Knowledge Representation

Here, the representation of knowledge and reasoning process are central to entire field of Artificial Intelligence.

→ These two also play a crucial role in dealing with partially observable environments

Knowledge Based Agents :

The central Component of knowledge-based Agent is its knowledge-base. [KB] = Knowledge level / Logical level / Implementation level

→ Intelligent Agents need knowledge about the world to choose good actions.

→ Knowledge base is a set of "sentences" and each sentence is expressed in a Language called "knowledge representation Language" that represents some assertion about the world.

It is Composed of 2 components

i.e, (1) knowledge base [domain Content specific]

(2) Inference mechanism [domain independent algorithm]

→ These agents must be able to;

- Represent states, actions etc...
- Incorporate new percepts.
- Update internal representations of world.
- Deduce hidden properties of world
- Deduce appropriate actions.

⟹ To build an agent there is a way that,

i) add new sentences to knowledge base
i.e, the standard name Tell..
[what it needs to know]

ii) A way to query what is known Ask
[what to do]

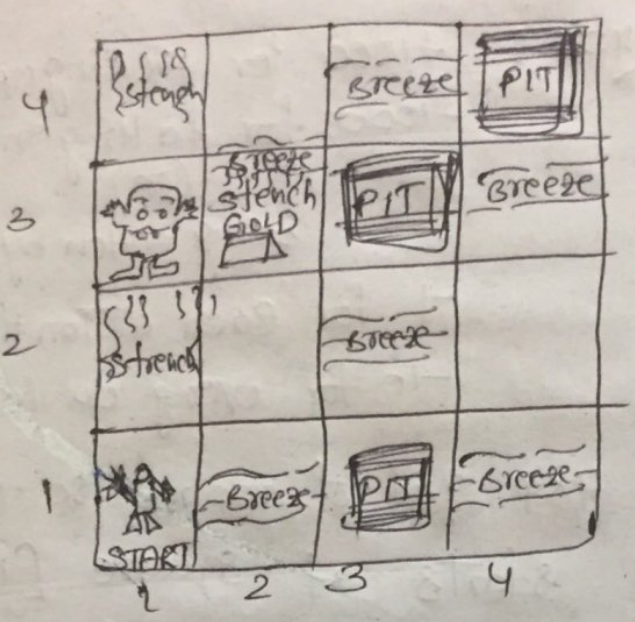
→ Both tasks may involve inference i.e, deriving new sentences from old.

---

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

Wumpus world :



Breeze → easy to achieve

It's a computer game.

→ Here the agent explores a cave consisting of rooms connected by passageways

→ Lurking somewhere in the cave is the wumpus, it's a beast/monster that eats any agent that enters the room

→ Some rooms contains bottomless pits that trap any agent that enters into the room.

→ Occasionally there is a room with gold and the goal is;

- → Collect the gold
- → exit the world
- → without being eaten

PEAS Description for wumpus:

Performance measure: +1000 for picking up gold
-1000 for falling in to pit
(or)
being eaten by wumpus
-1 for each action taken
-10 for using up the arrow

Environment: A 4X4 grid of rooms. so the agent always starts at square [1, 1] facing to right. the Locations of gold and wumpus are chosen randomly, with uniform distribution, from ~~start~~ squares other than start square. in addition to this the squares except start one remaining can be a pit with probability 0.2.

Actuators: Agent can move forward,
Turn left by 90°
(or)
turn Right by 90°
Grab used to pick up an object in the same square
Shoot used to fire an arrow in straight line
if continues to hit a wumpus/wall

Sensors: Five sensors are used.

a) The square which contains wumpus and (not diagonally) in the directly adjacent squares the agent will perceive stench.

b) squares directly adjacent to a pit, the agent will reciere Breeze.

c) in the square where the gold is, the agent will recieve glitter.

d) when an agent walks into a wall it will perceive a bump.

e) when the wumpus is killed, it emits a woeful scream that can be perceived any where in the cave.

So, for Example the percept will be given as five symbol list:

[i.e, Assume there is stench, Breeze but no glitter, no bump, and no Scream]

⇒ [stench, Breeze, None, None, None]

[Note: Agent can't perceive its own Location]

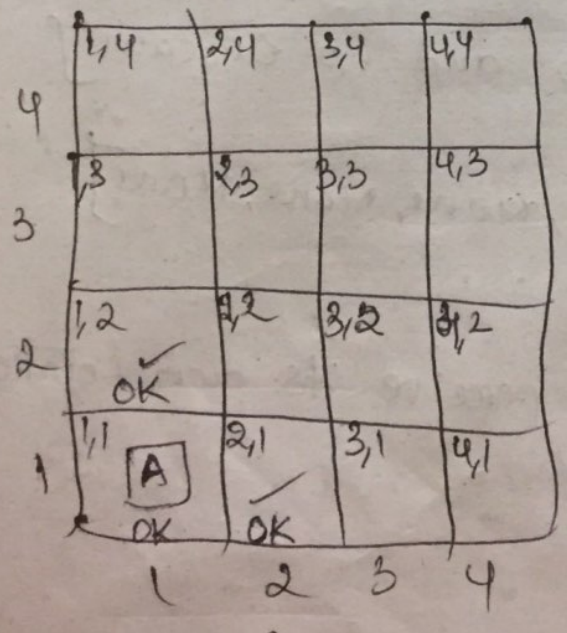
Let us Consider a knowledge-based wompos agent exploring the environment as given in above figure.

→ from start it is in [1,1] and it is safe square.

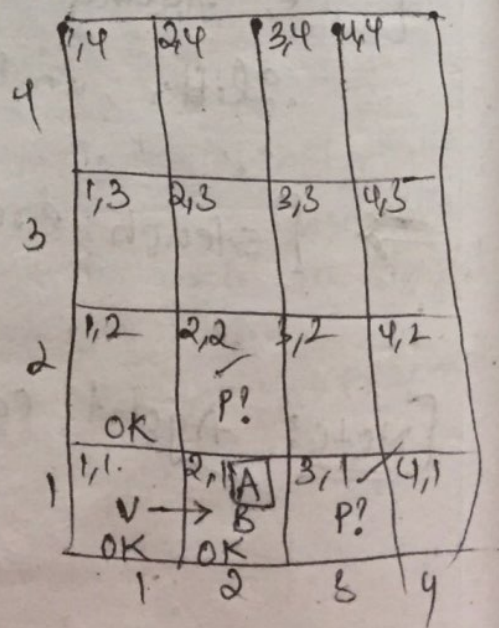
→ when his knowledge evolves as per new percepts received and actions taken.

→ for example the first percept recieved as [none, none, none, none, none]
i.e; the agent can understood that neighboring squares are safe

→ so we list some of the sentences in the knowledge Base using letters B (breezy), OK for (safe, neither pit nor wumpus)



(a)



(b)

[A] - Agent
B - Breeze
G - Glitter, Gold
OK - safe square
P - pit
S - stench
V - visited
W - wumpus.

