CS 188
Spring 2023

MT Review Logic Solutions

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. $(A \wedge B) \vee (C \wedge D)$ 7 (4 for $A \wedge B$, 4 for $C \wedge D$, minus 1 for the model that satisfies both).

2. $\neg(A \wedge B \wedge C \wedge D)$ 15 — it's the negation of a sentence with 1 model.

3. $B \Rightarrow (A \wedge B)$ 12 — it's true when B is false (8) and when B is true and A is true (4).

(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 $A \Leftrightarrow (C \vee D)$.

$A \Leftrightarrow (C \vee D)$
$(A \Rightarrow (C \vee D)) \wedge ((C \vee D) \Rightarrow A)$
$(A \vee \neg(C \vee D)) \wedge ((C \vee D) \vee \neg A)$
$(A \vee (\neg C \wedge \neg D)) \wedge ((C \vee D) \vee \neg A)$
$(A \vee \neg C \vee \neg D)) \wedge (C \vee D \vee \neg A)$

(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.

No.
A counterexample is any model where the two sentences have different truth values. The first clause in the final sentence says $(C \wedge D) \Rightarrow A$ rather than $(C \vee D) \Rightarrow A$. So countereamples are $\{A = false, C = true, D = false\}$ and $\{A = false, C = false, D = true\}$.

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.)

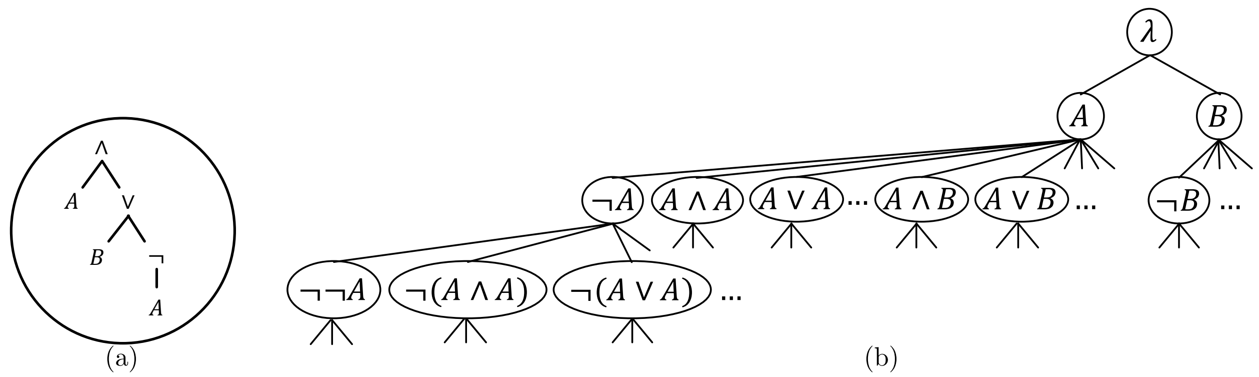

Figure 1: (a) The sentence $A \wedge (B \vee \neg A)$ drawn as a syntax tree with 5 edges. (b) Part of the "parsimonious" search space for sentences with symbols A and B.

(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.

Yes. It is more efficient because it has a smaller branching factor and eliminates redundant copies of sentences. For any sentence with an occurrence of p in some location, there is another sentence identical except that the occurrence of p is replaced by an occurrence of q. (Essentially this is the inductive hypothesis in the proof.) Hence there is no need to replace p by other symbols.

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

Each step adds no more than 1 symbol occurrence; so at depth d there are up to d symbols. There is one way to negate a symbol and n ways to add each of four binary connectives, so the branching factor is $O(4nd)$.

(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.

The off-the-shelf answer of b^d doesn't work because b is not constant. The number of nodes is the product of the branching factors, so $O((4n)^d \cdot d!)$.

(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.
Yes. Every propositional logic sentence can be expressed in CNF, which uses only $\neg$, $\wedge$, and $\vee$.

(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

Identical sentences can automatically be discarded by the original definition of graph search. Additionally, because our goal is to find the smallest propositional logic sentence ϕ that satisfies $G(\phi)$, if we know that G is a semantic condition, then any sentence that has already been expanded must not have met that condition; therefore, any new sentence that is logically equivalent to would also not satisfy G, so we can safely discard those as potential states. Also, since UCS expands nodes from smallest to largest backward cost, any sentence on the frontier that is logically equivalent to a new sentence will have a cost less than or equal to the cost of the new sentence; therefore, we would never choose the new sentence anyway as the best choice, so we can discard it. Note that entailing a sentence in the frontier / reached set is not sufficient, because logical entailment does not mean they are logically equivalent such that they would both satisfy the semantic condition.