1.)

Problem Solving using search

(i) States are treated as a black box - an indivisible structure.

(ii) The problem is solved when the current state matches one of the goal states.

(iii) They use problem and domain-specific heuristics to find solutions.

(iv) The states support direct tests to determine if it's a goal state.

Constraint Satisfaction problem

(i) States are treated as a structure made up of different components - a set of variables, each of which is assigned some value.

(ii) The problem is solved when each variable is assigned a value that satisfies all the constraints on the variable.

(iii) They use general-purpose heuristics

(iv) the goal test is determined by whether or not the set of constraints on variable values are satisfied.

Cryptarithmetic Problem

$$\begin{array}{r} C_3\,C_2\,C_1\;\; \\ EAR \\ +\,EAR \\ \hline DRUM \end{array}$$

- Variables: $\{D, E, A, R, U, M, C_1, C_2, C_3\}$
- Domains for each variable : $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Constraints : Alldiff (D, E, A, R, U, M)
  - $R + R = M + 10 \cdot C_1$
  - $C_1 + A + A = U + 10 \cdot C_2$
  - $C_2 + E + E = R + 10 \cdot C_3$
  - $C_3 = D$,
  - $E \neq 0$, $D \neq 0$.

20234747017

2

We know that the sum of two one-digit numbers plus a carry can at most be 19, i.e.

$C_2 + E + E \le 19$

$\Rightarrow C_3$ can at most be 1.

Since $C_3 = D$, and the constraint $D \neq 0$ exists, D must be 1.

$\therefore$, $C_3 = D = 1$

$$\begin{array}{rcccc} & 1 & C_2 & C_1 & \\ & & E & A & R \\ + & & E & A & R \\ \hline & D & R & U & M \end{array}$$

Now, $C_2 + 2E = 10 + R$

if R is even, $C_2 = 0$

if R is odd, $C_2 = 1$

Out of these two alternatives, let's first explore $C_2 = 0$

Since $C_2 = 0$, R must be even.

Possible values for $R = \{2, 4, 6, 8\}$

(R can't be 0 since then $R + R = M = 0$ violates constraints)

and,

$2E = 10 + R$

$\therefore$, possible values for $E = \{6, 7, 8, 9\}$

Now,

$C_1 + 2A = U$ ($\because C_2 = 0$)

Again, two alternatives for C_1 exist. Let's first explore $C_1 = 1$

then,

$1 + 2A = U$, i.e. U is odd.

$2R = 10 + M$

Since $C_1 = 1$, R can't be any of $\{2, 4\}$ since that will not generate a carry.

Thus, possible values of $R = \{6, 8\}$

possible values of $E = \{8, 9\}$

$$\begin{array}{rcccc} & 1 & 0 & 1 & \\ & & E & A & R \\ + & & E & A & R \\ \hline & 1 & R & U & M \end{array}$$

Let $R = 6$.

then, $E = 8$, ($\because 2E = 10 + R$)

and, $M = 2$ ($\because 2R = 10 + M$)

20234747017

Now, $1 + 2A = U$

$$\begin{array}{cccc} & 1 & 0 & 1 \\ \hline & 8 & A & 6 \\ & 8 & A & 6 \\ \hline 1 & 6 & U & 2 \end{array}$$

Possible values of $A = \{3, 4\}$
(all other values are either taken or ~~et~~ generate a carry.)

$\therefore$ possible values of $U = \{7, 9\}$ ($\because 1 + 2A = U$)

Let $A = 3$,
then, $U = 7$

$$\begin{array}{cccc} 1 & 0 & 1 & \\ & 8 & 3 & 6 \\ & 8 & 3 & 6 \\ \hline 1 & 6 & 7 & 2 \end{array}$$

All constraints are satisfied. Therefore, this is one of the solutions of the problem.

Variable Values

$C_3 = D = 1$, $C_2 = 0$, $C_1 = 1$

$R = 6$

$E = 8$

$M = 2$

$A = 3$

$U = 7$.

