrow B_{11})$ $\equiv (\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg (P_{12} \vee P_{21}) \vee B_{11})$ $\equiv (\neg B_{11} \vee P_{12} \vee P_{21}) \wedge ((\neg P_{12} \wedge \neg P_{21}) \vee B_{11})$ $\equiv (\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg P_{12} \vee B_{11}) \wedge (\neg P_{21} \vee B_{11})$
2. $\neg B_{11}$	$\neg B_{11}$

Resolve

(i) $\neg \cancel{B_{11}} \vee \cancel{P_{12}} \vee \cancel{P_{21}}$
(ii) $\neg \cancel{P_{12}} \vee \cancel{B_{11}}$
(iii) $\neg \cancel{P_{21}} \vee \cancel{B_{11}}$
(iv) $\neg \cancel{B_{11}}$
(v) $\cancel{P_{12}}$

false $\longrightarrow$ conclusion logically follows from premises

1. (i) & (v)
2. order does not matter

Limitations of Propositional Logic

- Cannot capture patterns
- Eg: we have to write separate rules about breeze and pits for every cell even though the rules convey the same information
- A breeze exists in cell (x,y) iff there is a pit in adjoining cells — $(x,y+1)$, $(x,y-1)$, $(x-1,y)$, $(x+1,y)$
- Can overcome using First Order Predicate Logic (FOPL)

Propositional Logic (PL)	First Order Predicate Logic (FOPL)
Declarative Language	Declarative Language; extention to PL
Used to represent logic in simple environments	Used to represent logic in complex environments
Lacks expressive power to concisely describe an environment	quantifiers Can represent rules generally (no need for separate rules for each cell)
Assumes world contains facts	Assumes world contains objects, relations and functions
Facts either hold or do not hold	Relations between objects either hold or do not hold

FOPL- Describing a Model

1) Set of objects - Domain of the model

2) Relations - specify how objects are related, whether relationship holds or not
   - Input : Object(s) or Function(s)
   - Output : true or false

3) Functions - special case of relation where object is related to exactly one object. Must be a total function (defined for all valid inputs)
   - Input: Object(s)
   - Output: Object

Example

Relation: mother(Gunjan, Ravi)
Function: mother(Ravi) returns Gunjan

Basic Elements of FOPL Syntax

Constant symbols (Objects)	1, 2, A, John, Mumbai, Chair
Predicate symbols (relations)	Brother, Father, >
Function symbols	sqrt, multiply
Variables	x, y, z, a, b
Term	Constant/function Symbol or variable
Equality	$=$
Connectives	$\neg, \vee, \wedge, \rightarrow, \leftrightarrow$
Quantifier	$\forall, \exists$

Quantifiers in FOPL

- Allows to make general assertions using quantifiers
- Universal quantifier – $\forall$ – for all, for each, for every.
  Eg: $\forall x$ = for each x
- Existential quantifier – $\exists$ – for some, at least one.
  Eg: $\exists x$ = for some x

Properties of Quantifiers

- $\forall x \forall y$ similar to $\forall y \forall x$
- $\exists x \exists y$ similar to $\exists y \exists x$
- $\forall x \exists x$ not similar to $\forall y \exists x$

- Main connector for $\forall$ is implication $\longrightarrow$
  Eg: Let the domain be all animals

$D(x)$: x is a dog   x is an animal
$M(x)$: x is a mammal
$C(x)$: x is cute

"All dogs are animals"

(i) $\forall x\,(D(x) \longrightarrow M(x))$ ✓ ← best connective
for every x, if x is a dog then x is a mammal

(ii) $\forall x\,(D(x) \wedge M(x))$ X
for every x, x is a dog and x is a mammal

- Main quantifier for $\exists$ is and $\wedge$

Eg: Let the domain be all animals

$D(x)$: x is a dog x is an animal
$M(x)$: x is a mammal
$C(x)$: x is cute

"Some dogs are cute"

(i) $\forall x\,(D(x) \rightarrow C(x))$ X
for some x, if x is a dog then x is cute

(ii) $\forall x\,(D(x) \wedge C(x))$ ✓ ← best connective
for some x, x is a dog and x is cute

Free and Bound Variables

1. Free Variable - if the variable occurs outside the scope of a quantifier

eg: $\forall x\ \exists(y)\,[P(x,y,z)]$

can take any value

2. Bound variable - if the variable occurs inside the scope of a quantifier

eg: $\forall x\,[A(x)\ B(y)]$

- Variables can take values in a domain and can only be used together with quantifiers

$(\forall x \forall y)$

- eg: $\forall x, y$ brother$(x, y) \rightarrow$ sibling(x, y)

  for all x, y if x is a brother of y then x is a sibling of y

- eg: $\exists x \; 2*5 + x = 18$

  there exists an x such that $2*5 + x = 18$

Term

- Constant symbols (objects)
- Function symbols
- Variables

- term equality (=)

Atomic Sentence

- Most basic sentences in first-order logic

- Predicate symbol followed by a parenthesis with a sequence of terms

- Prolog: fact
  eg: brothers(Ravi, Ajay).
  bigger(elephant, horse).

