CS221 Exam 2 Solutions

Winter 2021

Please read all of the following information before starting the exam:

- This test has 3 problems on 18 pages for a total of 100 possible points.
- **You must submit your exam by 2:30pm PT on Friday, March 19th**. Please be aware of Daylight Saving Time in your calculation if you're in a different time zone.
- Note that different questions are worth different amounts of points. Budget your time accordingly!
- Keep your answers precise and concise. We may award partial credit so show all your work clearly and in order.
- Don't spend too much time on one problem. Read through all the problems carefully and then plan your time accordingly.
- If you are unsure about a problem statement when taking the exam, state your assumptions **within your response to that problem**. We will take all reasonable assumptions into account when grading.
- This exam is open-book; you may use any inanimate resources, including the course website. You may consult only inanimate sources. You may not consult or collaborate with anyone about the questions. Such collaboration is a violation of the Honor Code.
- Good luck!

Problem	Part	Max Score	Score
1	a	8	
	b	6	
	c	9	
	d	4	
	e	3	
	Total	**30**	
2	a	8	
	b	25	
	c	20	
	Total	**53**	
3	a	1	
	b	1	
	c	8	
	d	7	
	Total	**17**	

Total Score: $\square + \square + \square = \square$

0. **Honor Code** (*0 points*) Please write or type down the honor code below and sign your name. Your exam will not be graded if this question is not completed.

"I have carefully read the instructions on Page 1. I will not consult or collaborate with anyone about the questions. Such collaboration is a violation of the Honor Code."

1. CA Assignments (*30 points*) Every quarter, the Stanford computer science department assigns graduate students as course assistants (CAs). Students who wish to serve as CAs fill out an application in which they can list the classes they'd like to CA for. After the application due date, the department matches applicants to courses, taking into account student preferences as well as how many course assistants each class needs.

Here's the formal CA-assignment problem setup:

1. There are n students $S_1, \dots, S_n$ who apply for CAships.
2. There are m courses $C_1, \dots, C_m$ that have CA openings.
3. Each student S_i specifies arbitrary non-negative preferences $P_1^{(i)}, \dots, P_m^{(i)} \geq 0$ for each of the m classes. A large preference value $P_j^{(i)}$ means student S_i really wants to CA for class C_j, and a preference value of 0 for $P_j^{(i)}$ means student S_i does not want to CA for class C_j.

The CA-matching process must adhere to the following requirements:

1. Each course C_i can have a maximum of M_i course assistants.
2. Every student must be matched to exactly one class for which they have specified a *positive* preference (assume each student has at least one such preference).

a. (*8 points*) We can model the CA-matching process with a CSP with n variables, one for each student $S_1, \dots, S_n$. Our CSP should find the *maximum weight assignment*, where the weights are determined by student preferences.

Finish the specification of this CSP by stating the domains of each variable and the factors needed.

You may define any notation/helper functions to help you concisely express your answers below.

- **Variables (Already given):** We have n variables for the students $S_1, \dots, S_n$.
- **Domains (how large is each and what are the values?):**

  Solution The domain for each student is of cardinality m with values $\{C_1, \dots, C_m\}$.

- **Factors (state the arity of each and write them as functions from variables to scalars):**

Solution We have 2 sets of factors. The first group encodes the maximum CA openings for each course. This can be written as n-ary factors $f_1, \dots, f_m$, where

$$f_j(S_1, \dots, S_n) = \left[\sum_{i=1}^{n} [S_i = C_j] \le M_j\right].$$

The second set encodes individual student preferences for which class they'd like to CA. These are unary factors $g_1, \dots, g_n$, where

$$g_i(S_i) = P^{(i)}_{S_i},$$

which is the preference of the ith student to CA for class S_i.

b. (*6 points*) We imagine a small setting of this problem for 3 students S_1, S_2, S_3 and 3 courses C_1, C_2, C_3. The student preferences are given by the following table:

	C_1	C_2	C_3
S_1	3	0	0
S_2	2	1	3
S_3	5	3	0

Additionally, classes C_1 and C_2 can have a maximum of 1 CA each, and class C_3 can have at most 2 CAs.