There can be many more solutions that take different choices for C_2, C_1, and other possible values for R, and A. For example, another solution is if we choose $A = 4$ ($\Rightarrow U = 9$)

$$\begin{array}{ccccc} & 1 & 0 & 1 \\ \hline & 8 & 4 & 6 \\ + & 8 & 4 & 6 \\ \hline 1 & 6 & 9 & 2 \end{array}$$

20 234 747017

2.) Types of Knowledge Representation techniques:

(i) Logic-based representations : contain facts, premises, rules for propositional logic, predicate logic, etc. They have well defined syntax and semantics and ~~eff~~ inferencing techniques (which may or may not be very efficient).

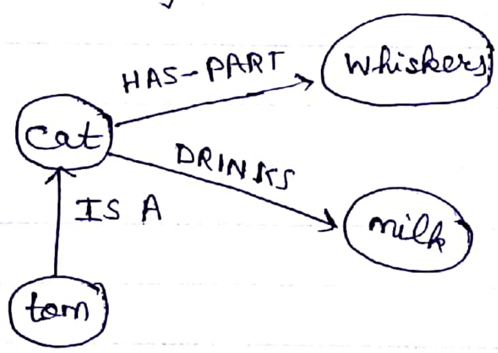
eg: HUMAN(x): x is a human
MORTAL(x): x is mortal

then, the sentence "all humans are mortal" is represented as:

$\forall x$ HUMAN(x) $\rightarrow$ MORTAL(x).

(ii) Semantic Networks: allow us to define relations using nodes and links. Related information is bound together using these links.

eg:



only restricted inference based on inheritance is supported.

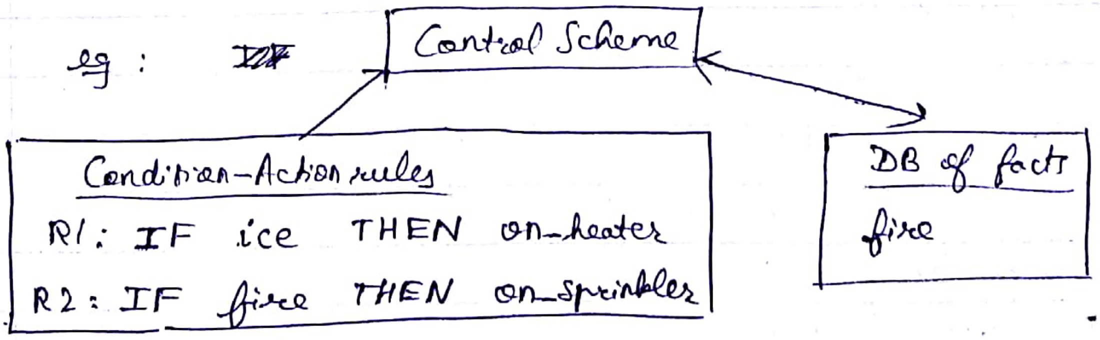
(iii) Frames : Knowledge about an object or an event is stored together in memory as a unit. they consist of slot and slot values. The slots specify general or specific characteristics of the entity which the frame represents.

eg : Elephant :
subclass : mammal
size : large
haspart : trunk

Nellie
instanceof : Elephant
likes : Banana,

(iv) Rule based representations are used in some specific problem solving context. they involve production rules that say what to do, given various conditions are satisfied.

eg :



20234747017

6

Sentence S: Someone walked slowly to the supermarket

Lexicon L

Pronoun → someone

Verb → walked

Adv → slowly

Prep → to

Article → the

Noun → Supermarket

Grammar A

S → NP VP

NP → Pronoun | Article Noun | Noun

VP → VP PP | VP Adv Adv | Verb

~~VP → Verb~~

Prep → Prep NP.

Attempt to derive S using grammar A and given lexicon

S ⇒ NP VP

⇒ Pronoun VP

⇒ someone VP

⇒ someone V

⇒ someone walked

fail!

or S ⇒ NP VP

as before, ⇒ someone VP

⇒ someone VP PP

no grammar rules for PP! fail!

or S ⇒ NP VP

⇒ someone VP Adv Adv

This will keep generating either verbs or adverbs, and we will not get the sentence ever.

