CS 188 Spring 2021 | Introduction to Artificial Intelligence | Final Review Logic Soluti

Q1. Logic

(a) Prove, or find a counterexample to, each of the following assertions:

(i) If $\alpha \models \gamma$ or $\beta \models \gamma$ (or both) then $(\alpha \wedge \beta) \models \gamma$

True. This follows from monotonicity.

(ii) If $(\alpha \wedge \beta) \models \gamma$ then $\alpha \models \gamma$ or $\beta \models \gamma$ (or both).

False. Consider Consider $\alpha \equiv A$, $\beta \equiv B$, $\gamma \equiv (A \wedge B)$.

(iii) If $\alpha \models (\beta \vee \gamma)$ then $\alpha \models \beta$ or $\alpha \models \gamma$ (or both).

False. Consider $\beta \equiv A$, $\gamma \equiv \neg A$.

(b) Decide whether each of the following sentences is valid, unsatisfiable, or neither.

(i) $Smoke \implies Smoke$

Valid

(ii) $Smoke \implies Fire$

Neither

(iii) $(Smoke \implies Fire) \implies (\neg Smoke \implies \neg Fire)$

Neither

(iv) $Smoke \vee Fire \vee \neg Fire$

Valid

(v) $((Smoke \wedge Heat) \implies Fire)((Smoke \implies Fire) \vee (Heat \implies Fire))$

Valid

(vi) $(Smoke \implies Fire) \implies ((Smoke \wedge Heat) \implies Fire)$

Valid

(vii) $Big \vee Dumb \vee (Big \implies Dumb)$

Valid

(c) Suppose an agent inhabits a world with two states, S and $\neg S$, and can do exactly one of two actions, a and b. Action a does nothing and action b flips from one state to the other. Let S^t be the proposition that the agent is in state S at time t, and let a^t be the proposition that the agent does action a at time t (similarly for b^t).

(i) Write a successor-state axiom for S^{t+1}.

$S^{t+1} \iff [(S^t \wedge a^t) \vee (\neg S^t \wedge b^t)]$.

(ii) Convert the sentence in the previous part into CNF.

Because the agent can do exactly one action, we know that $b^t \equiv \neg a^t$ so we replace b^t throughout. We obtain four clauses:

1: $(\neg S^{t+1} \vee S^t \vee \neg a^t)$

2: $(\neg S^{t+1} \vee \neg S^t \vee a^t)$

3: $(S^{t+1} \vee \neg S^t \vee \neg a^t)$

4: $(S^{t+1} \vee S^t \vee a^t)$

Q2. Disjunctive Normal Form

A sentence is in disjunctive normal form (DNF) if it is the disjunction of conjunctions of literals. For example, the sentence $(A \land B \land \neg C) \lor (\neg A \land C) \lor (B \land \neg C)$ is in DNF.

(a) Any propositional logic sentence is logically equivalent to the assertion that some possible world in which it would be true is in fact the case. From this observation, prove that any sentence can be written in DNF.

Each possible world can be expressed as the conjunction of all the literals that hold in the model. The sentence is then equivalent to the disjunction of all these conjunctions, i.e., a DNF expression.

(b) Construct an algorithm that converts any sentence in propositional logic into DNF. (*Hint*: The algorithm is similar to the algorithm for conversion to CNF.)

A trivial conversion algorithm would enumerate all possible models and include terms corresponding to those in which the sentence is true; but this is necessarily exponential-time. We can convert to DNF using the same algorithm as for CNF except that we distribute $\land$ over $\lor$ at the end instead of the other way round.

(c) Construct a simple algorithm that takes as input a sentence in DNF and returns a satisfying assignment if one exists, or reports that no satisfying assignment exists.

A DNF expression is satisfiable if it contains at least one term that has no contradictory literals. This can be checked in linear time, or even during the conversion process. Any completion of that term, filling in missing literals, is a model.

(d) Apply the algorithms in the previous two parts to the following set of sentences:

$$A \implies B$$
$$B \implies C$$
$$C \implies \neg A$$

The first steps give

$$(\neg A \lor B) \land (\neg B \lor C) \land (\neg C \lor \neg A) .$$

Converting to DNF means taking one literal from each clause, in all possible ways, to generate the terms (8 in all). Choosing each literal corresponds to choosing the truth value of each variable, so the process is very like enumerating all possible models. Here, the first term is $(\neg A \land \neg B \land \neg C)$, which is clearly satisfiable.

(e) Since the algorithm in (b) is very similar to the algorithm for conversion to CNF, and since the algorithm in (c) is much simpler than any algorithm for solving a set of sentences in CNF, why is this technique not used in automated reasoning?

The problem is that the final step typically results in DNF expressions of exponential size, so we require both exponential time AND exponential space.