Apply the CSP you designed to this small setting and enforce arc-consistency amongst its variables. In particular, write out each variable and its domain *after* arc-consistency has been enforced. For example, if you have a variable X_i with a domain $\{a, b, c\}$ after enforcing arc-consistency, you should write

$$X_i : \{a, b, c\}.$$

Solution

$$S_1 : \{C_1\}$$
$$S_2 : \{C_3\}$$
$$S_3 : \{C_2\}$$

A commonly seen incorrect answer was not enforcing arc consistency recursively and thus thinking that domain of S_2 is $\{C_2, C_3\}$ rather than $\{C_3\}$. Remember that *arc consistency* is a definition, not just an algorithm that you apply once.

c. (*9 points*) We leave the above example and revisit our *general* CSP. Recall that we seek the maximum-weight assignment as the solution for our CSP.

Answer the following true/false questions. You will receive 1 point for choosing the correct option and 2 points for a correct 1-2 sentence justification for your choice.

If you print out the exam then you can clearly circle your choice of True or False, otherwise please clearly write "True" or "False" on your submission.

(i) (3 points) **True/False:** The least constrained value (LCV) heuristic would be a useful optimization for our CA-assignment CSP.

Justification:

Solution **False.** LCV is a useful optimization when *all* of our factors are constraints. Since student preferences are arbitrary non-negative values, we need to try all of the consistent values anyway.

Most wrong answers failed to interpret LCV properly, and a few answers assumed there are only constraints in the CA CSP.

(ii) (3 points) **True/False:** The most constrained variable (MCV) heuristic would be a useful optimzation for our CA-assignment CSP.

Justification:

Solution **True.** MCV is a useful optimization when *some* of our factors are constraints. Since we have constraints for the maximum number of CAs for a particular class, MCV can help.

Most wrong answers failed to interpret MCV properly.

(iii) (3 points) **True/False:** If we use the ICM algorithm to solve our CA-assignment CSP, everytime we modify a single variable assignment our factor recomputation will be on the order of n (recall that n is the number of students applying for a CA assignment).

Justification:

Solution **True.** While in general for ICM we only need to compute factors involving the single variable whose assignment we changed, the constraints on the maximum number of classes (which depends on all n variables) will need to be recomputed.

A number of students had different interpretations of what "factor recomputation" meant. Full credit was given to solutions that correctly gave the runtime complexity for some aspect/interpretation of the ICM algorithm.

(iv) (**Bonus:** 3 points) **True/False:** If we use beam search with different beam sizes k to solve our CA-assignment CSP, our solution's assignment weight will always *increase* as we increase the beam size k.

Justification:

Solution **False.** Consider going from $k = 1$ (greedy) to $k = 2$. The greedy solution might be the globally optimal assignment, but when $k = 2$ we may find more partially optimal solutions as we expand more paths that cause us to drop the greedy solution from our beam. We are only guaranteed a global optimum with an unbounded beam size.

This was a tricky question which is why we labeled it as bonus. A common mistake was to state that the weight will never decrease or the weight will stay the same. This is actually not true even though intuitively it seems like it should be!

d. (*4 points*) Unfortunately, in the real CA-matching process at Stanford, students aren't guaranteed a CAship if there aren't enough openings. Right now, your CSP only permits assignments that grant all students CAships. **Explain how you would modify your CSP from part a. to allow for the possibility that some students aren't matched to a course. You should encode the (realistic) assumption that not receiving a CAship is the least-preferable assignment for the student.**

(Note that students can still give a preference of 0 for a class if they do not want to be a CA for that class, and your modification should not prohibit this.)

Explanation of CSP modification:

Solution A simple way to do this is to extend the domain for each student to include a non-assignment value, which we can denote $\varnothing$. We can then augment the unary factors that encode student preferences to output a positive value for $\varnothing$ that is smaller then all of the student's positive preferences.

Common mistakes that the teaching staff saw include:

- Giving negative preference value to "no assignment" class (factors must be non-negative);
- Not preserving 0 preference for classes that a CA doesn't want to teach.

e. (*3 points*) Even after performing the modification from part **d.**, another shortcoming of our CSP is that the maximum weight assignment can be unfairly influenced by the students. Explain one way in which students are incentivized *not* to report their honest CA preferences in order to manipulate the maximum weight CSP assignment.

Explanation:

Solution Here's one problem: since there is no restriction on the preferences values other than they are positive, a student can specify arbitrarily large preference values on their application. This will skew the CSP solver to prefer solutions that grants this student their top choice. One heuristic to address this is to normalize the preference values so that they are on the same scale all students. For example, you could limit students to ranking 10 classes on a scale from 1 to 10, 10 being their most desirable class to CA and 1 being their least.