- Functions do not state facts and form no sentence

Complex Sentence

- Complex sentences are made by combining atomic sentences using connectives ($\neg, \vee, \wedge, \rightarrow, \leftrightarrow$)
- Eg: find(x,z) :- bigger(x,y), bigger(y,z); bigger(x,z)

Grammar

$Sentence \rightarrow AtomicSentence \mid ComplexSentence$

$AtomicSentence \rightarrow Predicate \mid Predicate(Term, \ldots) \mid Term = Term$

$ComplexSentence \rightarrow (\ Sentence\ ) \mid [\ Sentence\ ]$

$\quad \mid \neg\ Sentence$

$\quad \mid Sentence \wedge Sentence$

$\quad \mid Sentence \vee Sentence$

$\quad \mid Sentence \Rightarrow Sentence$

$\quad \mid Sentence \Leftrightarrow Sentence$

$\quad \mid Quantifier\ Variable, \ldots\ Sentence$

$Term \rightarrow Function(Term, \ldots)$

$\quad \mid Constant$ → map to objects

$\quad \mid Variable$

$Quantifier \rightarrow \forall \mid \exists$

$Constant \rightarrow A \mid X_1 \mid John \mid \cdots$

fact

$Variable \rightarrow a \mid x \mid s \mid \cdots$

$Predicate \rightarrow True \mid False \mid After \mid Loves \mid Raining \mid \cdots$

$Function \rightarrow Mother \mid LeftLeg \mid \cdots$

OPERATOR PRECEDENCE : $\neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow$ (brackets first)

Number of Possible Words / Models

- In propositional logic, if there are 100 propositions / symbols, there are 2^{100} worlds (models) — each row
- In FOPL, no. of models depend on no. of objects and mapping of constant symbols to objects.
- Predicates are true or false

Question 13

Suppose that we have five constants, no functions and one predicate that takes in one argument. How many possible worlds are there?

Suppose the constants are v, w, x, y, z

Predicates can take in : $P(v)$, $P(w)$, $P(x)$, $P(y)$, $P(z)$

T F T F T F T F T F

Five possible arguments

Size of truth table $= 2^5$ (no. of models)

Model	P(V)	P(W)	P(X)	P(Y)	P(Z)
1	*false*	*false*	*false*	*false*	*false*
2	*true*	*false*	*false*	*false*	*false*
...	...	...	...	...	...
...	...	...	...	...	...
2^5 =32	*true*	*true*	*true*	*true*	*true*

Question 14

Suppose that we have five constants, no functions and three predicates that take in two arguments and one that takes in one. How many possible worlds are there?

Five constants (v, w, x, y, z)

$\underset{5^2}{P_1(\text{any } 2)}$, $\underset{5^2}{P_2(\text{any } 2)}$, $\underset{5^2}{P_3(\text{any } 2)}$, $\underset{5}{P_4(\text{any } 1)}$

$25 \times 3 + 5 = 80$ possibilities

$\therefore 2^{80}$ possible worlds

Model checking is too inefficient

Question 15

For the Wumpus World, to say that "pits cause breezes in adj squares" using propositional logic, 16 rules are required.

$$B_{12} \rightarrow (P_{11} \vee P_{22} \vee P_{13})$$

How can we say the same thing in FOPL?

Objects: Pits, Squares
Relations: Breeze, adjacent

$$\forall x_1, y_1\ Breeze(x_1, y_1) \longleftrightarrow (\exists x_2, y_2\ Pit(x_2, y_2) \text{ AND } Adjacent(x_1, y_1, x_2, y_2))$$

