Knowledge Based Agents:-

- The central component of a knowledge-based agent is its knowledge base, or KB.
- A knowledge base is a set of sentences.
- Knowledge Representation Language:-

Each sentence is expressed in a language called a knowledge Representation Language and represents some assertion about the world.

- Inference:-

There is a way to add new sentences to the knowledge base is TELL and to query what is know ASK.

TELL:- A way to add new sentences to the knowledge base.

ASK: A way to query what is known.

Both tasks may involve INFERENCE that is, deriving new sentences from old.

- In LOGICAL AGENTS, inference must obey the fundamental requirement that when one asks a question of the knowledge base, the answer should follow from what has been told to the knowledge base previously.

* Knowledge Based Agent program:-

function KB-AGENT(percept) returns an action

static: KB, a knowledge base

t, a counter, initially 0, indicating time

TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))

action ← ASK(KB, MAKE-ACTION-QUERY(t))

TELL(KB, MAKE-ACTION-SENTENCE(action, t))

t ← t+1

return action.

Like all our agents, it take a percept as input and returns an action.

- The agent maintains a knowledge base, KB, which may initially contain some background knowledge.
- first, it TELLS the knowledge base what it perceives.
- second, it ASK the knowledge base what action it should perform.
- thrid, the agent records its choice with TELL and executes the action.

- MAKE-PERCEPT-SENTENCT:-

takes a percept and a time and return a sentence asserting that the agent perceived the given percept at the given time.

- MAKE-ACTION-QUERY:-

takes a time as input and returns a sentence that asks what action should be done at the correct time.

• MAKE-ACTION-SENTENCE:

constructs a sentence asserting that the chosen action was executed

Knowledge level:-

The knowledge-based agent is not an arbitrary program for calculating actions.

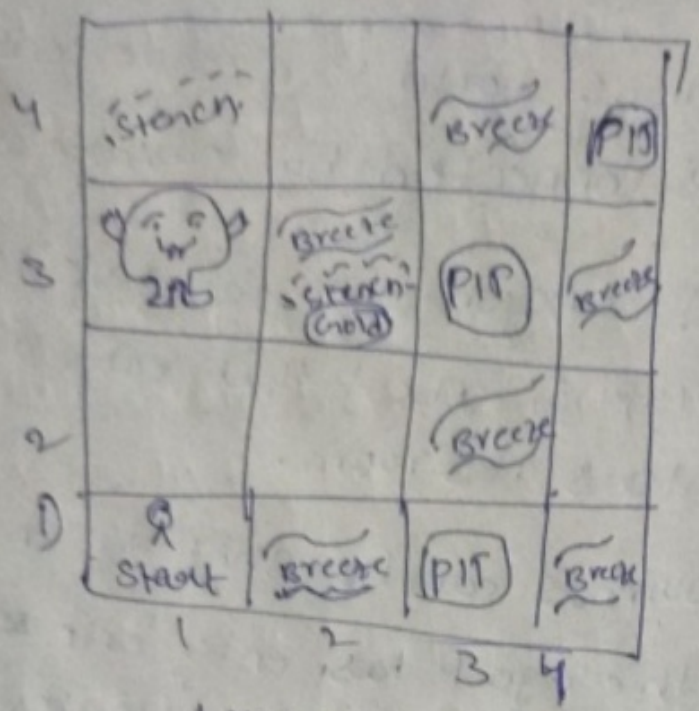
Implementation level:-

—Eg: an automated taxi might have the goal of dropping a passenger to Marin country and might know that it is in San Francisco and that the Golden Gate Bridge is the only link blw the two locations.

(2) The wumpus world environment:-

The wumpus world is a cave consisting of rooms connected by passage ways.

- 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 test bed environment for intelligent agents.
- Michael Genesereth was the first to suggest this.



Wumpus world.

The precise definition of the task environment is given by the 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 and -10 for using up the arrow.
- Environment:- A 4x4 grid of rooms. The agent always starts in the square labeled facing to the right.

• The locations of the gold and the wumpus are chosen randomly, with a uniform distribution; from the squares other than the start square.

Actuators:-

• the agent can move forward, turn left by 90°, or turn right by 90°.

• The agent dies a miserable death miserable death if it enters a square containing a pit or alive wumpus.

Sensors:-

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

1. In the squares di
2. In the squares directly adjacent to a pit, the agent will perceive a breeze

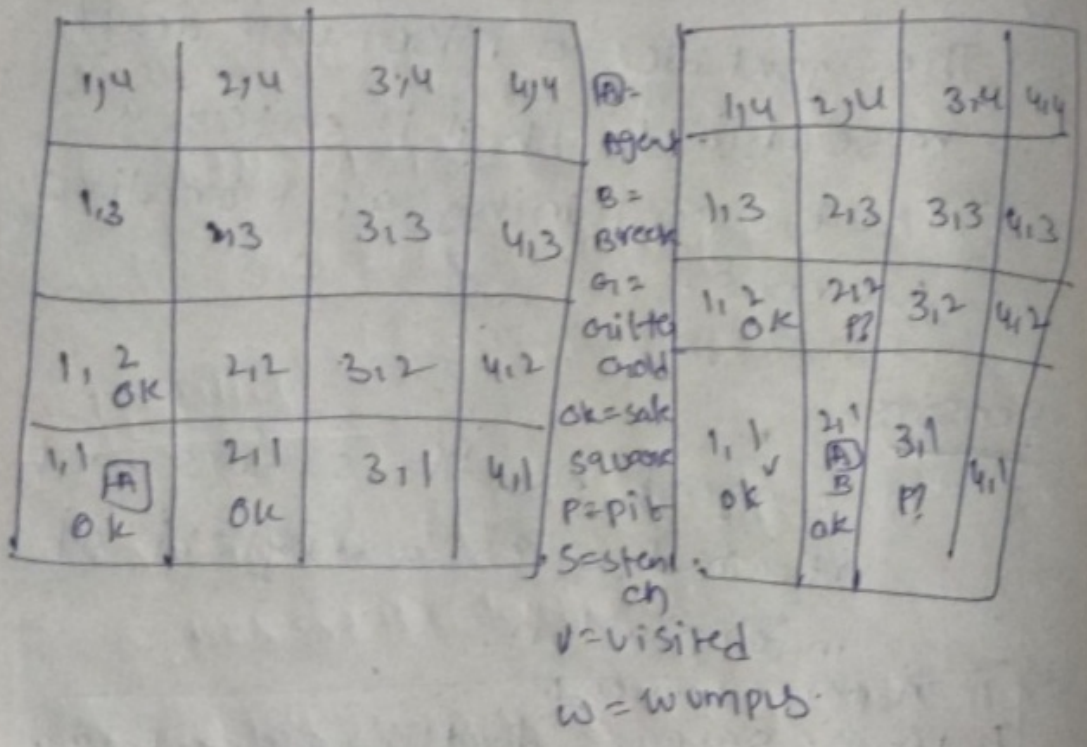
• In the square containing the wumpus and in the directly adjacent square the agent will perceive a stench.

