Ishita Haridasmalani
C14-2103058
THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

# EXPERIMENT NO.6.

**Aim:** Knowledge Representation & Knowledge Base for Wumpus World.

Q1. Represent the following sentences in First Order Logic, using consistent vocabulary.

a. Some students took French in Spring 2001.
b. Every student who takes French passes it.
c. Only one student took Greek in Spring 2001.
d. The best score in Greek is always higher than the best score in French.
e. Every person who buys a policy is smart.
f. No person buys an expensive policy.
g. There is an agent who sells policies only to people who are not insured.
h. There is a barber who shaves all men who do not shave themselves.
i. A person born in UK each of whose parents is a UK citizen or a UK resident, is a UK citizen by birth.
j. A person born outside UK, one of whose parents is a UK citizen by birth, is a UK citizen by descent.

* vocabulary :

Takes (x, C, S) :
student 'x' takes course 'C' in semester S;

Passes (x, C, S) :
student x passes course C in semester S;

Score (x, C, S) :
the score obtained by student x in course C in semester S;

x > y : x is greater than y

F & G : specific French & Greek courses

Buys (x, y, z) :
x buys y from z

Sells (x, y, z) :
x sells y to z;

Shaves (x, y) :
person x shaves person y

Born (x, C) :
person x is born in country C;

Parent (x, y) : x is parent of y.

Citizen(x, C, r):
x is a citizen of country C for reason r;

Resident(x, C):
x is a resident of country C;

Student(x)
Person(x)
Man(x)
Barber(x)
Expensive(x)
Agent(x)
Insured(x)
Smart(x)

a. Some students took French in spring 2001.

$\exists x$ Student(x) $\wedge$ Takes(x, F, Spring2001)

b. Every student who takes French passes it.

$\forall x, s$ Student(x) $\wedge$ Takes(x, F, s) $\Rightarrow$ Passes(x, F, s)

c. Only one student took Greek in Spring 2001.

$\exists x$ Student(x) $\wedge$ Takes(x, G, Spring 2001) $\wedge \forall y$, $y \neq x \Rightarrow \neg$ Takes(y, G, Spring 2001).

d. The best score in Greek is always higher than the best score in French.

$\forall s \exists x \forall y \; Score(x, G, S) \neg Score(y, F, S)$

e. Every person who buys a ~~poly~~ policy is smart.

$\forall x \; Person(x) \wedge (\exists y, z \; Policy(y) \wedge Buys(x, y, z)) \rightarrow Smart(x)$

f. No person buys an expensive policy.

$\forall x, y, z \; Person(x) \wedge Policy(y) \wedge Expensive(y) \Rightarrow \neg Buys(x, y, z)$

g. There is an agent who sells policies only to people who are not insured.

$\exists x \; Agent(x) \wedge \forall y, z \; Policy(y) \wedge Sells(x, y, z) \rightarrow (Person(z) \wedge \neg Insured(z))$.

h. There is a barber who shaves all men who do not shave

$\exists x \; Barber(x) \wedge \forall y \; (Man(y) \wedge \neg Shaves(y, y)) \rightarrow Shaves(x, y)$

i) A person born in UK each of whose parents is a UK citizen or a UK resident, is a UK citizen by birth

$\forall x \; Person(x) \wedge Born(x, UK) \wedge \forall y \; Parent(y, x) \Rightarrow ((\exists r \; Citizen(y, UK, r)) \vee Resident(y, UK)) \rightarrow Citizen(x, UK, Birth)$

j) $\forall x \; (Person(x) \wedge \neg Born(x, UK) \wedge (\exists y \; Parent(y, x) \wedge Citizen(y, UK, Birth)) \Rightarrow Citizen(x, UK, Parent)$.

Q2. Describe the wumpus world problem in detail & create a knowledge base for the wumpus world & prove that the wumpus is present in [1, 3].