for every (x_1, y_1) there is a breeze iff there is a pit in an adjacent square

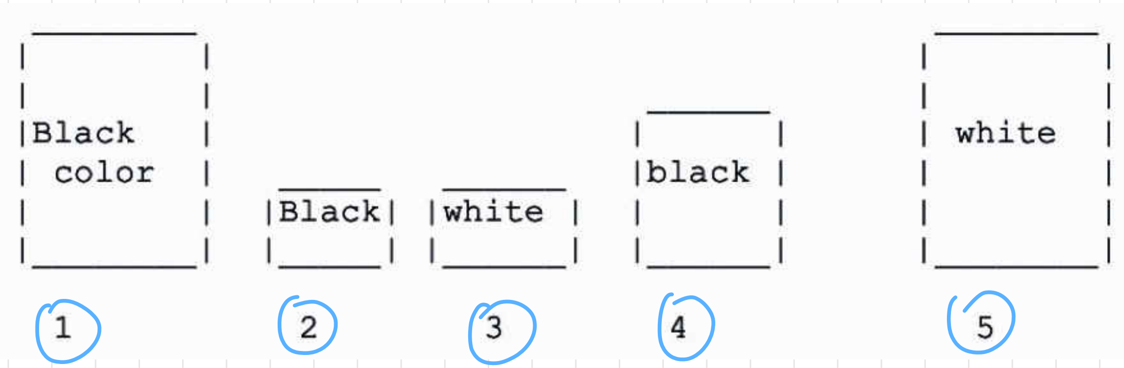
Knowledge Representation using FOPL

- Box solver
- Kinship
- Mathematical sets

Box Solver Problem

- There are 5 boxes
- Each of Ann, Bill, Charlie, Don and Eric own a box but we don't know which
- Try to find owners if you know that

  1) Ann & Bill – same colour
  2) Don & Eric – same colour
  3) Charlie & Don – same size
  4) Eric's is smaller than Bill's



Object: box
Unary predicate: box
Ternary predicate: box_details
Function: Owners

FOPL Representation

$\exists$ A, B, C, D, E, c1, c2, c3, s1, s2, s3, s4
(A $\neq$ B $\neq$ C $\neq$ D $\neq$ C) $\wedge$
box-details (A, c1, s1) $\wedge$ box-details (B, c1, s2) $\wedge$ % Ann & Bill same colours
box-details (D, c2, s3) $\wedge$ box-details (E, c2, s4) $\wedge$ % Don & Eric same colours
box-details (D, c2, s3) $\wedge$ box-details (C, c3, s3) $\wedge$ % Don & Charlie same size
s4 < s2 % Eric smaller than Bill

Prolog

```
% Numbers for each box
% Unary predicate
box(1).
box(2).
box(3).
box(4).
box(5).

% Box number, colour, size
% Ternary predicate
box_details(1, black, 3).
box_details(2, black, 1).
box_details(3, white, 1).
box_details(4, black, 2).
box_details(5, white, 3).

% Owners
owners(A,B,C,D,E):-
    box(A), box(B), box(C), box(D), box(E),
    A\=B, A\=C, A\=D, A\=E,
    B\=C, B\=D, B\=E,
    C\=D, C\=E,
    D\=E,

    box_details(A,C1,_), box_details(B,C1,S2),
    box_details(D,C2,S3), box_details(E,C2,S4),
    box_details(D,C2,S3), box_details(C,_,S3),
    S4 < S2.
```

run on https://swish.swi-prolog.org

```prolog
% Numbers for each box
% Unary predicate
box(1).
box(2).
box(3).
box(4).
box(5).

% Box number, colour, size
% Ternary predicate
box_details(1, black, 3).
box_details(2, black, 1).
box_details(3, white, 1).
box_details(4, black, 2).
box_details(5, white, 3).

% Owners
owners(A,B,C,D,E):-
    box(A), box(B), box(C), box(D), box(E),
    A\=B, A\=C, A\=D, A\=E,
    B\=C, B\=D, B\=E,
    C\=D, C\=E,
    D\=E,

    box_details(A,C1,_), box_details(B,C1,S2),
    box_details(D,C2,S3), box_details(E,C2,S4),
    box_details(D,C2,S3), box_details(C,_,S3),
    S4 < S2.
```

don't care: _

A: 2, B: 4, C: 1, D: 5, E: 3

solution

owners(A,B,C,D,E)

A	B	C	D	E
2	4	1	5	3

Next 10 100 1,000 Stop

?- owners(A,B,C,D,E)

Examples History Solutions table results Run!

Kinship Problem

Object: people
Unary predicates: male, female
Binary predicates: parent, brother, sister
Functions: father, mother
Relations: brotherhood, sisterhood

$\forall x, \ Male(x) \leftrightarrow \neg Female(x)$

$\forall x,y \ \ Brother(x,y) \rightarrow Sibling(x,y)$

$\forall p,c \ \ Parent(p,c) \leftrightarrow Child(c,p)$

learn to translate to English

Mathematical Set Representation

Constant: empty set ($s = \{\}$)
Predicate: member and subset
Function: intersection and union O/P: sets

Equal sets

$$\forall s_1, s_2 \ (s_1 = s_2) \Leftrightarrow subset(s_1, s_2) \wedge subset(s_2, s_1)$$

x belongs to intersection

$$\forall x, s_1, s_2 \ x \in (s_1 \cap s_2) \Leftrightarrow x \in s_1 \wedge x \in s_2$$

x belongs to union

$$\forall x, s_1, s_2 \ x \in (s_1 \cup s_2) \Leftrightarrow x \in s_1 \vee x \in s_2$$

Whole Numbers

Constant: 0
Function: Successor
Predicate: Whole Num

- Whole Num(0).