Therefore, grammar A doesn't generate the given sentence,

20234747017

(7)

Grammar B

S → NP VP

NP → Pronoun | Noun | Article NP

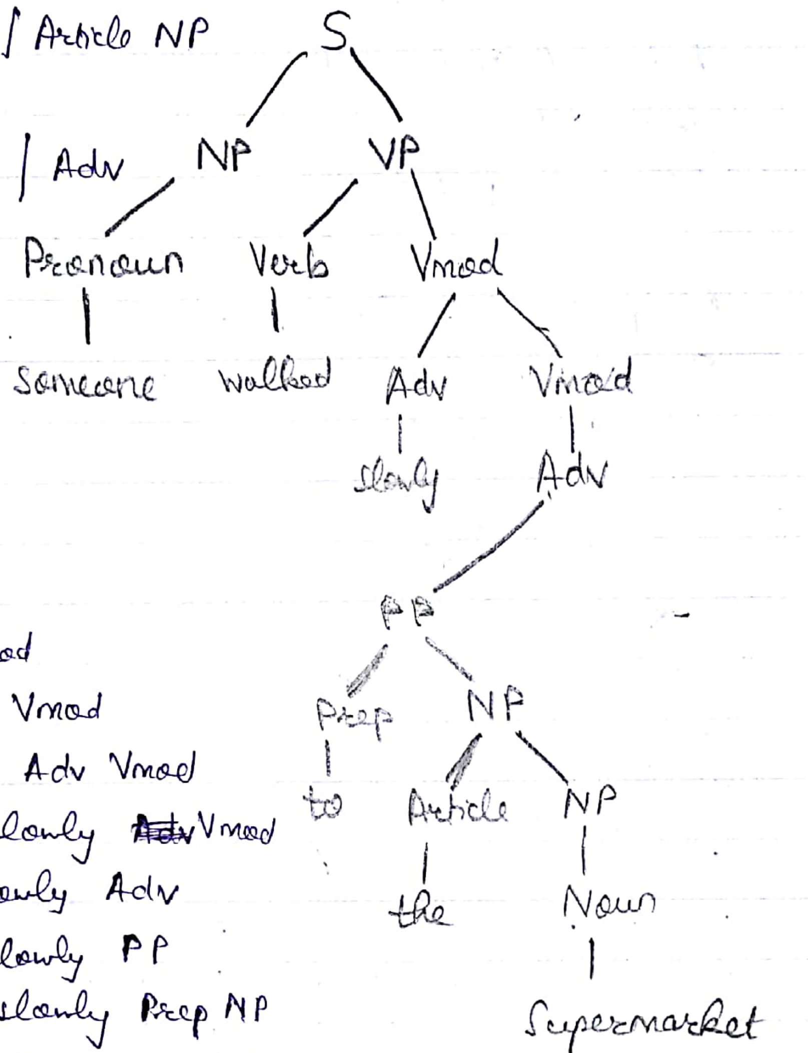
VP → Verb Vmod

Vmod → Adv Vmod | Adv

Adv → PP

PP → Prep NP

Parse tree for Grammar B



Attempt to derive S

S ⇒ NP VP

⇒ Pronoun VP

⇒ someone VP

⇒ someone Verb Vmod

⇒ someone walked Vmod

⇒ someone walked Adv Vmod

⇒ someone walked slowly ~~Adv~~ Vmod

⇒ someone walked slowly Adv

⇒ someone walked slowly PP

⇒ someone walked slowly Prep NP

⇒ someone walked slowly to NP

⇒ someone walked slowly to Article NP

⇒ someone walked slowly to the NP

⇒ someone walked slowly to the supermarket.

Success!

