CS 188 Spring 2021 | Introduction to Artificial Intelligence | Exam Prep 4

Q1. Propositional logic

(a) Consider a propositional model with only four symbols, A, B, C, and D. For each of the following sentences, how many possible worlds make it true?

1. $(A \wedge B) \vee (C \wedge D)$

2. $\neg(A \wedge B \wedge C \wedge D)$

3. $B \Rightarrow (A \wedge B)$

(b) A certain procedure to convert a sentence to CNF contains four steps (1-4 below); each step is based on a logical equivalence. Circle ALL of the valid equivalences for each step.

1. Step 1: drop biconditionals

   a) $(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$

   b) $(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \vee (\beta \Rightarrow \alpha))$

   c) $(\alpha \Leftrightarrow \beta) \equiv (\alpha \wedge \beta)$

2. Step 2: drop implications

   a) $(\alpha \Rightarrow \beta) \equiv (\alpha \vee \neg\beta)$

   b) $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$

   c) $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \wedge \beta)$

3. Step 3: move "not" inwards

   a) $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$

   b) $\neg(\alpha \vee \beta) \equiv (\neg\alpha \vee \neg\beta)$

   c) $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$

4. Step 4: move "or" inwards and "and" outwards

   a) $(\alpha \vee (\beta \wedge \gamma)) \equiv (\alpha \vee \beta \vee \gamma)$

   b) $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$

   c) $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$

(c) A group of Stanford students write a Convert-to-CNF-ish procedure. In their implementation, they simply apply the *first* equivalence (the one labeled "(a)") from each of the four steps in part (b). Show the transformed sentence generated by Convert-to-CNF-ish at each stage, when applied to the input sentence $A \Leftrightarrow (C \vee D)$.

(d) Is the final output of the Convert-to-CNF-ish equivalent to the input sentence in part (c)? If not, give a possible world where the input and output sentences have different values.

Q2. Search, logic, and learning

In this question we consider the problem of searching for the smallest propositional logic sentence ϕ that satisfies some condition $G(\phi)$. (E.g., "find the smallest unsatisfiable sentence containing two distinct symbols.") Recall that a propositional logic sentence is a proposition symbol, the negation of a sentence, or two sentences joined by $\wedge$, $\vee$, $\Rightarrow$, or $\Leftrightarrow$. The proposition symbols are given as part of the problem. The size of a sentence is defined as the sum of the sizes of its logical connectives, where $\neg$ has size 1 and the other connectives have size 2. (It is helpful to think of the sentence as a syntax tree with proposition symbols at the leaves; then the size is the number of edges in the tree. Figure 1(a) shows an example.)

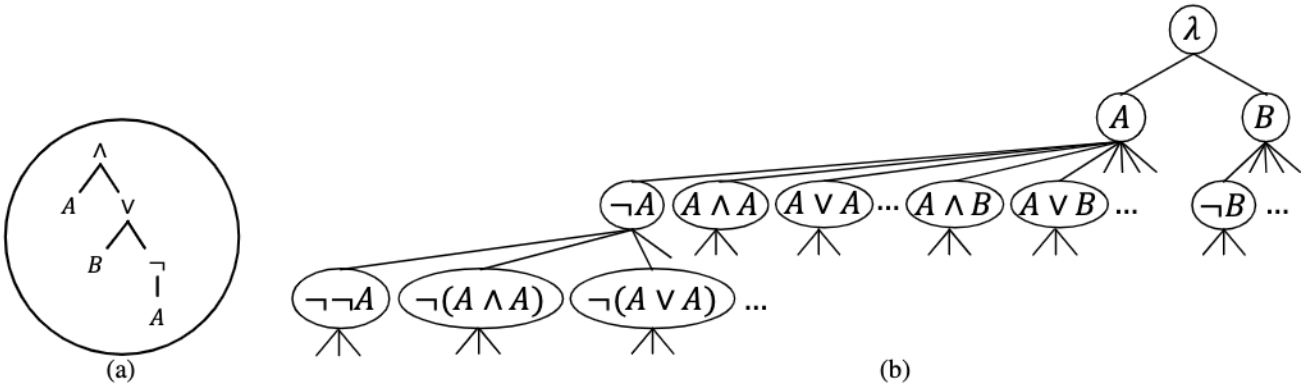

Figure 1: (a) The sentence $A \wedge (B \vee \neg A)$ drawn as a syntax tree with 5 edges. (b) Part of the "parsimonious" search space for sentences with symbols A and B.

(a) Hank Harvard proposes the following problem formulation to explore the search space of sentences:

- There is a dummy start state λ, which is an empty sentence that never satisfies G. The actions applicable in this state simply replace λ by one of the proposition symbols.
- For all other states, the actions take any occurrence of a proposition symbol (call it p) in the sentence and replace it with either $\neg q$ for any symbol q, or $r * s$ where r and s are any symbols and "$*$" is any binary connective.

Buzzy Berkeley agrees with Hank that this set of actions will generate all and only the syntactically valid sentences, but proposes her own formulation:

- For all other states, the actions take any occurrence of a proposition symbol (call it p) in the sentence and replace it with either $\neg p$, or $p * q$ where q is any symbol and "$*$" is any binary connective.

Part of the search space generated by Buzzy's formulation is shown in Figure 1(b). Buzzy claims that her formulation is more efficient than Hank's but still generates all and only the syntactically valid sentences. Is she right? Explain.

(b) Assuming there are n symbols, give a $O()$-expression for the branching factor at depth d of Buzzy's search tree.

(c) Using your $O()$ answer for the branching factor, give a $O()$-expression for the number of nodes at depth d of Buzzy's search tree.

(d) We will say that G is a *semantic* condition if $G(\phi)$ depends only on the *meaning* of ϕ, in the following sense: if two sentences ϕ and ψ are logically equivalent, then $G(\phi) = G(\psi)$. Not wishing to be outdone by Buzzy, Hank now makes the following claim: Whenever Buzzy's search space contains a solution for a semantic G then so does the reduced search space using only $\neg$, $\wedge$, and $\vee$. Is he right? Explain.

(e) Suppose we are running a uniform cost graph search, which maintains the property that $g(n)$ for every node in the frontier is the optimal cost to reach n from the root. In a standard graph search, we discard a newly generated state without adding it to the frontier if and only if the *identical* state is already in the frontier set or reached set. Assuming we have a semantic condition G, when is it possible to discard a newly generated state? Check all that apply.

- ☐ Sentences that are identical to a sentence already in the frontier or reached set
- ☐ Sentences that logically entail a sentence already in the frontier or reached set
- ☐ Sentences that are logically equivalent to a sentence already in the frontier or reached set
- ☐ Sentences that are logically entailed by a sentence already in the frontier or reached set