Ans. The wumpus world is a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the wumpus, a beast that eats anyone who enters its room. The wumpus can be shot by an agent, but the agent has only one arrow. Some rooms contain bottomless pits that will trap anyone who wanders into these rooms. The only mitigating feature of living in this environment is the possibility of finding a heap of gold. Although the wumpus world is rather tame by modern computer game standards, it makes an excellent testbed environment for intelligent agents.

The precise definition of the task environment is given PEAS description:

Performance measure: +1000 for picking up the gold, -1000 for falling into a pit or being eaten by the wumpus, -1 for each action taken & -10 for using up the arrow.

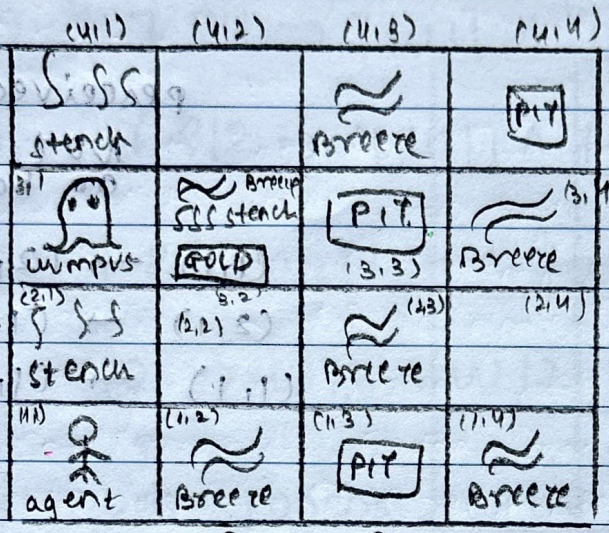
Environment: A 4 x 4 grid of rooms. The agent always starts in the square labeled [1, 1] facing to the right. The locations of the gold & wumpus are chosen randomly, with a uniform distribution, from the squares other than the start square. In addition,

each square other than the start can be a pit with probability 0.2

Actuators: The agent can move forward, turn by 90° or turn by 90°. The agent dies a miserable death if it enters a square containing a pit or a live wumpus. Moving forward has no effect if there is a wall in front of the agent. The action grab can be used to pick up an object that is in the same square as the agent. The action shoot can be used to fire an arrow in a straight line in the direction the agent is facing. The arrow continues until it either hits the wumpus or hits the wall. The agent only has one arrow, so only the first shoot action has any effect.

Sensors: The agent has five sensors, each of which gives a bit of information.

- In the square containing the wumpus and in the directly adjacent squares the agent will perceive a stench.




B = Breeze, A = Agent, P = Pit, V = visited, OK = safe,
S = stench, W = wumpus, G = Gold
First percept [NONE, NONE, NONE, NONE, NONE].

Step 1:

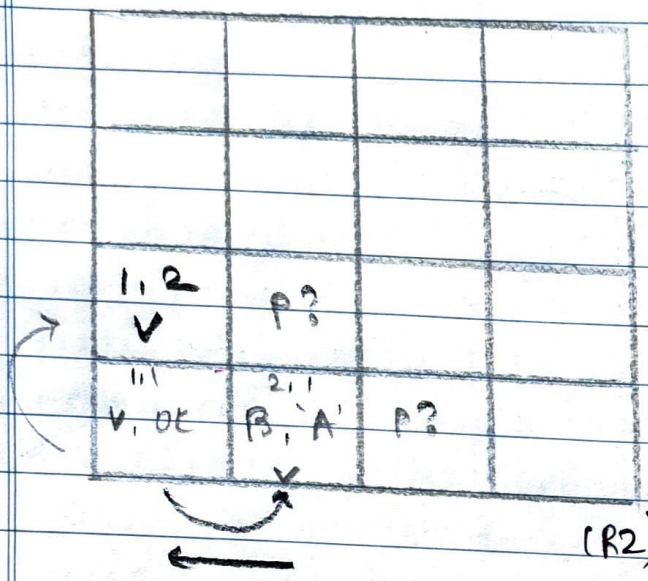
| | | | |
|---|---|---|---|
| (4,1) | (4,2) | (4,3) | (4,4) |
| (3,1) | (3,2) | (3,3) | (3,4) |
| (2,1) | (2,2) | (2,3) | (2,4) |
| (1,1) A OK | (1,2) | (1,3) | (1,4) |

room is safe,
no stench, no breeze

(R1) $\neg S11 \rightarrow \neg W11 \wedge \neg W12 \wedge \neg W21$

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Step 2:



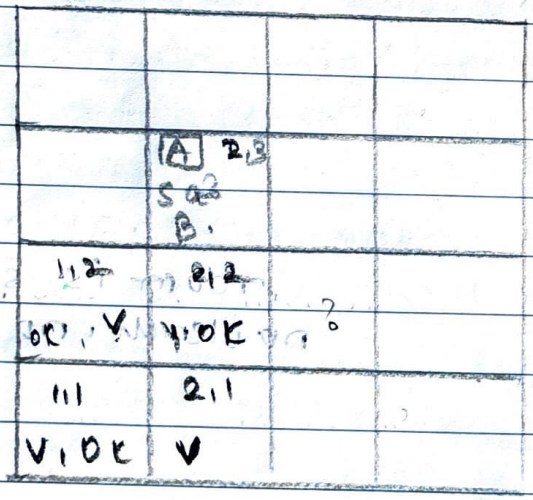

perceived breeze, NOT SAFE and backtrack.

(1,1) → (2,1)
(2,1) → (1,1)
(1,1) → (1,2).

(R2) ¬S21 → ¬W11 ∧ ¬W21 ∧ ¬W22 ∧ ¬W31.

(R4) S12 → W13 ∨ W12 ∨ W22 ∨ W11.

Step 3:
(1,2) stench




(1,2) → (2,2)
safe.

(2,2) → (2,3)

(2,3) - Grab Gold and return home.

Final traversed
(1,1), (1,2), (2,2) (2,3.

final rules:

(R1) $\neg S11 \rightarrow \neg W11 \wedge \neg W12 \wedge \neg W21$

(R2) $\neg S12 \rightarrow \neg W11 \wedge \neg W21 \wedge \neg W22 \wedge \neg W31$

(R3) $\neg S12 \rightarrow \neg W11 \wedge \neg W12 \wedge \neg W22 \wedge \neg W13$.

(R4) $S12 \rightarrow W13 \vee W12 \vee W22 \vee W11$ ..

① Apply modus ponens with $\neg S11$ & R1:

(R1) $\neg S11 \rightarrow \neg W11 \wedge \neg W12 \wedge \neg W21$

$\neg S11$

$\boxed{\neg W11 \wedge \neg W12 \wedge \neg W21}$

Apply And-elimination to this, we get,
$\neg W11$, $\neg W12$, $\neg W21$

② Apply modus ponens to $\neg S21$ & R2

(R2) $\neg S21 \rightarrow \neg W11 \wedge \neg W21 \wedge \neg W22 \wedge \neg W31$.

$\neg S21$

$\boxed{\neg W11 \wedge \neg W21 \neg W22 \wedge \neg W31}$

Apply And-elimination to this we get:

$\neg W21$, $\neg W22$, $\neg W31$, $\neg W11$

③ Apply modus ponens to S22 & R4

(R4) $S12 \rightarrow W13 \vee W12 \vee W22 \vee W11$

S12

$\boxed{W13 \vee W12 \vee W22 \vee W11}$

Apply unit resolution on
$W13 \vee W12 \vee W22 \vee W11$ ← $\neg W11$.

$\boxed{W13 \vee W12 \vee W22}$

Apply unit resolution with
$W13 \vee W12 \vee W22$ ← $\neg W22$

$\boxed{W13 \vee W12}$

Apply unit resolution with
$W13 \vee W12$ ← $\neg W12$

$\boxed{W13.}$

Hence, proved.

Analysing the time & space complexity for Wumpus world depends on factors such as grid size, search algorithm, complexity of rules, representation of the environment & agent's knowledge.

Ishita Hardasmalani
C14 - 2103058.