• In the square directly adjacent to pit, the agent will perceive a breeze

3. In the square where the gold is, the agent will perceive a glitter.

4. When an agent walks into a wall, it will perceive a bump.

5. When the wumpus is killed it emits a woeful scream that can be perceived anywhere in the cave.



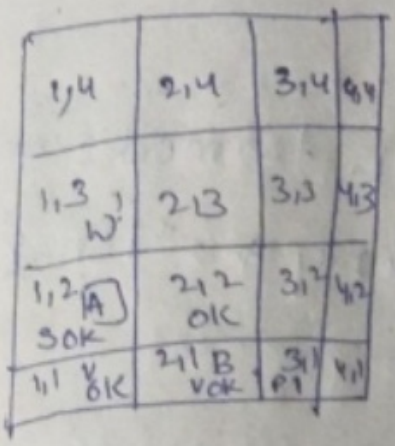
→ The first step taken by the agent in the wumpus world.

- The initial situation, after percept [None, None, None, None, None].

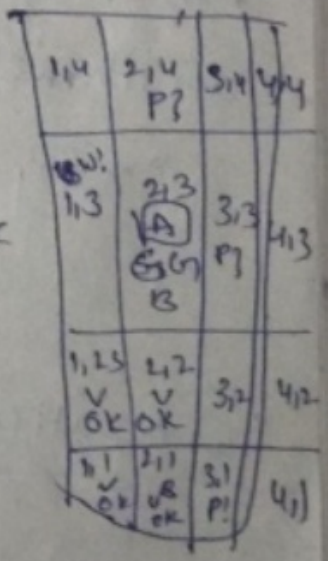
b) After on move, with percept [None, Breeze, None, None, None]

. from the fact that there was no stench or breeze in [1,1], the agent can infer that [1,2] and [2,1] are free of dangers. . They are marked with an ok to indica -te this.

. A cautious agent will move only a square that it knows is OK.



A = agent
B = Breeze
G = Glitter, Gold
OK = safe square
P = pit
S = stench
V = visited
W = Wompus



-Two later stages in the progress of the agent.

(a) ~~After~~ the third move, with percept [stench, None, None, None, None].

(b) After the fifth move, with percept [stench, Breeze, Glitter, None, None].

LOGIC:-

- Knowledge bases consists of sentences.
- These sentences are expressed according to the syntax of the representation language, which specifies will the sentences that are well formed.
- The syntax is clear enough in ordinary arithmetic

"x+y = 4" is a well-formed sentence, whereas "x2y + =" is not.

Semantics:

- A logic must also define the semantics of the language.
- semantics means "meaning" of sentences in logic; the definition is more precise.
- for eg "x+y = 4" is true in a world where x is 2 and y is 2, but false in a world where x is 1 and y is 1.

Entailment:

Entailment means that one thing follows from another.

- Relation of logical entailment b/w sentences is involved that a sentence follows logical from another sentence.

- KB = a knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true

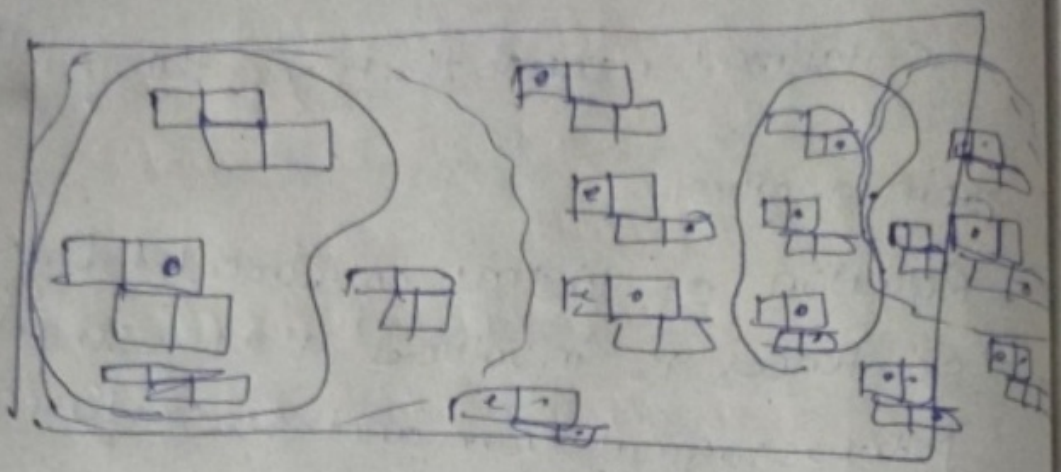
eg: $x+y=4$ entails $4=x+y$.