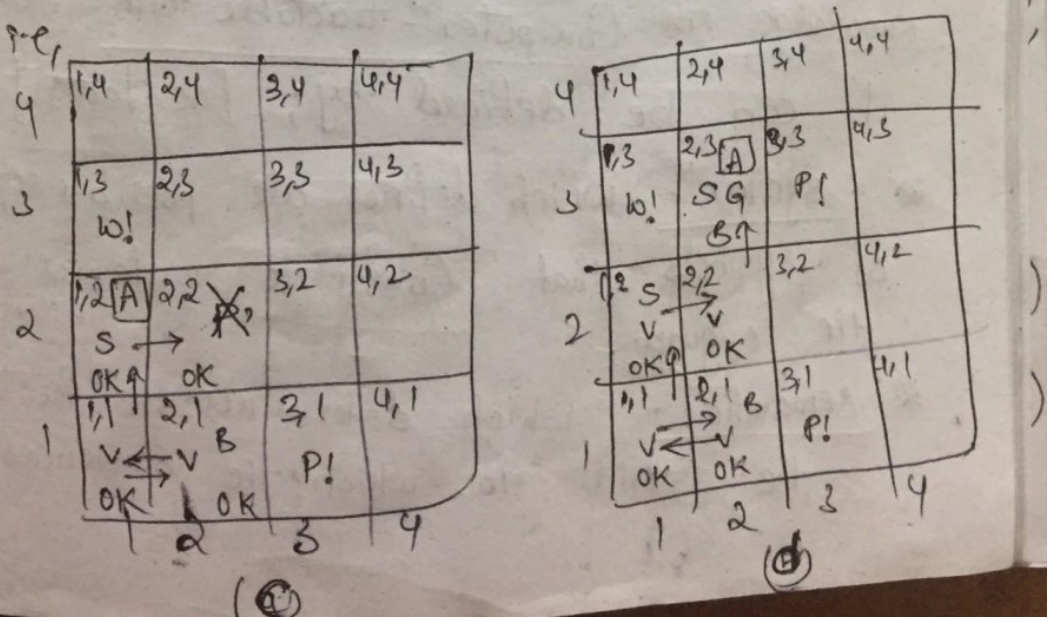
→ [1,2] [2,1] are OK, so agent decided to move forward to [2,1] shown in fig(b)

→ He detects breeze in [2,1] so there must be a pit in neighboring square. i.e, it may be in [2,2] (or) [3,1] (or) both

→ In this case there is only one known square with OK so he go back to visited square [1,1] and proceed to [1,2]

→ New percept in [1,2] is

[Stench, None, None, None, None]



→ After third move with percept

[Stench, None, None, None, None]

→ After fifth move with percept

[Stench, Breeze, Glitter, None, None]

Here, in each case where the agent draws a conclusion from the available information that conclusion is guaranteed to be correct if the available information is correct, this is a fundamental property of logical reasoning.

Logic:

It provides the fundamental concepts of Logical representation and Reasoning.

→ Here the agent should express his knowledge in Computer-tractable form. i.e., it can be defined by; [sentences]

* syntax, which defines all possible sequences of symbols that constitute sentences of the language.
* semantics, which determines the facts in the world to which the sentences refer.

"$x+y=4$" → a sentence but not "$x\,2y+=$"

→ semantics is the meaning of sentences

→ The semantics of the language defines the truth of each sentence with respect to each possible world.

i.e, "$x+y=4$" is true where x and $y=2$
but false where x and $y=1$

[Note: The possible models are just all possible assignments of numbers to the variables. x and y].

Entailment: · logical reasoning involves the relation of logical entailment between sentences.

i.e; A sentence follows logically from another sentence. It can be represented as

$$\alpha \models \beta$$

i.e, $\alpha \models \beta$ if and only if in every model in which α is true, β is also true.

i.e, if 'α' is true then β must be true

$\Rightarrow$ Truth of β "Contained" in truth of α.

