2015

Notes

Everything is probably wrong lol

Question 1

a

A turing machine is an abstract machine that operates on a tape of symbols using a table of rules.

An agent can be modelled as a turing machine where the tape is the environment.

b

Determining from an arbitrary computer program and an input, will the program ever finish running.

It's impossible to determine if an agent will ever find a solution.

c

Determining if there is a solution that satisfies a given boolean formula.

You can phrase any problem as a boolean formula so solving it generally is the same as solving SAT.

c

Does every problem whose solution is quickly verifable also quickly solvable?

If $P = NP$ then any problem that can be verified in polynomial can be solved in polynomial time.

SAT was the one of the first problem to be proved as NP complete. The question is SAT has a polynomial algorithm is equivalent to P vs NP.

Question 2

a

An algorithm is non-deterministic is given a set of input, subsequent runs can yield different results.

`arc(Node, Next)` could have several `Next` node so the result is non-deterministic.

b

Bounded

```
search(Node, s(B)) :- goal(Node).
search(Node, s(B)) :- arc(Node, Next), search(Next, B).
```

Iterative Deepending

```
itSearch(Node) :- bound(B), search(Node, B).

bound(0).
bound(s(B)) :- bound(B).
```

c

Heuristic function $h(n)$

- Estimates the cost of the shortest path from node n to the goal node
- An underestimate if no path between n and the goal is less than $h(n)$

Cost function $c(n)$

- The cost of getting from the current node to node n
- Must be bounded above 0

A-star combines these two ideas to get a cost of $f(n) = h(n) + c(n)$. The frontier is a priority queue ordered by smallest to largest f-value.

d

It is admissible if it will find the shortest solution if there one.

Admissible if:

- branching factor is finite
- cost is bounded above 0
- heuristic is an underestimate

e

- Variables
  - $V = \{X_1, V_2, \ldots, V_n\}$
- Domain: Set of values each variable can take
  - $D = \{R, G, B\}$
- Constraints: A set of tuples of variables and a list of values that the tuple is allowed to take
  - $C = \{(R,G),(R,B),(G,R),(G,B),(B,R),(B,G)\}$

f

1. Generate possible solutions
2. Test to see if its the expected solution
3. If not, go back to 1

I would draw a diagram for this

$(V_1 = ?, V_2 = ?)$

- $(V_1 = R, V_2 = ?)$
  - $(V_1 = R, V_2 = R)$
    - Backtrack!
  - $(V_1 = R, V_2 = G)$
    - Success!

Question 3

a

- Integrity constraint: constraint in the form $false \leftarrow a \wedge b$
- Definite clause: a prolog statement
- Horn clause: A definite clause or an integrity constraint

Integrity Constraints:

```
false :- p.
false :- a, b.
```

Definite Clause:

```
p :- b, c.
p :- q.
a :- r, s.
b.
```

Horn clauses:

```
false :- p.
false :- a, b.
p :- b, c.
p :- q.
a :- r, s.
b.
```

b

- Minimal conflicts:
  - $\{q\}$
  - $\{r, s\}$
- Conflicts:
  - $\{r, s, q\}$

c

Assume that a database of facts contains all possible information. Everything not listed is assumed to be false.

- True:
  - $\{b\}$
- False:
  - Anything else

d

$\{b\}$ is a logical consequence.

$\{a, r, s, q, p, c\}$ is not a logical consequence.

e

A system is nonmonotonic if a conclusion can be invalidated by adding more clauses.

CKA leads to non-monotonicity: true

f

A knowledge base is consistent iff its negation isn't a tautology.

The knowledge base isn't consistent because if you negate it, it's valid in every interpretation of the model.