Model:-

- logicians typically think in terms of models, which are formally structured world with respect to which truth can be valuted.

m is a model of a sentence α if α is true in m.

$M(\alpha)$ is the set of all models of α.



Conclusion:-

- The KB is false in models that contradict what the agent knows - for eg, the KB is false in any model in which [1,2] contains a pit, because there is no breeze in [1,1].
- Now let us consider two possible conclusions:
- α_1 = "There is no pit in [1,2]."
- α_2 = "There is no pit in [2,2]:
- In every model in which KB is true, α_1 is also true.
- Hence, $KB \not\models \alpha_2$: so the agent cannot conclude that there is no pit in [2,2]

5) PROPOSITIONAL LOGIC

. propositional logic is the simplest logic.

. The syntax of propositional logic and its semantics - the truth of sentences is determined.

Syntax - The syntax of propositional logic defines the allowable sentences.

The atomic sentences :- the indivisible syntactic elements consist of a single proposition symbol.

Rules :-

i. Uppercase names are used for symbols (ie) P, Q, R and so on.

ii. Name are Arbitrary.

Complex sentences :-

. Complex sentences are constructed from simpler sentences using logical connections. There five connectives.

- If S is a sentence, ¬S is a sentence (nega
- If S1 and S2 are sentences, S1 ∧ S2 is a sentence (conjunction)
- If S1 and S2 are sentences S1 ∨ S2 is a sentence (disjunction)
- if S1 and S2 are sentences, S1 ⇒ S2 is a sentence (implication).
- if S1 and S2 are sentences S1 ⇔ S2 is a sentence (biconditional / IF AND ONLY IF)

BNF (Backus-Naur form)

- The fig 3.4 shows about the BNF grammar of sentence in propositional logic.

Sentence → Atomic sentence | complex sentence.

Atomic sentence → True | false | symbol

symbol → P | Q | R | ...

complex sentence → ¬ sentence.
(sentence ∧ sentence)

(sentence ∨ sentence)
(sentence ⇒ sentence)
(sentence ⇔ sentence)

Every sentence constructed with binary connectives must be enclosed in paren theses.

- This ensure that we have to write ((¬A∧B) ⇒ C) instead of A∧B⇒C
- order of precedence for the connectives is similar to the precedence used in arithmetic.

for eg:- ab+c is read as ((a b) + c) rather than a(b+c) because multiplication has higher precedence than addition.

- The order of precedence in propositional logic

- The order of precedence in pre positional logic (from highest to lowest) ¬, ∧, ∨, ⇒, ⇔.

- Hence, the sentence $\neg P \vee Q \wedge R \Rightarrow S$ is equivalent to the sentence $((\neg P) \vee (Q \wedge R)) \Rightarrow S$.
- precedence does not resolve ambiguity in sentences such as $A \wedge B \wedge C$, which could be read as $((A \wedge B) \wedge C)$ or as $(A \wedge (B \wedge C))$. These two readings mean the same thing according to the semantics.

Semantics:-

- The semantics defines the rules for determining the truth of a sentence with respect to a particular model.
- In propositional logic, a model simply fixes the truth value, true or false for every proposition symbol.
- for eg:- if the sentences in the knowledge base make use of the proposition symbols $P_{1,2}$, $P_{2,1}$ and $P_{3,1}$ then one possible model is.

$m_1 = \{P_{1,2} = \text{false}, P_{2,2} = \text{false}, P_{3,1} = \text{true}\}$.

Atomic sentences:-

Atomic sentences are easy:

- True is true in every model and false is false in every model.
- The truth value of every other proposition symbol must be specified directly in the model.
- For eg: in the model m_1 gives earlier, $P_{1,2}$ is false.

Complex sentences:-

- for any sentences s and any model m, the sentence s is true in m if and only if s is false in m.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
F	f	T	f	f	T	T
f	T	T	f	T	T	f
f	f	t	f	T	f	f
T	T	f	T	T	T	T

Using the above fig 3.5, truth values of any sentence S can be computed with respect to any model in by a simple process of recursive evaluation.

eg:- $P_{1,2}$ $P_{2,1}$ $P_{3,1}$

f T f

4) FIRST-ORDER LOGIC:-

FOL, a representation language of knowledge which is powerful that propositional logic.

(ie) Boolean logic: FOL is an expressive

- It follow procedural approach
- It derive facts from other facts (i.e) dependent

first-order logic (like Natural Lang) assumes the world contains:-

⇒ Noun - refers to objects.

eg: name, place, thing.

→ verbs refers to relations.
→ some relations are functions.
→ Relation has only one value.

functions have many values assigned to it:-

- object: people, houses, numbers, colors, baseball games, wars.
- properties: red, round, prime, small
- Relations: bigger than, part of, comes blw..
- functions: father of, best-friend, one more than, plus, . . .

eg:-

1. "Evil king John ruled England in 1200"

objects: John, England, 1200; Relation: ruled; properties: evil, king.

2. "one plus two equals three"

object: one, two, three, one pluse two; relation: equals;

function: plus. ("one pluse two" is a name for the object that is

obtained by applying the function "plus" to the objects "one" and "two". Three is another name for this object.)

3. "Squares neighboring the wumpus are smelly"

Object: wumpus, squares; property: smelly; Relation: neighboring.

Ontological commitment:-

- The primary difference blw propositional and first-order logic lies in the ontological commitment made by each language - that is, what it assumes about the nature of reality. Special purpose logic make fastest ontological commitment

Temporal logic:-

It assumes that fast hold at particular times and that those time (intervals) are ordered (arranged).