## EXPERIMENT NO. 7

**Aim:** Planning for Blocks world problem.

Q1. Describe the Blocks world planning problem & discuss how it can be solved using classical planning Method.

The Blocks world planning problem is a classic problem in artificial intelligence & planning. It involves a simplified world where blocks of different shapes & sizes are placed on a table. The set of actions, the goal is to rearrange the blocks from an initial configuration to a desired goal configuration using a set of actions while minimizing the cost associated with these actions.

Initial state: The initial state of the Blocks world consists of a set of blocks placed on a table. Blocks can be stacked on top of each other, & they can also be on the table with no blocks on top of them.

Goal state: The goal state specifies a desired configuration of the blocks. This configuration could involve specific blocks stacked on top of each other or certain blocks

placed at particular locations.

Actions: Actions in the Blocks world typically involved picking up a block, putting down a block, or moving a block from one position to another.

Pickup(x): Pick up block 'x' from its current position.

Putdown(x): Put down block x at a specific location.

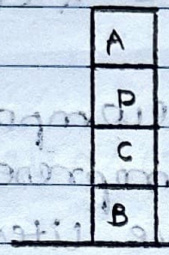
Stack(x, y): Stack block x on top of block y.

Unstack(x, y): Remove block x from the top of block y.

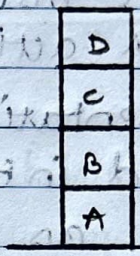
Step cost: Each action in the Blocks world may have an associated cost. The step cost could be uniform for all actions or vary depending on the actions performed.

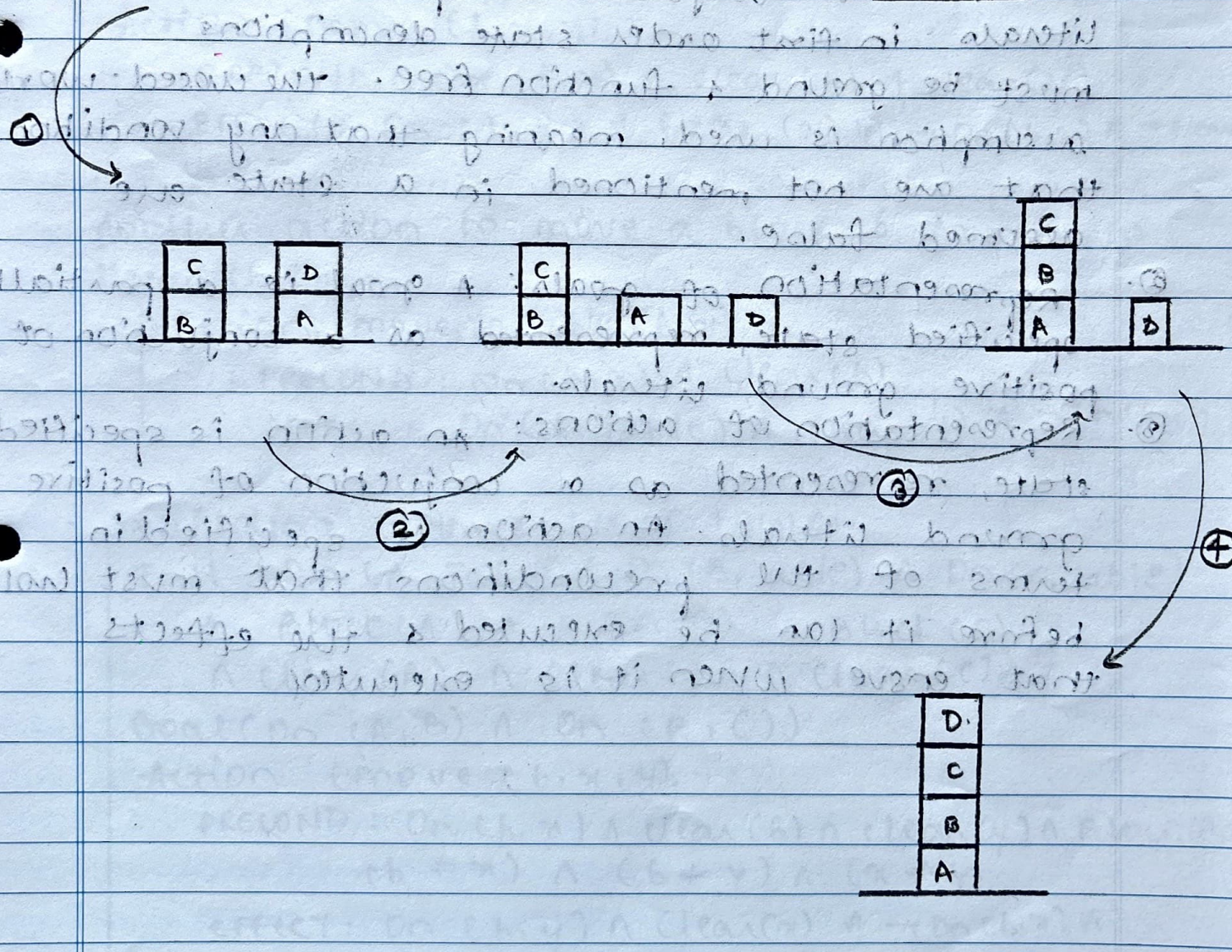
Example:

Start




Goal



Classical planning methods aims to find a sequence of actions that transform the initial state of the world into the goal state while obeying the preconditions & effects of each action.

① Representation of states: Planners decompose the world into logical conditions & represent a state as a conjuction of positive literals. literals in first order state descriptions must be ground & function free. The closed-world assumption is used, meaning that any condition that are not mentioned in a state are assumed false.

② Representation of goals: A goal is a partially specified state, represented as a conjuction of positive ground literals.

③ Representation of actions: An action is specified state, represented as a conjuction of positive ground literals. An action is specified in terms of the preconditions that must hold before it can be executed & the effects that ensue when it is executed

Solving block world problem using classical planning.

→ On(b,x): Block b is on block x or on the table

clear(x): There is nothing on block x

Action (move (b,x,y),
PRECOND: On(b,x) ∧ clear(b) ∧ clear(y),
EFFECT: On(b,y) ∧ clear(x) ∧ ¬On(b,x) ∧ ¬clear(y))

Another action to move a block b from x to the table:

Action (move to Table (b,x),
PRECOND: On(b,x) ∧ Clear(b),
EFFECT: On(b, Table) ∧ clear(x) ∧ ¬On(b,x)).

Building a three-block tower.

Init (On(A, Table) ∧ On(B, Table) ∧ On(C, Table)
∧ Block(A) ∧ Block(B) ∧ Block(C)
∧ clear(A) ∧ clear(B) ∧ clear(C))
Goal(On(A, B) ∧ On(B, C))
Action (move (b,x,y),
PRECOND: On(b,x) ∧ clear(b) ∧ clear(y) ∧ Block(b) ∧
(b ≠ x) ∧ (b ≠ y) ∧ (x ≠ y),
EFFECT: On(b,y) ∧ clear(x) ∧ ¬On(b,x) ∧
¬clear(y))

Action: moveToTable (b, x),
PRECOND : On(b, x) ∧ clear(b) ∧ block(b) ∧ (b ≠ x),
effect : On(b, table) ∧ clear(x) ∧ ¬ On(b, x)).

The time complexity & space complexity of the Blocks world problem can vary depending on the specific algorithm & representations used. It is typically exponential, as the problem involves searching through a large state space of possible configurations of blocks & actions to achieve a goal state. The SC also tends to be exponential or high polynomial due to the need to store & explore multiple states & actions during the search process, often scaling with the no. of blocks & the depth of the search.

Ishita Hardasmalani
C14-2103058
THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

# EXPERIMENT NO. 8.

**Aim:** Implement Family Tree Using Prolog.

**Theory:**

Prolog is the most widely used logic programming language. It is used primarily as a rapid prototyping language & for symbol-manipulation tasks such as: writing compilers & parsing natural language. Prolog programs are sets of definite clauses written in a notation somewhat different from standard first-order logic. Prolog uses uppercase letters for variables & lowercase for constants. The execution of Prolog programs is done through depth-first backward chaining, where clauses are tried in the order in which they are written in the knowledge base.