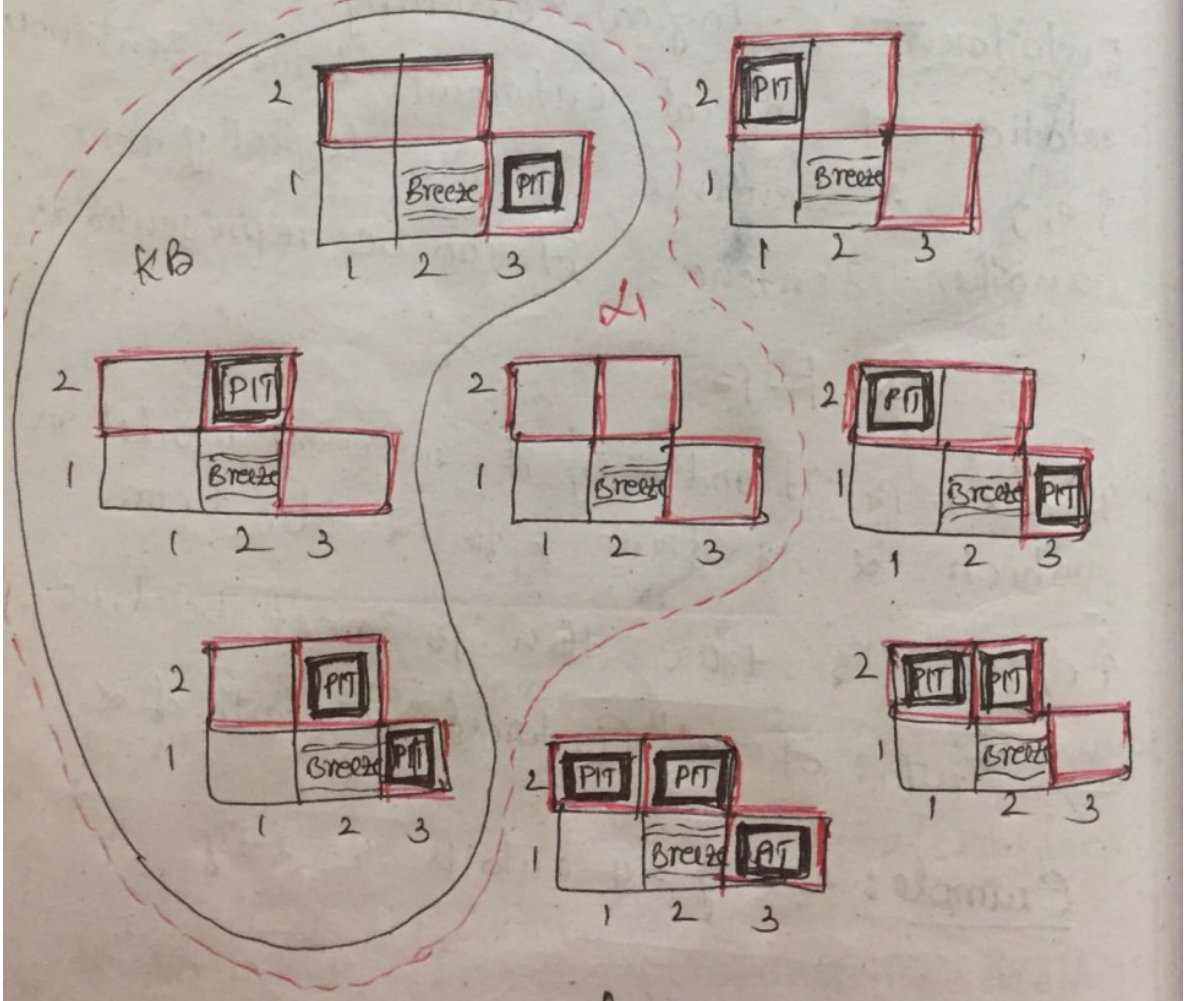
Example: $x+y=4$ entails $4=x+y$

Here if we Consider the analysis to wumpus-world

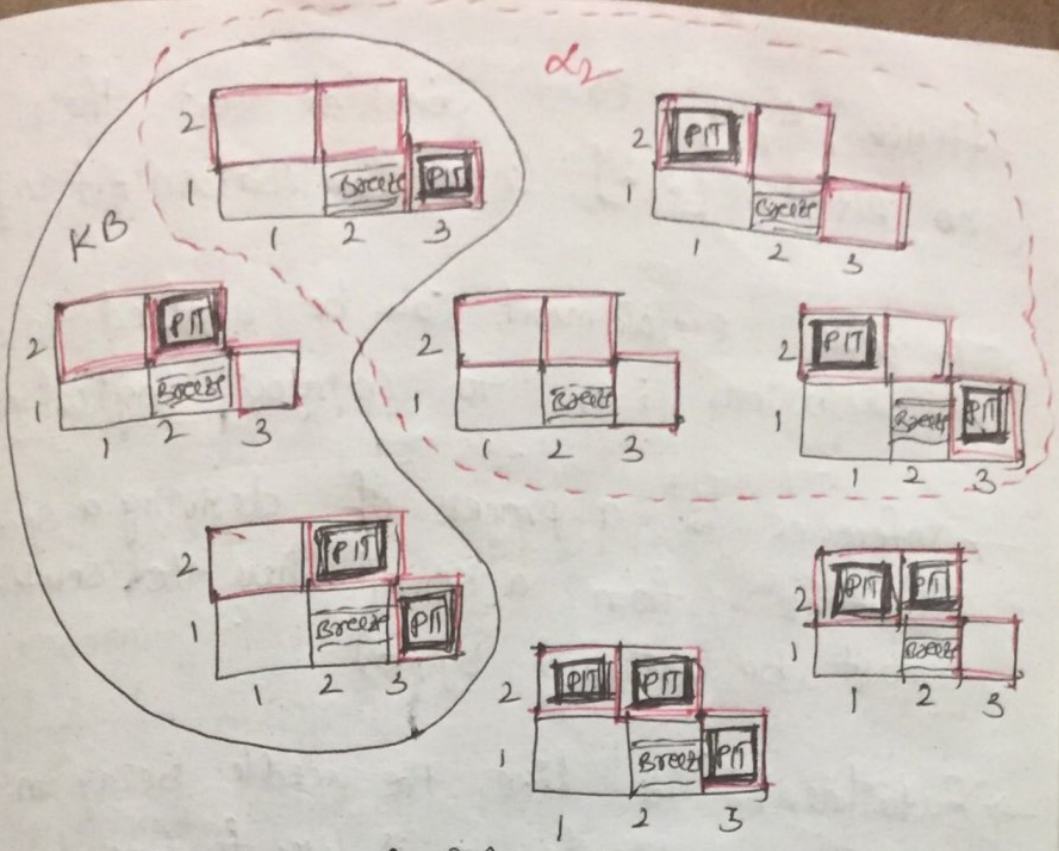
Consider situation in fig(b) the agent has detected nothing in [1,1] and a breeze in [2,1]

- There percepts Combined with the agent's knowledge of rules of wumpus world, constitute the KB.

→ The agent is interested in adjacent squares [1,2] [2,2] and [3,1] which might Con might not Contain pits. so, there are $2^3 = 8$ possible models. i.e,



fig(a)



fig(b)

→ The KB is false in models that contradict what the agent knows.

for Ex: KB fails / false in any model in which [1,2] contains a pit. because there is no breeze in [1,1]

→ So by considering the above possibilities we can have 2 conclusions:

α_1 = "There is no pit in [1,2]"

α_2 = "There is no pit in [2,2]"

from this, in every model, KB is true then α_1 is also true.

Hence $KB \models \alpha_1$ is true

→ In some models KB is true but α_2 is false

Hence $KB \not\models \alpha_2$

Hence, Agent Can't Conclude that there is no pit in [2,2] (nor) Can Confirm pit in [2,2]

→ Here entailment can be applied to derive Conclusions i.e, to carryout logical inference

→ Inference is a process of deriving a specific sentence from a KB. (where the sentence must be entailed by KB)

→ Entailment is like the needle being in the haystack and inference is like finding it

i.e, KB is haystack
α is needle
i is inference then,

$$KB \vdash_i \alpha$$

i.e, α is derived from KB by i.

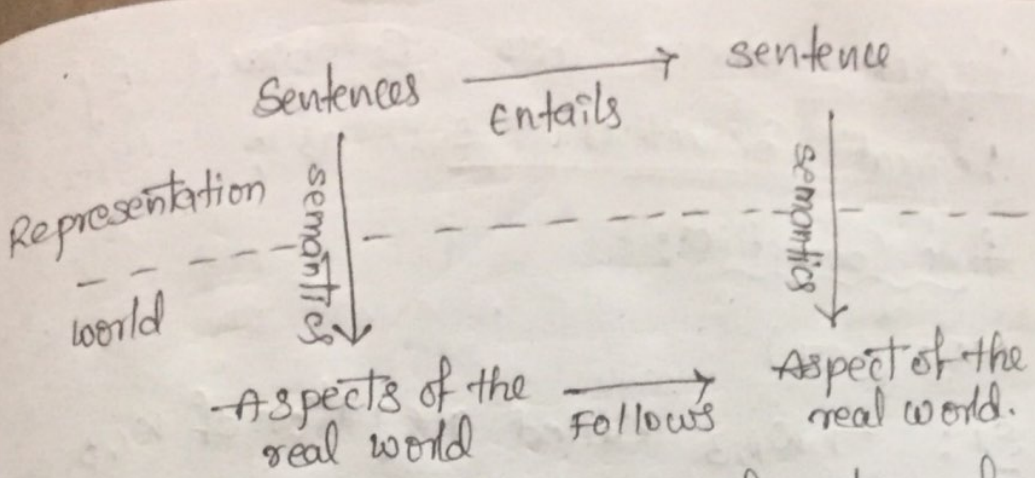
Soundness: if 'i' is sound means

KB $\vdash_i$ α is true, KB $\models$ α is true

Completeness: if 'i' is Complete. means

KB $\models$ α is true, KB $\vdash_i$ α is true.