2 0 2 3 4 7 4 7 0 1 7

8

Grammar C

S → NP VP

NP → Pronoun | Article NP | Noun

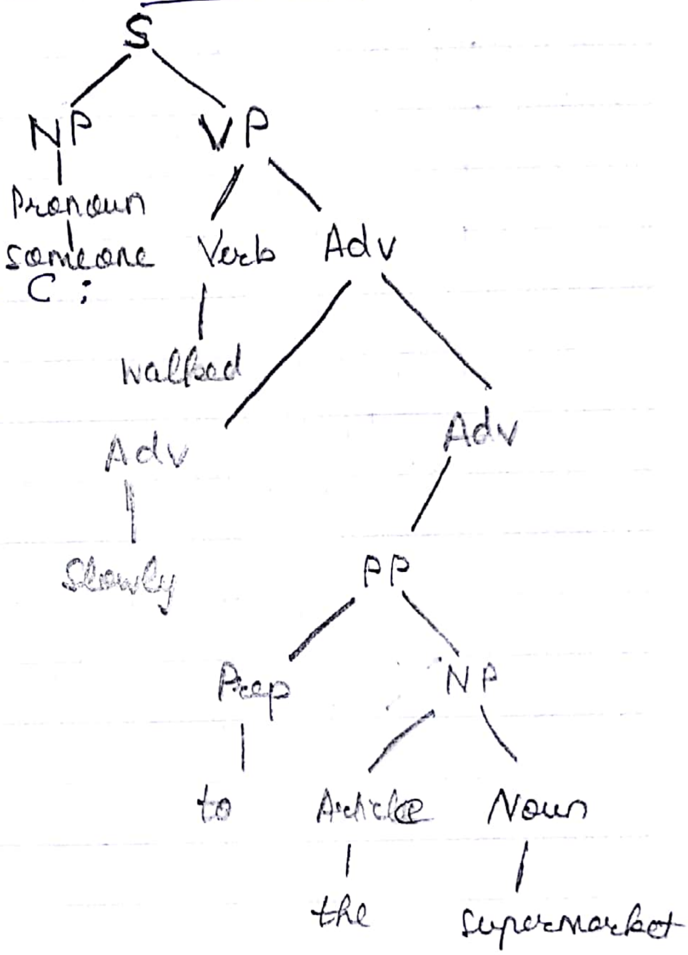
VP → Verb Adv

Adv → Adv Adv | PP

~~Adv → PP~~

PP → Prep NP

Parse Tree



Attempt to derive S using grammar C :

S ⇒ NP VP

⇒ Pronoun VP

⇒ Someone VP

⇒ Someone Verb Adv

~~⇒ Someone slowly Adv~~

~~⇒ Someone slowly Adv Adv~~

~~⇒ Someone slowly~~

⇒ someone walked Adv

⇒ someone walked Adv Adv

⇒ someone walked slowly Adv

⇒ someone walked slowly PP

⇒ someone walked slowly Prep NP

⇒ someone walked slowly to NP

⇒ someone walked slowly to Article NP

⇒ someone walked slowly to the NP

⇒ someone walked slowly to the Noun

⇒ someone walked slowly to the supermarket

Success !

20 234 747 017

9

3.) • A heuristic function $h(n)$ denotes the estimated cost of the cheapest path from the state at node n to a ~~goal~~ goal state.

• A heuristic function is said to underestimate when the heuristic's estimate is lower than the actual path cost.

• A heuristic function is said to overestimate when the heuristic's estimate is higher than the actual path cost.

Conditions for optimality of A*:

(i) $h(n)$ must be an admissible heuristic - i.e., a heuristic that never overestimates the cost to reach the goal.

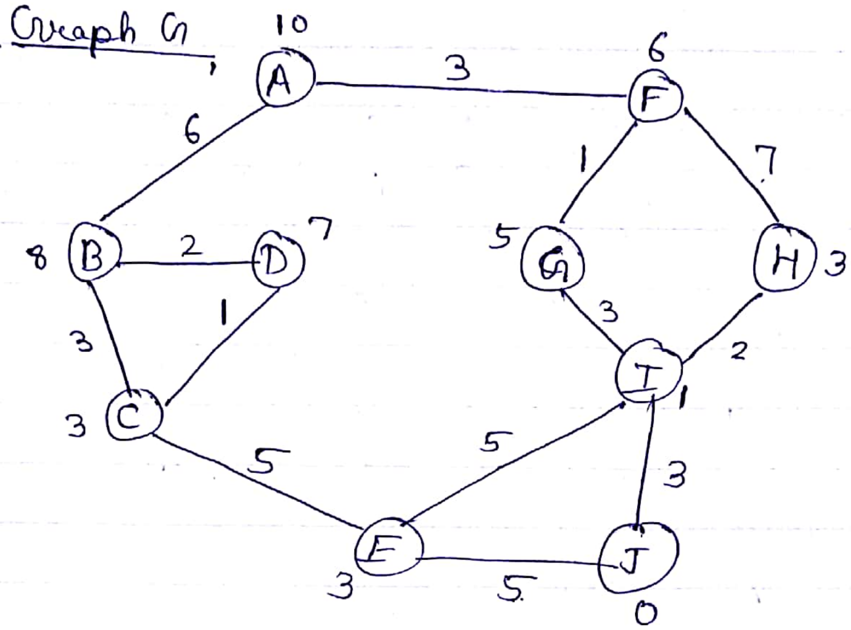
(ii) $h(n)$ must be consistent - for every node n and every successor n' of n generated by some action a,