High order logic:-

It is more expressive that FOL. It allows one to make assertions about all relations.

Epistemological Commitments:-

A logic that allow the possible state of knowledge that it allows with respect to each fact. In first order logic, a sentence represents a fact and the agent belives the sentence to be true, believes it to be false or has no opinion.

ontological Commitment and Epistemological commitment of different logics:-

language	ontological commitment (what exists in the world	epistemological commitment
propositional logic	fact	T/F/unknown
first-order logic	fact, object, relations	T/F/unknown
temporal logic	fact, object, relations, time	T/F/unknown
probability theory	fact	degree of belife [0,1]
fuzzy logic	fact with degree of truth	degree of knowon interval value

5) SITUATION CALCULUS:-

- The situation calculus is a logic formalism designed for representing and reasoning about dynamical domains

  - Mccarthy, Hayes 1969
  - Reiter 1991

- in first-order logic, sentences are either true or false and stay that way. Nothing is corresponding to any sort of change.
- Sit calc represents changing scenarios as a set of SOL formulae.
- A domain is encoded in SOL by three kind of formulae.
  - Action precondition axioms and action effects axioms.
  - successor state axioms, one for each fluent.
  - the foundation axioms of the situation calculus.

- Situation calculus: An Example.

- World:
  - robot
  - items
  - locations(x, y)
  - moves around the world.
  - picks up or drops items.
  - Some items are too heavy for the robot to pick up.
  - Some items are fragile so that they break when they are dropped.
  - robot can repair any broken item that it is holding.

- Actions:-
  - move(x,y): robot is moving to a new location(x,y)
  - pick up(o): robot pick up an object o.
  - drop(o): robot drops the object o that holds

-situations:-

- Initial situation S_0: no actions have yet occurred
- A new situation resulting from the performance of an action a in correct situation S_1 is denoted using the function symbol $do(a, s)$

fluents:- "properties of the world"

- Relational fluents:
  - statements whose truth value may change.
  - They take a situation as a final argument

Action precondition Axioms:-

- Some actions may not be executable in a given situation
- $poss(a, s)$: special binary predicate

Action-effect Axioms

- specify the effect of an action on the fluents
- examples:-
- poss (pickup(o), s) → is-carrying (o, do(pickup(o), s))
- poss (drop(o), s) ∧ fragile(o) → broken (o, do(drop(o), s)).
- Is that enough? No, because of the frame problems.

8) Boliding A knowledge Base:-

Knowledge engineering projects vary widely in content, scope, and difficulty, but all such projects include the following steps:-

1. Identify the task:-

The knowledge engineer must delineate (ie) describe the range of questions that the knowledge base will support and the kinds of facts that will be available for each problem instance

2. Assemble the relevant knowledge:-

the knowledge engineer might already be an expert in the domain to extract what this knowledge.

3. decide on a vocabulary of predicates, functions and constants:-

Translate the important domain-level concepts into logic-level names

4. Encode general knowledge about the domain

The Knowledge engineer writes down that axioms for all the vocabulary terms (ie) enabling the expert to check the content.

5. Encode a description of the specific problem instance:

It will involve writing simple atomic sentences about instances of concepts that are already part of the ontology.

6. Pose queries to the inference procedure and get answers:-

we can let the inference procedure operate on the axioms and problem-specific facts to derive

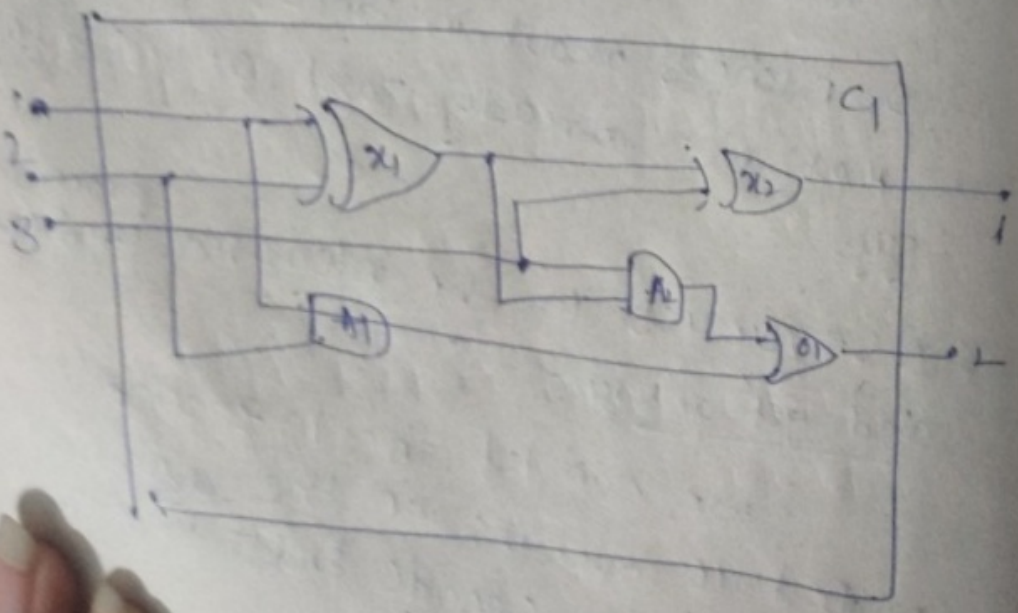
the facts we are interested in knowl

7. Debug the knowledge base:-

To understand this seven-step process better, we now apply it to an extended example the domain of electronic circuits.

4) The electronic circuits Domain:-

we will develop an ontology and knowledge base that allow us to reason about digital circuits of the kind shown.



the first two inputs are the two bits to be added and the third input is a carry bit.