If KB is true in the real world, then any sentence α derived from KB by a sound inference procedure is also true in the real world.



→ sentences are physical Configurations of agent
Reasoning is a process of Constructing new
physical Configurations from old ones.

→ Logical reasoning should ensure that new
Configurations represent aspects of world
that actually follow from the aspects of
old configurations.

Propositional Logic

It's a simple Logic on which the truth
of the sentences is determined.

→ The syntax of propositional logic defines
atomic sentences Consists of single proposition
symbol.

→ Each such symbol stands for proposition
that can be True (or) False.

→ propositional symbols are P, Q, R and so on.
Can be represented as $W_{1,3}$ is proposition
that the wumpus is in [1, 3]

→ Complex sentences are combined by using Logical Connectives. such as;

five Connectives:

1) Not ($\neg$)

Ex: $\neg W_{1,3}$ called Negation of $W_{1,3}$.

2) $\wedge$ (and)

Ex: $W_{1,3} \wedge P_{3,1}$ is called Conjonction

3) OR ($\vee$)

Ex: $W_{1,3} \vee P_{3,1}$ is called disjunction

4) Implication ($\Rightarrow$)

Ex: $(W_{1,3} \wedge P_{3,1}) \Rightarrow \neg W_{2,2}$

This symbol is Sometimes written as $\supset$ or $\rightarrow$

5) Biconditional ($\Leftrightarrow$) [if and only if]

Ex: $W_{1,3} \Leftrightarrow \neg W_{2,2}$ is a biconditiond

→ Formal Grammar of propositional Logic; with BNF notation:

Sentence → Atomic Sentences | Complex Sentence

Atomic Sentence → True | False | symbol

symbol → P | Q | R | ...

Complex sentence → $\neg$ sentence
| (sentence $\wedge$ Sentence)
| (sentence $\vee$ Sentence)
| (sentence $\Rightarrow$ Sentence)
| (sentence $\Leftrightarrow$ Sentence)

⇒ Every sentences Constructed with binary Connectives must be enclosed with parantheses

i.e, for example: $((A \wedge B) \Rightarrow C)$ instead of $A \wedge B \Rightarrow C$

⇒ To improve readability, we will omit parentheses in some cases.

for ex: $ab+c$ is read as $((ab)+c)$ ✓
but not $a(b+c)$ ✗

Because multiplication has higher precedence than addition.

⇒ In propositional Logic the order of precedence is highest to lowest i.e, $\neg$, $\wedge$, $\vee$, $\Rightarrow$, and $\Leftrightarrow$

for example: $\neg P \vee Q \wedge R \Rightarrow S$

is equal to

$((\neg P) \vee (Q \wedge R)) \Rightarrow S$

⇒ precedence doesn't resolve ambiguity in sentences such as $A \wedge B \wedge C$ i.e, it may be $((A \wedge B) \wedge C)$

(or)

$(A \wedge (B \wedge C))$

Semantics: Defines the rules for determining the truth of a sentence with respect to a particular model.

In propositional logic, a model fixes the truth value to TRUE (or) FALSE

forex: If wumpus world proposition symbols are $P_{1,2}$, $P_{2,2}$, $P_{3,1}$ then one possible model is

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

Truth-table: It specifies the truth value of any sentence s, that can be computed with respect to any model m by a simple process of recursive evaluation.

for ex: A sentence $\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1})$ evaluated on m_1, gives true $\wedge$ (false $\vee$ true) = true $\wedge$ true = true.

⇒ Truth table for five logical Connectives is;

P	Q	¬P	P∧Q	P∨Q	P⇒Q	P⇔Q
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

$P \Leftrightarrow Q$ is True iff $P \Rightarrow Q$ & $Q \Rightarrow P$ are True

forexample: A square is breezy if and only if a neighboring square has a pit.

i.e, $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

A simple Knowledge Base:

we can Construct a knowledge base for the wompus world. for that;

→ first we need to choose our vocabolary of proposition symbols.

→ For each i, j;

Let $P_{i,j}$ be true if there is a pit in $[i,j]$

let $B_{i,j}$ be true if there is a breeze in $[i,j]$

→ knowledge Base includes the following sentences,

i.e;

1) There is no pit in $[1,1]$

$R_1 : \neg P_{1,1}$

2) A square is breezy iff there is a pit in neighboring square. This has to be stated for each square. Here we considered relevant squares.

$R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$.

$R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

3) The following sequences are True

$R_4 : \neg B_{1,1}$

$R_5 : B_{2,1}$

Inference :

As we know the aim of Logical inference is to decide whether $KB \models \alpha$ for some sentence α.

$\Rightarrow$ For propositional Logic, models are assignments of true or false to every proposition symbol.

$\Rightarrow$ Consider wumpus-example;

Here, the relevant propositions are;

$B_{1,1}$, $B_{2,1}$, $P_{1,1}$ $P_{1,2}$ $P_{2,1}$ $P_{2,2}$ and $P_{3,1}$

With these 7 symbols there are $2^7 = 128$ possible models. In these 3 models $\neg P_{1,2}$ is True.

Hence, no pit on [1, 2].

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
F	F	F	F	F	F	F	T	T	T	T	F	F
F	F	F	F	F	F	T	T	T	F	T	F	F
:	:	:	:	:	:	:	:	:	:	:	:	:
F	T	F	F	F	F	F	T	T	F	T	T	F
F	T	F	F	F	T	F	T	T	T	T	T	T
F	T	F	F	F	T	T	T	T	T	T	T	T
:	:	:	:	:	:	:	:	:	:	:	:	:
T	T	T	T	T	T	T	F	T	T	F	F	F

Function TT-ENTAILS ? (KB, α) returns true or false

inputs: KB, the knowledge base, a sentence in propositional logic

α, the query, a sentence in propositional logic

symbols ← a list of proposition symbols in KB and α

return TT-CHECK-ALL (KB, α, symbols, [ ])

---

function TT-CHECK-ALL (KB, α, symbols, model)
returns true or false

if EMPTY ? (symbols) then

if PL-TRUE ? (KB, model) then return PL-TRUE? (α, model)

else return true.

else do.

P ← FIRST (symbols); rest ← REST (symbols)

return TT-CHECK-ALL (KB, α, rest, EXTEND (P, true, model) and TT-CHECK-ALL (KB, α, rest, EXTEND (P, false, model).

It's a truth table enumeration Algorithm for deciding propositional entailment.

→ This Algorithm is sound, because it implements directly the definition of entailment, and it is complete, because it works for any KB and α.