$$\underbrace{h(n)}_{\text{estimated cost of reaching the goal from } n} \leq \underbrace{c(n,a,n')}_{\text{step cost of getting from } n \text{ to } n'} + \underbrace{h(n')}_{\text{estimated cost of reaching the goal from } n'}$$

20 234 747 017

10

Graph G

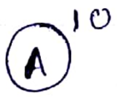


numbers on edges: distance between nodes.
numbers on node: $h(n)$, $n = A, B, \ldots, J$.
Best path from start state A to goal state J:

(a) Greedy Best First Search algorithm

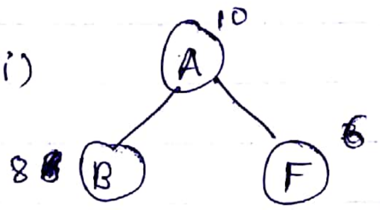
~~only~~ at each step, select the node with the lowest heuristic value out of the nodes generated so far.

(i)



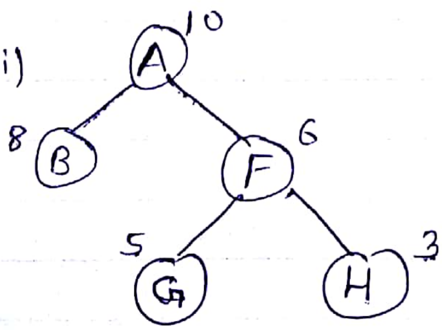
only one node, so we expand it to generate successor nodes.

(ii)



node F is the most promising, so expand F next. successor nodes G and H are produced.

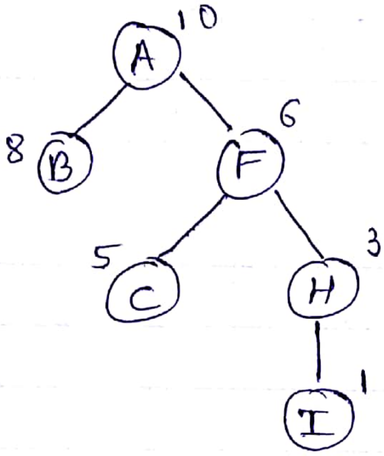
(iii)



node H is the most promising, so we expand node H to generate successor node I.

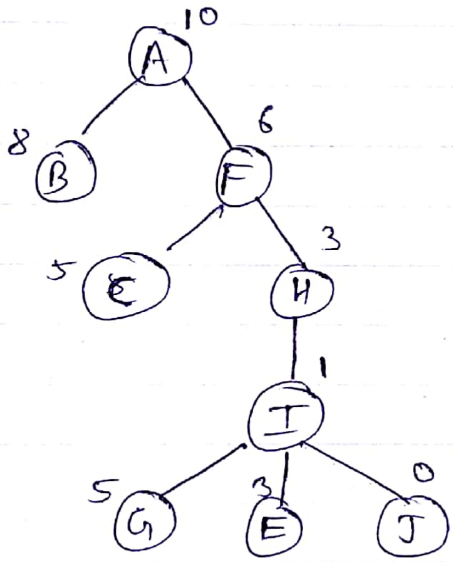
20 234 747 017

(iv)



node I most promising.
∴ generate successor nodes
G, E, J

(v)



node J most promising,
and it is the goal state.

thus, we have reached the goal state.

Best path : $A \longrightarrow F \longrightarrow H \longrightarrow I \longrightarrow J$

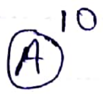
Actual cost of path $= 3 + 7 + 2 + 3 = 15$

→

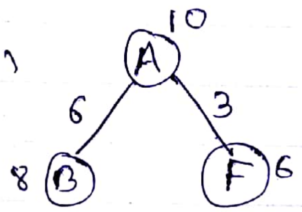
(b) A* algorithm $f(n) = g(n) + h(n)$

$g(n)$: cost of edge; $h(n)$: heuristic

Choose lowest $f(n)$ at each step.