The circuit contains two XOR gates, two AND gates and one OR gate.

We follow the seven-step process for knowledge engineering.

1. Identify the task:-

- Does the circuit actually add properly? (circuit verification).

2. Assemble the relevant knowledge:-

- composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
- irrelevant: size, shape, color, cost of gates.

3. Decide on a vocabulary:-

Type(X_1) = XOR

Type(X_1, XOR)

XOR(X_1)

4. Encode general knowledge of the dom

- $\forall t1, t2$ connected $(t1, t2) \Rightarrow$ signal $(t1) =$ signal $(t2)$
- $\forall t$ signal $(t) = 1 \vee$ signal $(t) = 0$
- $1 \neq 0$
- $\forall t_1, t_2$ connected $(t_1, t_2) \Leftrightarrow$ connected (t_2, t_1)

- $\forall g$ Type$(g) =$ OR $\Rightarrow$ signal $(out(1, g)) = 1 \Leftrightarrow \exists n$ signal $(in(n, g)) =$

- $\forall$ Type$(g) =$ AND $\Rightarrow$ signal $(out(1, g)) = 0 \Leftrightarrow \exists n$ signal $(in(n, g)) = 0$

- $\forall$ Type$(g) =$ XOR $\Rightarrow$ signal $(out(1, g)) = 1 \Leftrightarrow$ signal $(in(1, g)) \neq$ signal $(in(2, g$

- $\forall$ Type$(g) =$ NOT $\Rightarrow$ signal $(out(1, g)) \neq$ signal $(in(1, g))$

5. Encode the specific problem instance.

- Type(X1) : XOR Types(X2) = XOR
- Type (A1) = AND Types(A2) = AND
- Type(O1) = OR.

6. Pose queries to the inference procedure:

what are the possible sets of values of all the terminals for the adder circuit?

∃ i1, i2, i3, o1, o2 signal(in(1,C1)) = i1 ∧ signal(in(2,C1)) = i2 ∧ signal (in(3,C1) = i3 ∧ signal(out(1,C1)) = 0 ∧ signal(out(2,C1)) = 1.

7. Debug the Knowledge base may have omitted assertion like 1≠0

suddenly, the system will be unable to prove any outputs for the circuit, except for input cases 000 and 110.

8) Truth maintenance System:-

Many of the interences drawn by a knowledge representation system will have only default status, rather that being absolutely certain.

- This process is called belief revision
- Truth maintenance systems (TMS) have been developed as a means of implementing Non-Monotonic Reasoning systems.

Basically TMS:-

- all do some form of dependency directed backtracking.
- assertions are connected a via a network of dependencies.

Justification-Based Truth Maintenance System (JTMS)

. This is a simple JTMS in that it does not know anything about the structure of assertion themselves.

- AN IN-LIST - which support beliefs held.
- AN OUT-LIST - which support beliefs not held.

Logic-Based (LTMS)

Similar to JTMS except.

- Node assume no relationship among them except ones explicitly stated in justifications.
- If this happens nlw has to be reconstructed.

Assumption-Based TMS (ATMS)

- JTMS and LTMS pursue a single line of reasoning at a time and back track when needed - depth first search.

- ATMs maintain alternative paths in parallel -- breadth - first search
- Backtracking is avoided at the expense of maintaining multiple contexts

7) RESOLUTION:-

In 1930, the German mathematician kurt Godel proved the first completeness theorem for first-order logic showing that any entailed sentence has a finite proof.

- The theorem states that a logical system that includes the principle of induction without which very little of discrete mathematics can be constructed - is necessarily incomplete.

- Resolution-based theorem provers have been applied widely to derive mathematical theorems, including several for which no proof was known previously.
  - Conjunctive normal form for first-order logic.
  - The resolution inference rule.
  - completeness of Resolution.
  - Dealing with equality.

- Resolution Strategies.
- Theorem provers.

Conjunctive normal form for first-order logic:-

First-order resolution requires that sentences be in conjunctive normal form (CNF) - that is, a conjunction of clauses, where each clauses is a disjunction of literals.

$\forall x$ [American(x) $\wedge$ weapon(y)) $\wedge$ sells (x, y, z) $\wedge$ hostile(z)] $\Rightarrow$ criminal(x)

Becomes in CNF

$\neg$ [American(x) $\vee$ $\neg$ weapon(y)] $\vee$ $\neg$ sells (x, y, z) $\vee$ $\neg$ hostile(z)] $\vee$ criminal(x)

1. Eliminate biconditionals and implications:

$\forall x$ [$\neg$ $\forall$ y $\neg$ Animal(y) $\vee$ loves(x, y)] $\vee$ [$\exists$ y loves(y, x)]

2. Move ¬ inwards :-

¬∀x p becomes ∃x ¬p

¬∃x p becomes ∀x ¬p

The transforms are :-

∀x [∃y ¬[¬Animal(y) ∨ Loves(x,y)]] ∨
[¬∃y Loves(y,x)]

∀x (∃y ¬[¬Animal(y) ∧ ¬Loves(x,y)] ∨
[∃y Loves(y,x)]

∀x [∃y Animal(y) ∧ ¬Loves(x,y)] ∨
[∃y Loves(y,x)].

3. Standardize variables :-

For sentences like (∀x P(x)) ∨ (∃x Q(x)) which use the same variable name twice, change the name of one of the variables.

∀x [∃y Animal(y) ∧ ¬Loves(x,y)]
∨ [∃z Loves(z,x)]

4. Skolemize:-

A more general form of existential instantiation.

$\forall x[\text{Animal}(A) \wedge \neg \text{Loves}(x, A)] \vee \text{Loves}(B, x)$

5. Drop universal quantifiers:-

At this point, all remaining variables must be universally quantified.