0 succeeds nothing

$$\forall x \ Successor(x) \neq 0$$

inequality of successors of unique numbers

$$\forall m, n \quad m \neq n \rightarrow Successor(m) \neq Successor(n)$$

$0 + n = n$

$$\forall m \ \text{WholeNum}(m) \Rightarrow +(0,m) = m$$

Addition in terms of successor

$$\forall m,n \ \text{WholeNum}(m) \wedge \text{WholeNum}(n) \Rightarrow +(S(m),n) = S(+(m,n))$$

Successor of WN is WN

$$\forall x \ \text{WholeNum}(x) \Rightarrow \text{WholeNum}(S(x))$$

Knowledge Engineering in FOPL

Process of constructing a knowledge

identify the task

↓

assemble relevant knowledge

↓

decide vocabulary func, pred

↓

encode general knowlege of domain

↓

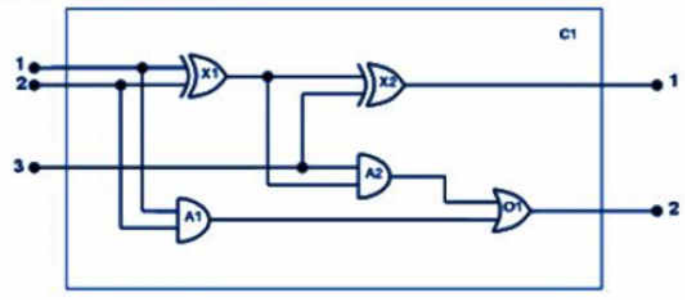
pose queries and get answers

↓

define knowledge base

Electronic Circuit Domain

One Bit Full Adder



$1 \oplus 2 \oplus 3$

$1 \cdot 2 + 3 \cdot (1 \oplus 2)$
$= 1 \cdot 2 + 2 \cdot 3 + 1 \cdot 3$
(reduce using laws)

Object: gates, circuits, terminals etc

Constants: $X_1, X_2, A_1, A_2, O_1, 1, 2, 3, C_1$

Predicates: Gate(x), Circuit(x), Terminal(x), Arity(x, i, j), Connected(t1, t2)

#in #out (terminals in & out)

terminals connected

Functions: Type(x), In(n, Gate), Out(n, Gate), Signal(t)

AND, OR etc.

nth input terminal

value of signal at terminal t (0 or 1)

Encode Knowledge

$$\forall t_1, t_2 \; Connected(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$$

$$\forall t \; Signal(t) = 1 \lor Signal(t) = 0$$

$$1 \neq 0$$

OR is 1 if input is 1 (at least 1)

$$\forall g \; Type(g) = OR \Rightarrow Signal(Out(1, g)) \Leftrightarrow \exists n \; Signal(In(n, g)) = 1$$

AND is 0 if any input 0

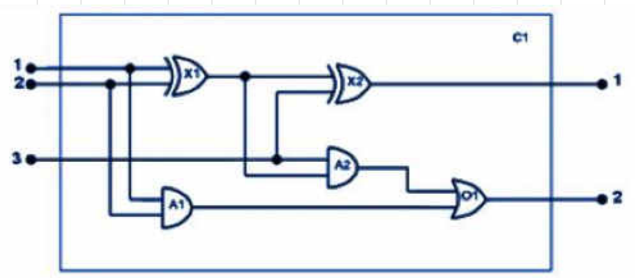
$\forall g \; Type(g) = AND \Rightarrow Signal(Out(1,g)) = 0 \Leftrightarrow \exists n \; Signal(In(n,g)) = 0$

XOR is 1 if inputs are different

$\forall g \; Type(g) = XOR \Rightarrow Signal(Out(1,g)) = 1 \Leftrightarrow Signal(In(1,g)) \neq Signal(In(2,g))$

Description of Specific Instance (Encoding)

not ln; In



Connected(In(1,C1), In(1, X1))
Connected(In(2,C1), In(2, X1))

Connected(In(1,C1), In(1, A1))
Connected(In(2,C1), In(2, A1))

Connected(Out(1,X1), In(1, X2))
Connected(In(3,C1), In(2, X2))

Connected(In(3,C1), In(1, A2))
Connected(Out(1,X1), In(2, A2))

Connected(Out(1,A2), In(1, O1))
Connected(Out(1,A1), In(2, O1))

Query

$\exists \; i1, i2, i3, o1, o2$
$signal(in(1,c1)) = i1 \; \wedge$
$signal(in(2,c1)) = i2 \; \wedge$
$signal(in(3,c1)) = i3 \; \wedge$
$signal(out(1,c1)) = o1 \; \wedge$
$signal(out(2,c1)) = o2$

Prolog

-? find (i1,i2,i3,o1,o2)