This was an intentionally open-ended question, and we were pleased with the variety of good answers we saw!

2. Two Coins (*53 points*)

Consider a gambling machine with two coins X and Y inside.

At each time step t, the machine secretly selects a coin $C_t \in \{X, Y\}$ with probability $\lambda \in (0,1)$ of being the same selection as time $t-1$. The machine then tosses the selected coin. You can observe whether the result $O_t \in \{H, T\}$ is Head (H) or Tail (T), but you can't observe which coin was selected.

At time $t = 1$, the machine starts by selecting coin X with probability $\lambda_0 \in (0,1)$ and coin Y with probability $1 - \lambda_0$. The two coins show head with probability $p_X, p_Y \in (0,1)$. You decide to model this gambling machine as an HMM.

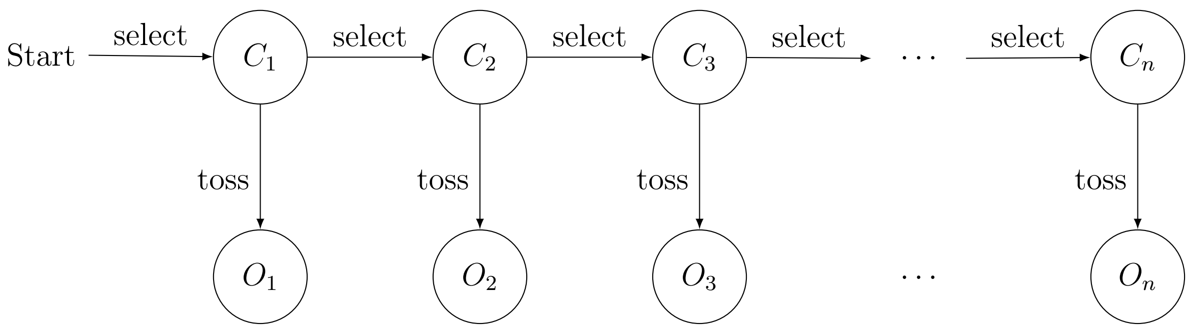


Figure 1: An HMM for the gambling machine.

a. (*8 points*) **Setup**

(i) [4 points] Fill in the table for the transition probability $P(C_{t+1}|C_t)$:

C_t \ C_{t+1}	X	Y
X		
Y		

Solution Answer:

C_t \ C_{t+1}	X	Y
X	λ	$1-\lambda$
Y	$1-\lambda$	λ

Most wrong answers used λ_0 or assumed fair coin.

(ii) [4 points] Fill in the table for the emission probability $P(O_t|C_t)$:

O_t C_t	H	T
X		
Y		

Solution Answer:

O_t C_t	H	T
X	p_X	$1 - p_X$
Y	p_Y	$1 - p_Y$

Most wrong answers assumed fair coin.

b. (*25 points*) **Inference**

You wish to use the forward-backward algorithm to guess which coin was selected at each step. Note that in this part, we assume that all the HMM parameters (i.e. $\lambda_0, \lambda, p_X, p_Y$) are known to you.

Let's play with the toy scenario where the machine stops after two steps. You observe that the results of the two tosses are $\{H, H\}$.

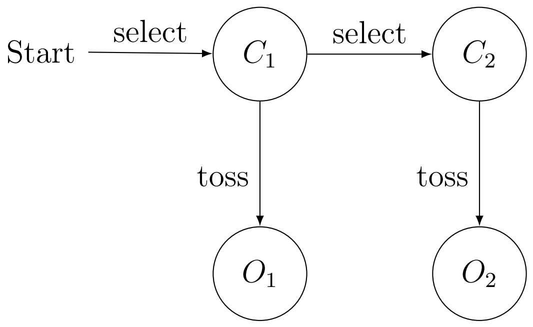
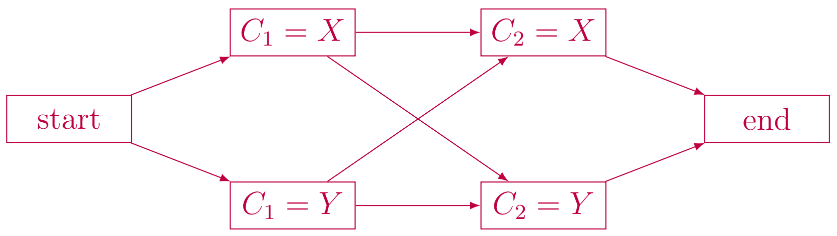


Figure 2: A toy scenario: The machine stops after two steps

(i) [4 points] Please draw the lattice representation of this two-step HMM (No need to label the edge weights).