→ The time complexity of the Algorithm is $O(2^n)$

Equivalence, validity, and satisfiability:

① Logical equivalence:

Two sentences are logically equivalent iff they are true in same set of models.

for Ex: α and β are 2 sentences then,

$\alpha \Leftrightarrow \beta$. | $\alpha \equiv \beta$ iff $\begin{cases} \alpha \models \beta \\ \beta \models \alpha \end{cases}$

By using Truth table we can easily show that $P \wedge Q$ and $Q \wedge P$ are logically equivalent

The following are some standard Logical equivalences α, β, γ are sentences of PL.

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ Commutativity of $\wedge$

$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ Commutativity of $\vee$

$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$ associativity of $\wedge$

$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ associativity of $\vee$

$\neg(\neg\alpha) \equiv \alpha$ double negation elimination

$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ Contraposition

$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ Bidirectional Elimination

$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de Morgan

$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ de Morgan

$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of $\wedge$ over $\vee$

$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of $\vee$ over $\wedge$

② validity: A sentence is valid if it is true in all models.

forex: $P \vee \neg P$ is a sentence valid one

i.e,

P	¬P	P∨¬P	
T	F	T	✓ valid
T	F	T	
T	F	T	
F	T	T	
F	T	T	

these valid sentences are also called as "Tautologies"

i.e, they are necessarily True.

→ This validity can be proved by entailment using "deduction theorem".

i.e, For any sentences α and β

$\alpha \models \beta$ if and only if the sentence $(\alpha \Rightarrow \beta)$ is valid.

③ satisfiability

A sentence is satisfiable if it is true in some model.

forex: if a sentence α is true in a model m then we say that m satisfies α.

(or)

m is a model of α.

→ satisfiability is checked by enumerating possible models until one is found and that which satisfies the sentence.

Validity and satisfiability are Connected eachother i.e,

- α is valid iff $\neg\alpha$ is unsatisfiable.

(or)

- α is satisfiable iff $\neg\alpha$ is not valid.

↳ Contrapositive.

[Note: $\alpha \models \beta$ if and only if $(\alpha \wedge \neg\beta)$ is unsatisfiable

Reasoning PATTERNS IN propositional Logic:

Patterns of inference that can be applied to derive chains of conclusions that Lead to the desired goal. These Patterns of inference are called "inference rules".

① Modus ponens

$$\frac{\alpha \Rightarrow \beta \;,\; \alpha}{\beta}$$

i.e, when we have a sentence of the form $\alpha \Rightarrow \beta$ and α are given, then the sentence β can be inferred.

forEx: (wumpusAhead ∧ wumpusAlive) ⟹ shoot

and (wumpusAhead ∧ wumpusAlive)

⟹ Here shoot can be inferred.

② AND-Elimination

It says that, from a Conjoction any of the Conjucts can be inferred.

$$\frac{\alpha \wedge \beta}{\alpha}$$

Ex: (WumposAhead ∧ WumposAlive),

from this WumposAlive can be inferred.

③ Demorgan's: $\neg(\alpha \wedge \beta)$ written as $\neg\alpha \vee \neg\beta$

→ All the logical equivalences we discussed can be used as inference rules

→ Let us see how these inference rules and equivalences can be used in "wumpos world".

→ we start with KB containing R_1 through R_5. and show how to prove $\neg P_{1,2}$ i.e, No pit in [1,2].

Now, we apply Bidirectional elimination to R_2 i.e, it produces R_6.

$$R_6: (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

Apply And-Elimination to R_6.

$$R_7: ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

Logical equivalence for Contrapositives gives

$$R_8: (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$$

→ Now, apply modus ponens with R8. and the percept R4 (i.e, $\neg B_{1,1}$)

$R_9 : \neg(P_{1,2} \vee P_{2,1})$

finally apply demorgan's rule.

$R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$

i.e, [1,2] [2,1] not Contains pit location.

→ So, Inference roles are <u>Sound</u> because it can be proven by truth table argument

→ If we allow all possible inference roles, we are searching in an infinite space hence <u>not Complete</u>

→ <u>monotonicity</u> : Set of entailed sentences Can increase as information added to knowledge base. i.e, For any sentences α and β

if $KB \models \alpha$ then $KB \wedge \beta \models \alpha$

Suppose the knowledge base have additional assertion β stating that there are exactly 8 pits in the world. So, it can't invalidate the conclusion α that was already inferred in previous percept.

Resolution:

Resolution allows a complete inference algorithm (search-Based) using only a single inference rule.

Let us Consider a simple version of resolution rule in wumpus world.

Consider the steps leading in wumpus world fig(c) i.e, agent returns from [2,1] to [1,1] and then goes to [1,2] when it recieves a stench, but no breeze

so, we add following rules to knowledge base;

R_{11} : $\neg B_{1,2}$

R_{12} : $B_{1,2} \iff (P_{1,1} \vee P_{2,2} \vee P_{1,3})$

By this process we come back to R_{10} rule we can now derive the absence of pits on [2,2] and [1,3]

[note: [1,1] is already known to be pitless]

R_{13} : $\neg P_{2,2}$

R_{14} : $\neg P_{1,3}$

we can apply bidirectional elimination to R_3 followed by modus ponens with R_5 to obtain fact that there is a pit in [1,1] [2,2] (or) [3,1]

R_{15} : $P_{1,1} \vee P_{2,2} \vee P_{3,1}$

Now, Comes first application of resolution rule: i.e, $\neg P_{2,2}$ in R_{13} resolves with the literal $P_{2,2}$ in R_{15} to give;

R_{16} : $P_{1,1} \vee P_{3,1}$

From R_1 resolves $\neg P_{1,1}$ in R_{16} to give,

R_{17} : $P_{3,1}$

Resolution principle: (l_i m_j $\neg P, P$)

$$\frac{l_1 \vee \cdots \vee l_k, \quad m_1 \vee \cdots \vee m_n}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

where l_i and m_j are Complementary literals.

Resolution takes two clauses and produces a new clause containing all the literals of the two original clauses except the two complementary literals.

i.e, $$\frac{l_1 \vee l_2, \quad \neg l_2 \vee l_3}{l_1 \vee l_3},$$

Ex: wumpus world

$$\frac{P_{1,1} \vee P_{3,1}; \quad \neg P_{1,1} \vee \neg P_{2,2}}{P_{3,1} \vee \neg P_{2,2}}$$

Factoring: The resulting clause of resolution should contain only one copy of each literal. i.e, the removal of multiple copies is called factoring.

Ex: if we resolve (A∨B) with (A∨¬B) we get (A∨A) it is reduced to just A

CNF [Conjuctive Normal Form]:

Resolution rule only apply to disjunction of literals, so how it lead to a complete inference procedure for all propositional Logic? [i.e, every sentence of PL is Logically equivalent to a conjunction of disjunctions of literals]

⇒ so, a sentence expressed in conjunction of disjunctions of literals it is said to be Conjunctive normal form [CNF]

⇒ Consider a family of K-CNF sentences, consists of K literals exactly per clause:

$$(l_{1,1} \vee \cdots \vee l_{1,K}) \wedge \cdots \wedge (l_{n,1} \vee \cdots \vee l_{n,K})$$

it states that, every sentence can be transformed into a 3CNF sentence that has an equivalent set of models.