Prolog plays a significant role in artificial intelligence by providing a powerful framework for symbolic computation, logical reasoning & rule-base programming, making it a valuable tool for a wide range of AI applications.

Features of prolog:

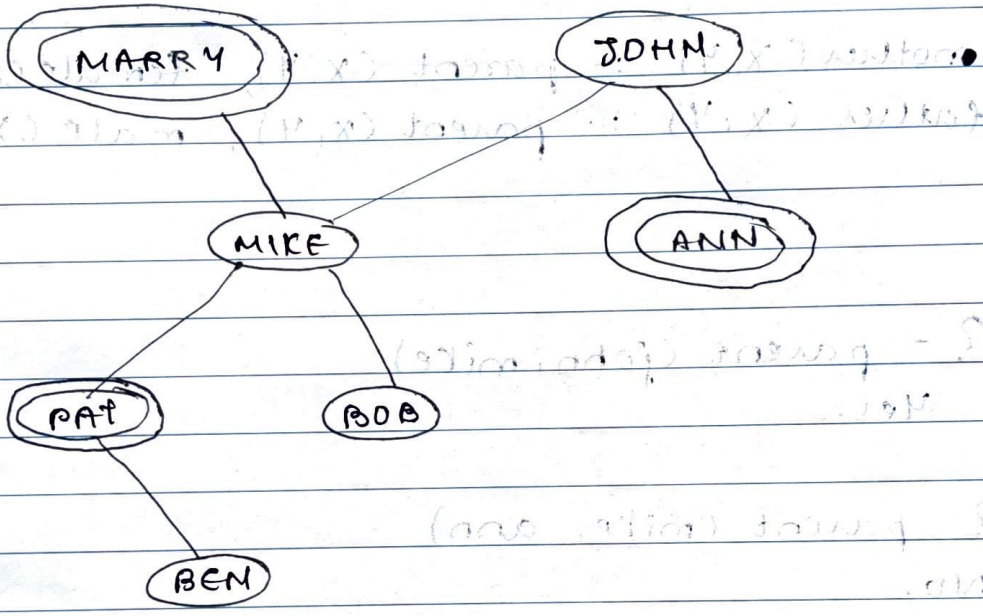
1. Declarative style: prolog programs are written in a declarative style, where the programmer specifies the desired relationships & properties without explicitly detailing the control flow. This allows for concise & expressive code that focuses on what needs to be achieved rather than

how to achieve it.

2. Logic based programming: Prolog is based on first order logic. programs in prolog are composed of predicates, which represent logical relations between entities, & rules which define how these predicates are derived from other predicates or facts.
3. Pattern Matching & Unification: Prolog employs powerful mechanisms for pattern matching & unification. When a query is made, prolog attempts to unify the query with the available rules & facts.
4. Backtracking & Search: Prolog uses a backtracking search strategy to explore the solution space when trying to specify a query. If a branch of the search strategy tree fails to produce a solution, prolog backtracks & explores alternative branches.
5. Built-in list processing: Lists are a fundamental data structure in prolog & they can be manipulated using built in predicates or defined recursively using prolog's expressive syntax. It provides built-in support for list processing, including operations such as appending, splitting & searching lists.

Example :

Smith's family.




DATASET :

```
parent ( john, ann)
parent (john, mike)
parent (mary, mike)
parent (mike, pat)
parent (mike, bob)
parent (pat, ben)

female (mary)
female (ann)
female (pat)
```

male(john)
male(mike)
male(bob)
male(ben)

mother(X, Y) :- parent(X, Y), female(X)
father(X, Y) :- parent(X, Y), male(X)

?- parent(john, mike)
Yes.

?- parent(mike, ann)
No.

? parent(mike, X)
X = pat

? mother(pat, Y)
Y = BEN.

20/03/24