$\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$

6. Distribute $\wedge$ over $\vee$:-

$[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge$

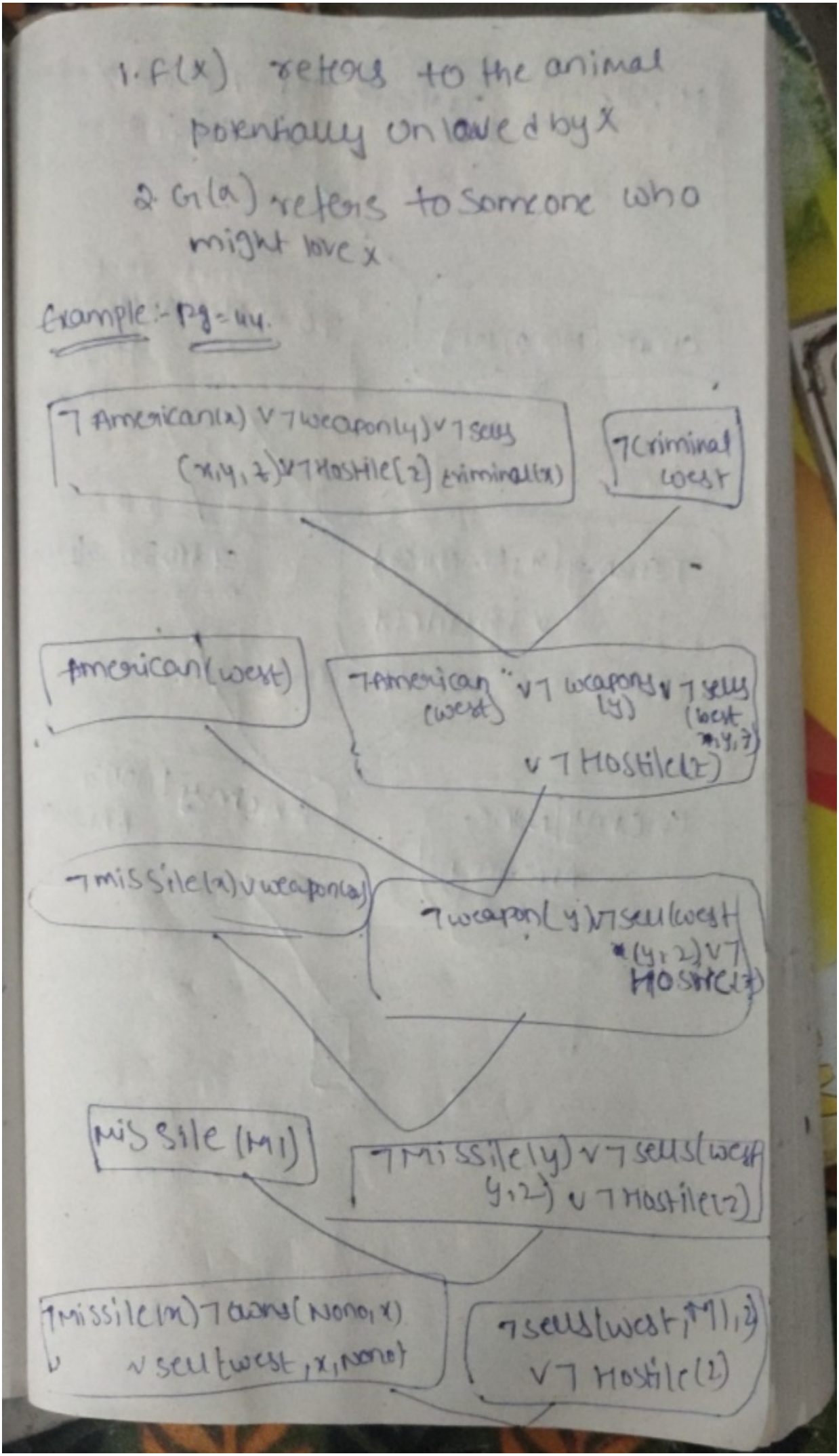
$[\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$