Instead of proving these assertions, Consider Simple Conversion procedure;

i.e, By converting R_2, the sentence $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ in to CNF. it is as follows;

(1) Eliminate $\Leftrightarrow$ replacing $\alpha \Leftrightarrow \beta$ with

$(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

(2) Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

(3) CNF requires $\neg$ to appear only in literals, so we move $\neg$ inwards by repeated application of following equivalences

$\neg(\neg\alpha) \equiv \alpha$ (double negation elimination)

$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ (demorgan's)

$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ (demorgan's)

Apply on above statement i.e,

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

(4) Now we have sentence Containing $\vee$ and $\wedge$ applied to literals. Distribute $\vee$ over $\wedge$

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

so the final sentence is now in CNF having conjunction of 3 clauses.

Resolution Algorithm:

As we know that inference procedure based on resolution work so, show that, $KB \models \alpha$, and $(KB \wedge \neg\alpha)$ is unsatisfiable we do this by proving a contradiction.

so Resolution Algorithm shown below Converts $(KB \wedge \neg\alpha)$ into CNF. Then the resolution rule is applied to resulting clauses.

Algorithm:

function PL-RESOLUTION (KB, α) returns True/false
inputs: KB, the knowledge base, a sentence in propositional logic
α, query, a sentence in propositional logic

clauses ← The set of clauses in the CNF representation of $KB \wedge \neg\alpha$.
new ← { }
Loop do
  for each C_i, C_j in clauses do
    resolvents ← PL-RESOLVE(C_i, C_j)
    if resolvents contains the empty clause then return TRUE
    new ← new ∪ resolvents
  if new ⊆ clauses then return false
    clauses ← clauses ∪ new

⇒ Here, is the partial application of PL-RESOLUTION to a simple inference in the wumpus world.

Here each pair that contains complementary literals is resolved to produce a new clause which is added to the set if it is not already present. This process continues until one the following 2 things happen;

a) → There are no new clauses to add, in such case α doesn't entail β.

b) → Resolution rule derives empty clause in such case α entails β.

for Example Consider the clauses resolved in CNF at last; and Consider agent is in [1,1] so no breeze at [1,1] i.e, no pits at neighboring square.

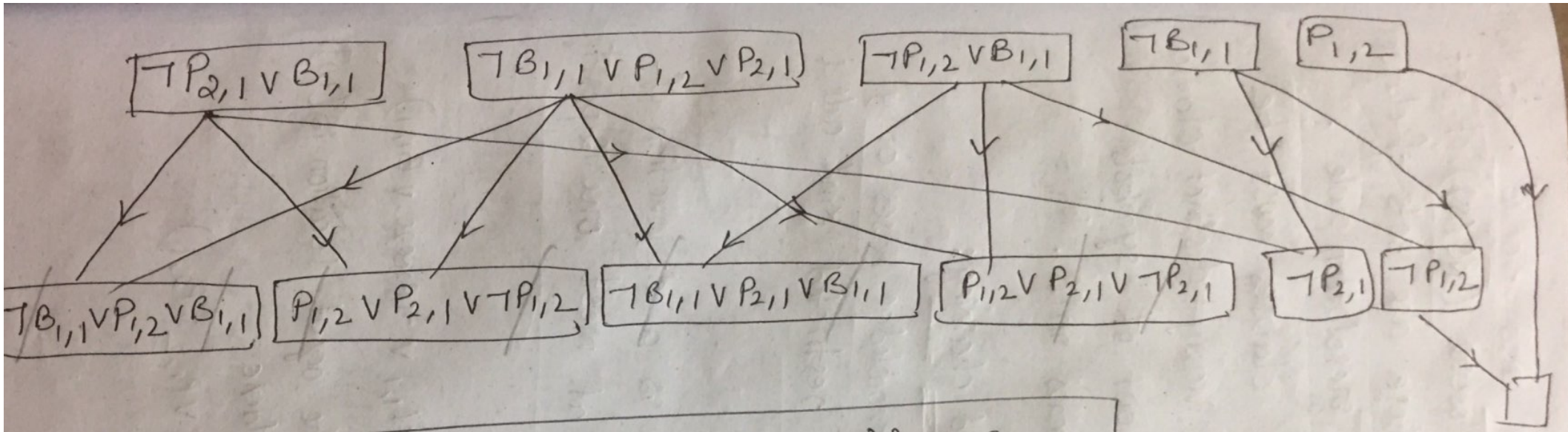
So the KB is;

$$KB = R_2 \wedge R_4 = \left[(B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1} \right]$$

we wish to prove that suppose α is $\neg P_{1,2}$.

So Consider ~~Contra~~ ~~comp~~ Contradiction of query i.e, $\boxed{\neg P_{1,2}}$ — no pit in [1,2]

and $\boxed{\neg B_{1,1}}$ also



i.e, $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$

$\neg\alpha = \neg(\neg P_{1,2}) = P_{1,2}$

$KB \wedge \neg\alpha = \underbrace{(\neg P_{2,1} \vee B_{1,1}) \wedge (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg B_{1,1})}_{KB} \wedge \underbrace{(P_{1,2})}_{\alpha}$

* Resolution is Complete and Sound. for this, Consider resolution closure (RC(S)) of set of clauses S. These are sets of clauses are derived by resolution rule. There RC(S) set of clauses must be finite. Then only many distinct clauses can be constructed out of symbols $P_1 \dots P_k$ that appears in S.

forward and Backward chaining:

Real world knowledge bases only Contain clauses of restricted kind called "Horn clauses".

A Horn clause is a disjunction of literals of which at most one is positive.

for Ex:

The clause $(\neg L_{1,1} \vee \neg Breeze \vee B_{1,1})$

where $L_{1,1}$ means the agent Location is [1,1]

so, it is a Horn clause

But not $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$

→ The restriction to just one positive literal is actually very important for 3 reasons;

(1) Every Hornclause can be written as an implication whose premise is a conjunction of positive literals and whose conclusion is a single positive literal.

for Ex: Hornclause $(\neg L_{1,1} \vee \neg Breeze \vee B_{1,1})$

can be written as $(L_{1,1} \wedge Breeze) \Rightarrow B_{1,1}$.

[i.e, $(L_{1,1} \wedge Breeze) \Rightarrow B_{1,1}$.]

i.e, if the agent is in [1,1] and there is a breeze, then [1,1] is Breezy.

→ Horn clauses like the one with exactly one positive literal are called definite clauses.

The positive literal is → Head
negative literal is → body of clause

→ A definite clause with no negative literals simply asserts a given proposition clause sometimes called fact.