Solution The lattice representation (correct nodes 2pts, correct arrows 2pts):



A commonly seen mistake was the missing start/end states.

(ii) [4 points] What is the weight of the edge going from $C_1 = X$ to $C_2 = Y$ in the lattice representation? Your answer should be expressed in terms of one or more parameters from $\{\lambda_0, \lambda, p_X, p_Y\}$.

Solution The transition probability is $P(C_2 = Y|C_1 = X) = 1 - \lambda$ and the emission probability is $P(O_2 = H|C_2 = Y) = p_Y$, therefore the edge weight is their product:

$$(1-\lambda)p_Y$$

(iii) [12 points] You start to run the forward-backward algorithm by computing the values of the forward messages $F_i(c)$ and backward messages $B_i(c)$ (where, $i \in \{1, 2\}$ indexes the time step, and $c \in \{X, Y\}$ denotes the coin selection).

① [4 points] Please write down the values of $F_1(X), F_1(Y), B_2(X), B_2(Y)$.

Solution

$$F_1(X) = \lambda_0 p_X$$
$$F_1(Y) = (1-\lambda_0)p_Y$$
$$B_2(X) = 1$$
$$B_2(Y) = 1$$

A frequently seen mistake was missing the p_X (or p_Y) term in $F_1(X)$ (or $F_1(Y)$).

② [8 points] Please write down the values of $F_2(X), F_2(Y), B_1(X), B_1(Y)$ in terms of $F_1(c_1), B_2(c_2)$ and the weights of the edges going from $C_1 = c_1$ to $C_2 = c_2$ (denoted as $w(c_1, c_2)$), where $c_1, c_2 \in \{X, Y\}$.

Solution

$$F_2(X) = w(X,X) \cdot F_1(X) + w(Y,X) \cdot F_1(Y)$$
$$F_2(Y) = w(X,Y) \cdot F_1(X) + w(Y,Y) \cdot F_1(Y)$$
$$B_1(X) = w(X,X) \cdot B_2(X) + w(X,Y) \cdot B_2(Y)$$
$$B_1(Y) = w(Y,X) \cdot B_2(X) + w(Y,Y) \cdot B_2(Y)$$

(iv) [5 points] The gambling machine now asks you to guess which coin was picked at time $t = 1$. Given the current observation, under which condition should you predict coin X as the answer? Formulate this condition with an inequality using $F_i(\cdot)$s and $B_i(\cdot)$s ($i = 1, 2$). Briefly justify your answer.

Solution

$$\text{Probability of coin X: } P(C_1 = X|O) \propto S_1(X) = F_1(X)B_1(X)$$

$$\text{Probability of coin Y: } P(C_1 = Y|O) \propto S_1(Y) = F_1(Y)B_1(Y)$$

$$\therefore \text{The condition is: } F_1(X)B_1(X) > F_1(Y)B_1(Y)$$

c. (*20 points*) **Learning**

Now assume that you know how the machine works (i.e. λ_0, λ are known), but you have no idea about the coins (i.e. p_X, p_Y are unknown). You want to use the EM algorithm to estimate p_X and p_Y.

(i) [4 points] Which one of the following algorithms can NOT be used for posterior inference in the E-step? No justification required.

(a) Gibbs sampling

(b) Forward-backward algorithm

(c) Maximum likelihood

(d) Particle filtering

Solution (c)

(c) is a parameter learning algorithm. The other three are inference algorithms.
A frequently seen incorrect answer was (a) Gibbs sampling.

(ii) [12 points] Let's consider the scenario where the machine stops after three steps. You observe that the results of the three tosses are $o = \{H, H, T\}$.

① [4 points] The E-step computes the posterior probability $q_i(c), i \in \{1, 2, 3\}, c \in \{X, Y\}$ of the coin selections. Please rewrite $q_2(X)$ as a conditional probability (Hint: No computation needed. Your answer should be a 1-line equation that includes the observation o and the parameters p_X, p_Y). No justification required.

Solution

$$q_2(X) = P(C_2 = X | O = o; \{p_X, p_Y\})$$

Mistakes that the teaching staff saw include:

- Skipping the p_X, p_Y parameters;
- Expanding the conditional probability incorrectly.