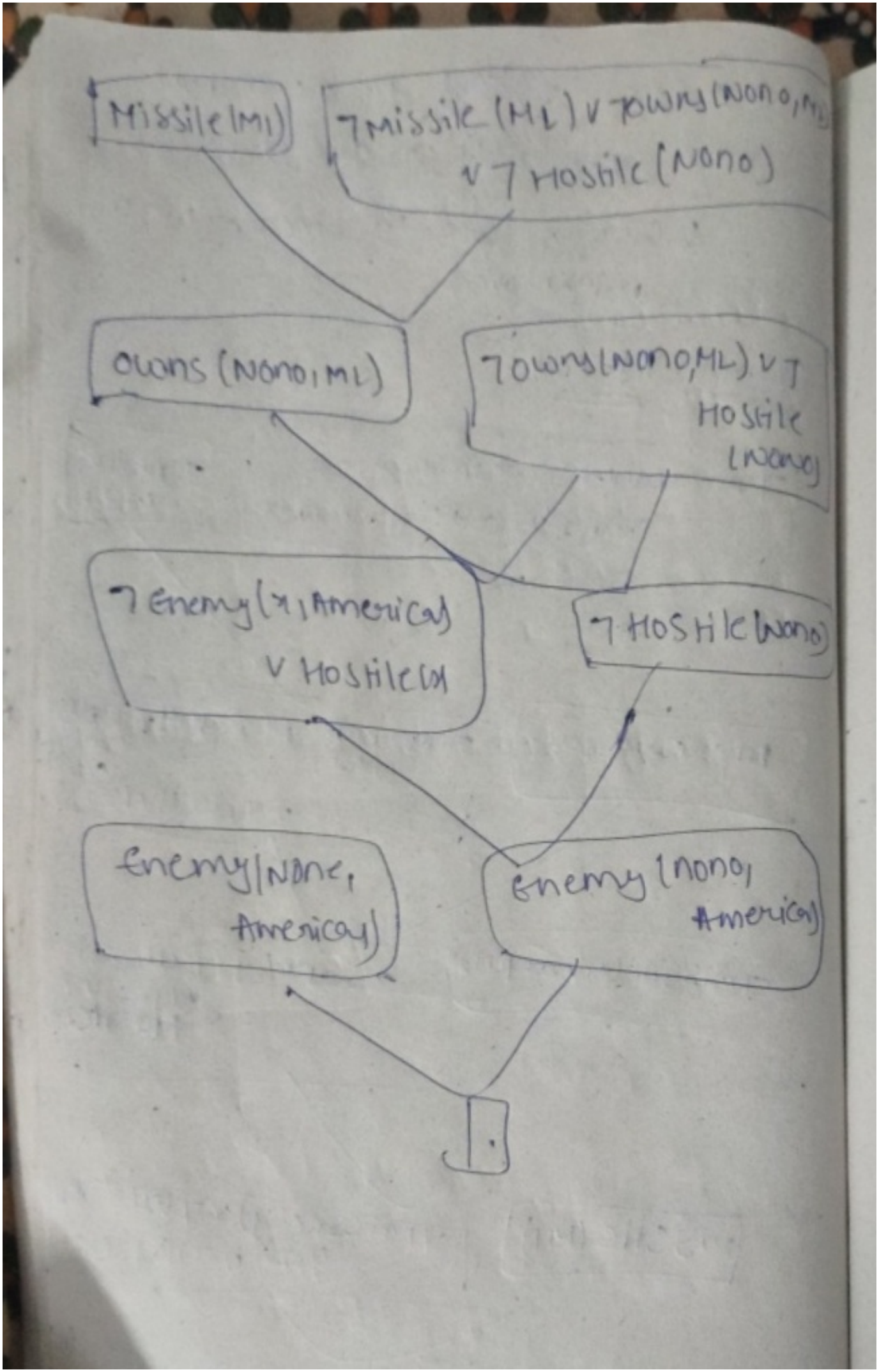
This step may also require flattening out nested conjunctions and disjunctions.

1. F(x) refers to the animal potentially owned by x

2. G(a) refers to someone who might have x.

Example :- pg - 44.