→ A Horn clause with no positive literals can be written as an implication, whose conclusion is false.

for Ex: $(\neg W_{1,1} \vee \neg W_{1,2})$ is equal to

$$W_{1,1} \wedge W_{1,2} \Rightarrow \text{false}.$$

Such sentences are called integrity constraints in database and are used to signal errors in data.

(2) Inference with Horn clauses can be done through forward chaining and backward chaining algorithms.

(3) Deciding the entailment with Horn clauses that can be done in time that is linear in size of knowledge base.

Algorithm:

function PL-FC-Entails (KB, q) returns true or false

inputs: KB, the knowledge base, a set of propositional Horn clauses

q, query, a proposition symbol

local variables: count, a table, indexed by clause, initially the no. of premises

: inferred, a table, indexed by symbol, each entry initially false

: agenda, a list of symbols, initially symbols known to be true in KB.

while agenda is not empty do.

p ← POP (agenda)

unless inferred [p] do

inferred [p] ← TRUE

for each Horn clause c in whose premise p appears do

decrement count [c]

if count [c] = 0 then do

if HEAD[c] = q then return true

PUSH (HEAD [c], agenda)

return false.

---

this algorithm PL-FC-ENTAILS? (KB, q) determines whether a single proposition symbol q - the query - is entailed by a KB of Horn clauses.

→ It begins from known facts of KB.
↳ positive literals

→ if all premises of an implication are known then its Conclusion is added

to set of known facts.

forex: if $L_{1,1}$ and Breeze are known and $(L_{1,1} \wedge Breeze) \Rightarrow B_{1,1}$ is in KB, then $B_{1,1}$ is added.

This process continues until the query q is added or no further inferences can be made.

Let us Consider the example;

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

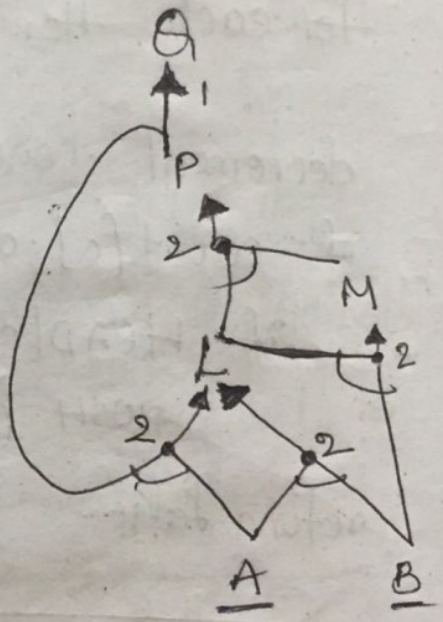
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a) Simple knowledge base of Horn clauses



(b) AND-OR Graph

[Note: premise = Conjunction of +ve literals]

→ Suppose Q is the query. So we have to prove $P \Rightarrow Q$.

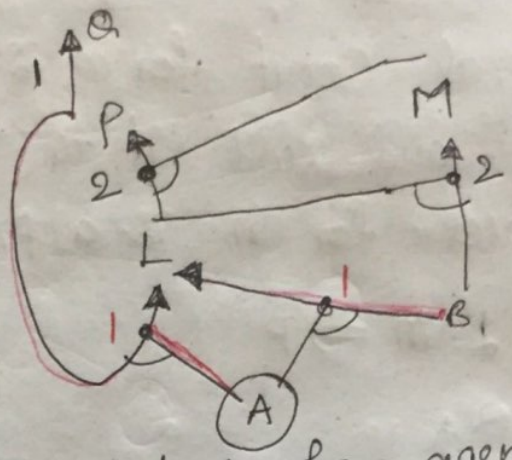
→ Now, from A start examining by looking the things that are going to be true

Consider All clauses in which A is a premise. i.e,

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

So decrease no. of literals for clauses given above,



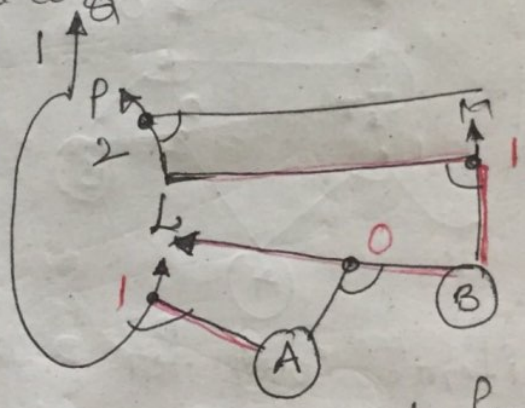
Next known literal from agenda is B

Consider All clauses in which B is a premise.

& Reduce literal count

$B \wedge L \Rightarrow m$

$A \wedge B \Rightarrow L$



→ Now the literal count for L is 0

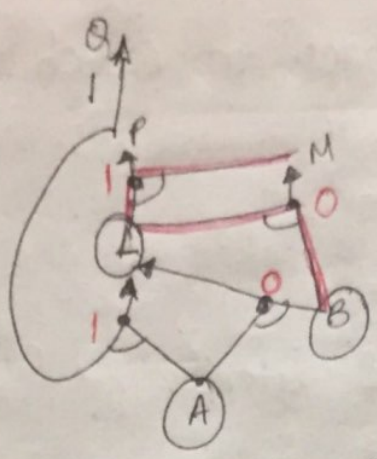
So, add L to agenda and explore it

i.e, All clauses in which L is a premise.

$L \wedge m \Rightarrow P$

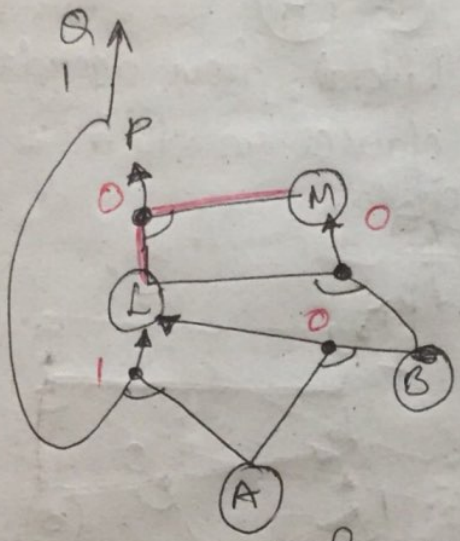
$B \wedge L \Rightarrow m$

Now, Reduce count of literals.



Next literal count for m is '0' so, explore m and add it to the agenda i.e, clause for m is a premise is;

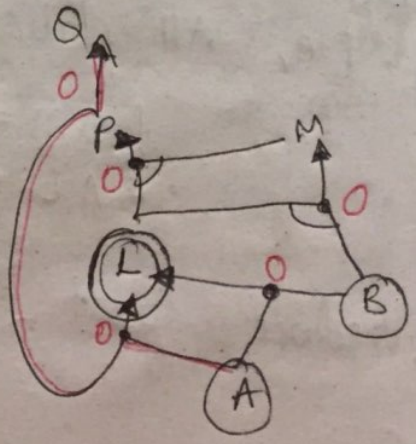
$$L \wedge M \Rightarrow P$$



Reduce literal count

Now, literal count for P is 0 so add P to agenda and explore all clauses in which P is a premise i.e,

$$A \wedge P \Rightarrow L$$
$$P \Rightarrow Q$$



Backward chaining algorithm, works backwards from the query. If the query q is known to be true, then no work is needed. otherwise, the algorithm finds those implications in the Knowledge base that conclude q.