(i) 

only one node, expand to generate successor nodes.

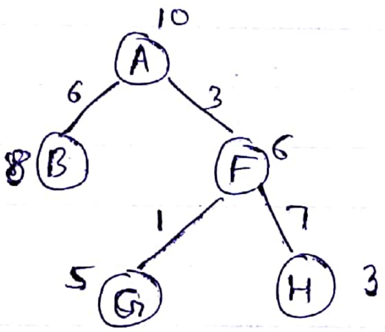
(ii) 

$f(B) = 6+8 = 14$

$f(F) = 3+6 = 9$

∴ F is more promising

Expand F.

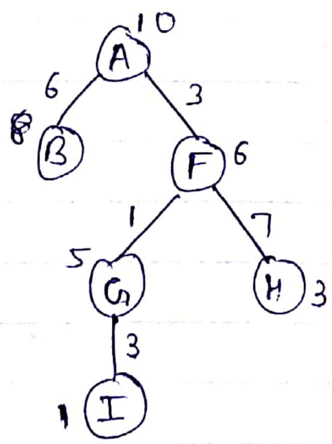
(iii) 

$f(G) = 3+1+5 = 9$

$f(H) = 3+7+3 = 13$

∴ G is more promising

Expand G

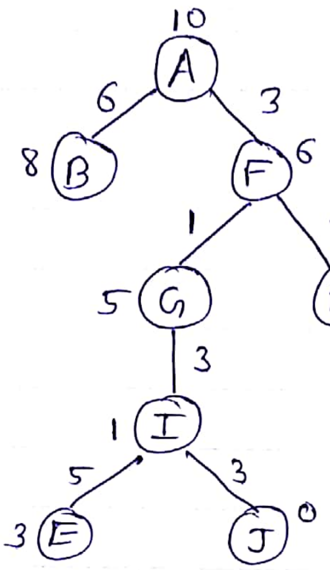
(iv) 

$f(I) = 3+1+3+1 = 8$

∴ I is more promising

Expand I

(v)



$f(E) = 3+1+3+5+3$
$= 15$

$f(J) = 3+1+3+3+0$
$= 10$

J is more promising,
and J is a goal node!

Thus, we have reached the goal.

Best path using A*:

$A \rightarrow F \rightarrow G \rightarrow I \rightarrow J$

Actual Cost of path $= 10$

Clearly, this path is better than greedy Best first search.

20234747017

5.) Let the predicates be:

LOVES(x, y) : x loves y

ANIMAL(y) : y is an animal

KILLS(x, y) : x kills y

CAT(y) : y is a cat

(a) Sentences in Predicate Logic

A1 : $\forall x\ (\forall y\ ANIMAL(y) \rightarrow LOVES(x,y)) \rightarrow (\exists z\ LOVES(z,x))$

~~A2:~~

A2 : $\forall x\ (\exists y\ ANIMAL(y) \wedge KILLS(x,y)) \rightarrow (\forall z\ \sim LOVES(z,x))$

A3 : $\forall y\ ANIMAL(y) \rightarrow LOVES(Jack, y)$

A4 : $KILLS(Jack, Pussy) \vee KILLS(John, Pussy)$

A5 : $\forall y\ CAT(y) \rightarrow ANIMAL(y)$

A6 : $CAT(Pussy)$

(b) Prenex Normal Form

~~A1~~

- ~~Eliminate $\rightarrow$~~
- ~~$\forall x \sim(\sim\forall y\ ANIMAL(y) \vee LOVES(x,y)) \vee (\exists z\ LOVES(z,x))$~~
- ~~Move negations inwards~~

~~$\forall x\ \cancel{\exists x} \sim(\exists y \sim ANIMAL(y) \wedge \sim LOVES(x,y)) \vee (\exists z\ LOVES(z,x))$~~

~~$\forall x\ \cancel{\exists x}\ \forall y\ ANIMAL(y) \vee LOVES(x,y) \vee \exists z\ LOVES(z,x)$~~