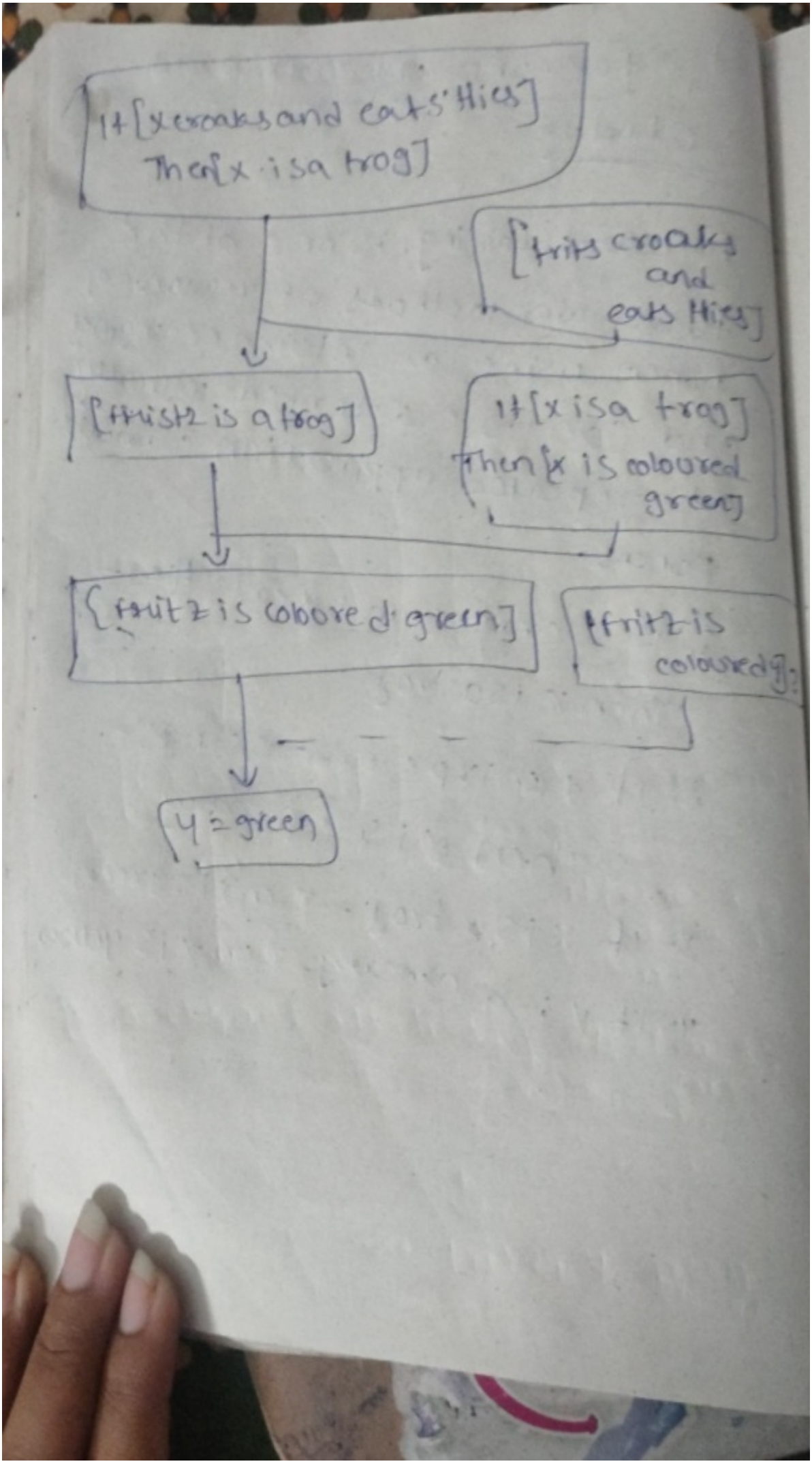


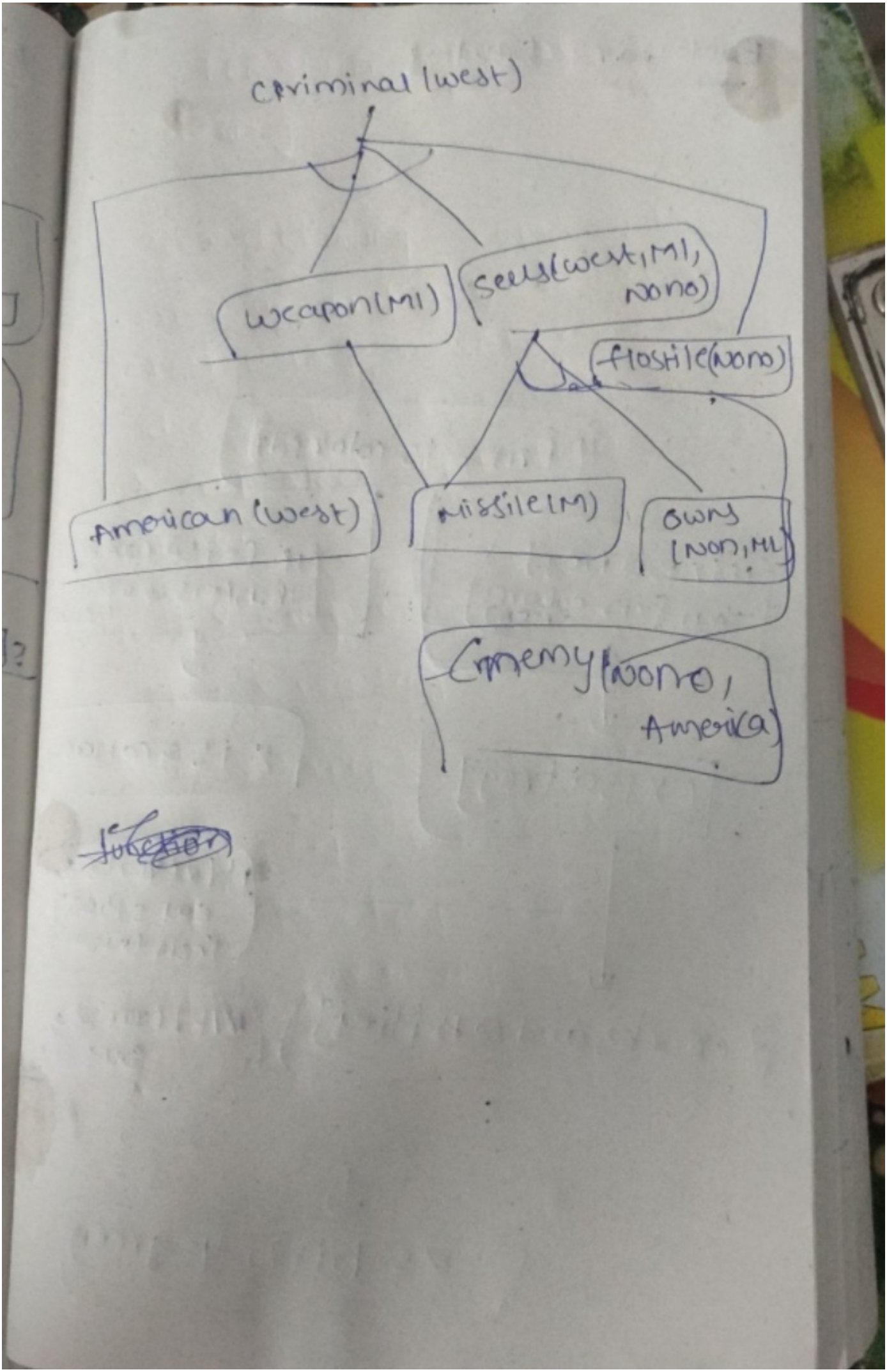

Missile(M1)
¬Missile(ML) ∨ Owns(NONO,ML)
∨ Hostile(NONO)
Owns(NONO,ML)
¬Owns(NONO,ML) ∨
Hostile
(NONO)
¬Enemy(x,America)
∨ Hostile(x)
¬Hostile(NONO)
Enemy(NONE,
America)
Enemy(NONO,
America)

18) FORWARD AND BACKWARD Chaining

Forward chaining is one of the two main methods of reasoning when using an inference engine and can be described logically as repeated application of modus ponens.

1. If X croaks and X eats flies - Then X is a frog.
2. If X chirps and X sings - Then X is a canary
3. If X is a frog - Then X is green
4. If X is a canary - Then X is yellow


If [x croaks and eats flies]
Then [x is a frog]
[fritz croaks and eats flies]
[fritz is a frog]
If [x is a frog]
Then [x is coloured green]
[fritz is coloured green]
[fritz is coloured]
y = green


Criminal (west)
weapon(M1)
sells(west, M1, Nono)
Hostile(Nono)
American (west)
Missile(M1)
owns (Non), M1
Enemy (Nono, America)

Backward Chaining