Back ward chaining is a "goal directed reasoning".

Effective propositional Inference:

Here we describe two families of efficient algorithms for propositional inference based on model checking.

① Based on backtracking search
② hillclimbing search [localsearch]

these algorithms are used to describe the satisfiability checking.

① Davis-putnam Algorithm:

This algorithm was proposed by Martin Davis and Hilary putnam, Logemann and Loveland in (1962). so it is also called DPLL Algorithm.

→ It takes input as a CNF i.e, a set of clauses.

→ Like Backtracking search and, TT-ENTAILS? it ~~is~~ ~~and~~ embodies three improvements to the scheme of entailment.

① Early Termination: The algorithm detects whether the sentence must be true or false even with a partially completed model.

i.e, a clause is true if any literal is true. before completing a model.

Ex: The sentence $(A \vee B) \wedge (A \vee C)$ is true if A is true, regardless of values of B & C

→ Early termination avoids examination of entire subtrees in the search space.

② pure symbol Heuristic:

It is a symbol that always appears with same sign in all clauses.

Ex: $(A \vee \neg B)$, $(\neg B \vee \neg C)$ and $(C \vee A)$

symbol A is pure because only possitive literals

B is pure because only negative literals but, C is impure because of different sign.

→ The algorithm ignore clauses that are already known to be true in model constructed so far.

forEx: if model contains B=false then clause (¬B ∨ ¬C) is already true and C becomes pure because it appears only in (C ∨ A).

③ unit clause heuristic:

A unit clause is a clause with just one literal.

forEx: if the model contains B=false then (B ∨ ¬C) becomes a unit clause because it is equivalent to (false ∨ ¬C)

= ¬C.

To make this clause true, C must be false.

David putnam Algorithm:

→ Consider a sentence in CNF

→ $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_3)$

→ This algorithm picks a variable & assigns a value for it i.e, either (0 or 1)

→ Consider for example $x_2 = 0$, substitue it in the formula to produce smaller one.

→ From this, three effects may be possible.

→ Some clauses are already satisfied by this assignment, so they can be removed

early termination { i.e, $(x_1 \vee \bar{x}_2 \vee x_3)$ is satisfied for $x_2 = 0$

$(x_1 \vee \bar{0} \vee x_3) \Rightarrow (x_1 \vee 1 \vee x_3) \Rightarrow$ True irrespective of x_1 and x_3

→ Some clauses simply get smaller

unit clause { Ex: $(\bar{x}_1 \vee x_2 \vee \bar{x}_3)$ becomes $(\bar{x}_1 \vee \bar{x}_3)$

→ In some clauses variable are not effected i.e, $(x_1 \vee \bar{x}_3)$

→ After this situation the Algorithm tries to find pure symbol to satisfy this. If succeds solution is returned,

otherwise, the algorithm checks satisfiability with $x_2 = 1$.

Algorithm:

Function DPLL-SATISFIABLE?(s) returns true or false

inputs: s, a sentence in PL

clauses: a set of clauses in CNF representation of s.

symbols: a list of proposition symbols in s.

return DPLL(clauses, symbols, [])

Function DPLL(clauses, symbols, model) returns true or false

if every clause in clauses is true in model then return True.

if some clauses in clauses is false in model then return false

P, value ← FIND-PURE-SYMBOL(symbols, clauses, model)

if P is non-null then return DPLL(clauses, symbols-P, EXTEND(P, value, model)

P, value ← FIND-UNIT-CLAUSE(clauses, model)

if P is non-null then return DPLL(clauses, symbols-P, EXTEND(P, value, model)

P ← FIRST(symbols); rest ← REST(symbols)

return DPLL(clauses, rest, EXTEND(P, true, model)) or DPLL(clauses, rest, EXTEND(P, false, model))

Walksat Algorithm:

It is a Radomized Algorithm that Searches for satisfying assignment of input. i.e,

→ the goal of this Algorithm is to find an assignment that satisfies every clause.

→ Here, the evaluation function is used to Count the number of unsatisfied clauses.

→ It belongs to the family of Localsearch

→ first it starts by random assignment to all literals in the given input

for ex: $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_3)$

let the assignment be 'A'

so $A = 001$ [assume it]

→ if it finds/satisfies every clause then it returns true.

→ if it doesn't find a satisfying i/p within bound it stops and return failure.

→ It works iteratively by improving 'A' i.e, picks an unsatisfied clause and picks a symbol in that clause to flip.

→ It chooses randomly to pick a symbol to flip depends on 2 ways:

a) "min-conflicts" step (minimizes no. of unsatisfied clauses in new state)

b) "random-walk" (picks a symbol randomly)

→ In the above assignment

x_1	x_2	x_3
0	0	1

with this only 3rd clause was not satisfied so, pick randomly among x_1 and x_3 and flip its value.

i.e, $(x_1 \vee \bar{x}_3)$ is not satisfied

so, for example flip x_1 i.e, $\boxed{A = \underline{1}01}$

Now, check the sentence whether all clauses are satisfying (or) not.

second clause is not satisfying Now, so, choose x_3 to flip Now.

i.e, $\boxed{A = 00\underline{0}}$

Hence it is satisfying every clause in i/p

→ Hex it chooses a random unsatisfied clause and flips a random variable in that clause with a probability $P > 0$

i.e, $P = 0.5$

WalkSAT Algorithm :

function WALKSAT(clauses, p, max_flips) returns a satisfying model or failure

inputs : clauses, a set of clauses in PL

p, probability of choosing to do a "random walk" move typically around 0.5

max_flips, no. of flips allowed

model ← A random assignment of true/false to symbols in clauses

for i = 1 to max_flips do

if model satisfies clauses then return model

clause ← a randomly selected clause from clauses that is false in model.

with probability p flip the value in model of a randomly selected symbol from clause

else flip whichever symbol in clause maximizes the number of satisfied clauses

return failure.

Hard satisfiability problems

As we look at DPLL & WALKSAT and their performance. we are interested in hard problems.

for EX: N-queens problem. it is quite tricky for backtracking search Algorithms: there are trivially easy for Local-search methods because it is underconstrained problem.

Suppose, we look at satisfiability problems on CNF. it is not that much of under Constrained i.e, overconstrained problem. that is many clauses relative to nomber of variables to have no solution.

$CNF_K(m,n)$ - K-CNF with m clauses & n symbols and clauses are choosen uniformly / independently.

→ The probability of a random 3-CNF sentence is satisfiable, as a function of clause / symbol ratio;

i.e, $\boxed{m/n}$.