(b) Prenex Normal form

A1

(i) eliminate $\rightarrow$

$\forall x\ (\sim \forall y (\sim ANIMAL(y) \vee LOVES(x,y))) \vee \exists z\ LOVES(z,x)$

(ii) move negations inward

$\forall x\ \exists y\ (ANIMAL(y) \wedge \sim LOVES(x,y)) \vee (\exists z\ LOVES(z,x))$

(iii) renaming not necessary - bring quantifiers to the front

$\forall x\ \exists y\ \exists z\ (ANIMAL(y) \wedge \sim LOVES(x,y)) \vee \cancel{\exists z} (LOVES(z,x))$

A2

$\forall x\ (\sim \exists y\ (ANIMAL(y) \wedge KILLS(x,y))) \vee (\forall z \sim LOVES(z,x))$

$\equiv \forall x\ (\forall y\ (\sim ANIMAL(y) \vee \sim KILLS(x,y))) \vee \forall z \sim LOVES(z,x)$

$\equiv \forall x\ \forall y\ \forall z\ \sim ANIMAL(y) \vee \sim KILLS(x,y) \vee \sim LOVES(z,x)$

A3 $\forall y \sim ANIMAL(y) \vee LOVES(Jack, y)$

A4 $KILLS(Jack, Pussy) \vee KILLS(John, Pussy)$

A5 $\forall y \sim CAT(y) \vee ANIMAL(y)$

A6 $CAT(Pussy)$

(c) Skolemization

A1:

replace y by $f(x)$

replace z by ~~f~~ $g(x)$

drop existential quantifiers

~~$\forall x$ ANIMAL(f(x)) ∧ ∼LOVES(x,~~

$\forall x\ (ANIMAL(f(x)) \wedge \sim LOVES(x, f(x))) \vee LOVES(g(x), x)$

20 234 747 017

A2: already in skolemized form

$\forall x \forall y \forall z$ ~ANIMAL(y) V ~KILLS(x, y) V ~LOVES(z, x)

A3: $\forall y$ ~ANIMAL(y) V LOVES(Jack, y)
A4: KILLS(Jack, Pussy) V KILLS(John, Pussy)
A5: $\forall y$ ~CAT(y) V ANIMAL(y)
A6: CAT(Pussy)

(d) Resolution algorithm

- Drop universal quantifiers,
- write each clause on a separate line

~~A1~~ A1 (i) ANIMAL(f(x)) V LOVES(g(x), x)
(ii) ~LOVES(x, f(x)) V LOVES(g(x), x)
A2: ~ANIMAL(y) V ~KILLS(x, y) V ~LOVES(z, x)
A3: ~ANIMAL(y) V LOVES(Jack, y)
~~A3: ~ANIMAL(y) V LOV~~
A4: KILLS(Jack, Pussy) V KILLS(John, Pussy)
A5: ~CAT(y) V ANIMAL(y)
A6: CAT(pussy)

Question: who killed the cat? Assume negations. First try John
~~A7: ~KILLS(Jack, Pussy)~~
A7: ~KILLS(John, Pussy)

Resolving A5 and A6 with the substitution {Pussy / y},
we get:
A9: ANIMAL(Pussy)

2023477017

(17)

Resolving A3 and A7, ~~we get~~ with {Pussy/y}, we get

A10 : LOVES(Jack, Pussy)

~~Resolving A2 and A10 with {Jack/z, Pussy/x}~~
~~A11 : ~ANIMAL~~

Resolving A8 and A4, we get
A11: Kills (Jack, Pussy)

Resolving A7 and A2, we got. Substitution {Pussy/y}
A12: ~ KILLS(x, Pussy) V ~LOVES (z, x)

Resolving A12 and A11, we get (substitution {Jack/x})

A13 : ~LOVES (z , Jack)

Resolving ~~A78~~ A1 (ii) and A3, we get
A14: ~ANIMAL(f(Jack)) V LOVES(g(Jack), Jack))

Resolving A14 and A1(i), we get (substitution {Jack/x})
A15 : LOVES (g(Jack), Jack)

Resolving A15 and A13, we get
[ ] empty clause → a contradiction.

∴, John must have killed the cat, because assuming the negation of this leads to a contradiction.