② [8 points] In one EM iteration, suppose you already get the values of $q_i(c)$ from the E-step (shown in the Table below). What is the value of p_X and p_Y after the M-step? Show your work.

c	$q_1(c)$	$q_2(c)$	$q_3(c)$
X	0.1	0.5	0.3
Y	0.9	0.5	0.7

Solution Answer: $p_X = 2/3, \ p_Y = 2/3$

$$\frac{p_X}{1-p_X} = \frac{0.1+0.5}{0.3} = 2, \quad \therefore p_X = \frac{2}{3}$$
$$\frac{p_Y}{1-p_Y} = \frac{0.9+0.5}{0.7} = 2, \quad \therefore p_Y = \frac{2}{3}$$

A commonly seen incorrect answer was $p_X = 0.3, p_Y = 0.7$, which results from directly summing up and normalizing the two rows in the table.

(iii) [4 points] The gambling machine asks you to guess which coin was picked at time $t = 1$. How will you utilize the result of the EM algorithm to make your prediction? Describe your answer with 1-2 sentences (Hint: No extra computation needed).

Solution Predict coin X if $q_1(X) > q_1(Y)$, otherwise predict coin Y.
Some frequently seen incorrect answers included comparing p_X vs p_Y, or $F_1(X)$ vs $F_1(Y)$.

3. Alexar™ (*17 points*)

You and your friends are building Alexar, a personal assistant capable of natural language inference powered by first-order logic. You are responsible for writing unit tests to verify that Alexar behaves as expected.

For simplicity, consider a setting with only three objects (which are all people in this case): Alice, Bob, and Carol, represented by the three constant symbols `Alice`, `Bob`, and `Carol`, respectively. Consider only the following predicates:

- $Adult(x)$, which means x is an adult
- $Child(x)$, which means x is a child
- $Fever(x)$, which means x has a fever
- $Flu(x)$, which means x has the flu

Alexar is initialized with the following knowledge base:

$$\begin{aligned} KB_0 = \{ & Adult(\texttt{Alice}), \\ & Adult(\texttt{Bob}), \\ & Child(\texttt{Carol}), \\ & \forall x. \big(Adult(x) \leftrightarrow \neg Child(x)\big), \\ & \forall x. Fever(x) \rightarrow Flu(x) \} \end{aligned}$$

a. (*1 point*) You tell Alexar: "If some adult has the flu, then every adult has the flu." Alexar translates this into a formula f_1 and adds it to the knowledge base. Which of the following is a valid expression of f_1?

(A) $\exists x. \big(Adult(x) \wedge Flu(x) \rightarrow \forall y. Adult(y) \wedge Flu(y)\big)$

(B) $\exists x. \big(Adult(x) \wedge Flu(x) \rightarrow \forall y. Adult(y) \rightarrow Flu(y)\big)$

(C) $\big(\exists x. Adult(x) \wedge Flu(x)\big) \rightarrow \big(\forall y. Adult(y) \wedge Flu(y)\big)$

(D) $\big(\exists x. Adult(x) \wedge Flu(x)\big) \rightarrow \big(\forall y. Adult(y) \rightarrow Flu(y)\big)$

Solution D.

b. (*1 point*) Now, Alexar's knowledge base $KB_1 = KB_0 \cup \{f_1\}$. More specifically,

$$\begin{aligned}KB_1 = \{&Adult(\texttt{Alice}),\\ &Adult(\texttt{Bob}),\\ &Child(\texttt{Carol}),\\ &\forall x.\big(Adult(x) \leftrightarrow \neg Child(x)\big),\\ &\forall x.Fever(x) \rightarrow Flu(x),\\ &\textsc{if some adult has the flu, then every adult has the flu}\}.\end{aligned}$$

You then tell Alexar: "Every adult has the flu, or no adult has the flu." Alexar translates this into a formula f_2 and adds it to the KB. Do you expect the set of models representing the KB to change given the update from f_2? Select the true statement.

(A) $\mathcal{M}(KB_1 \cup \{f_2\}) = \mathcal{M}(KB_1)$

(B) $\mathcal{M}(KB_1 \cup \{f_2\}) = \emptyset$.

(C) $\emptyset \subsetneq \mathcal{M}(KB_1 \cup \{f_2\}) \subsetneq \mathcal{M}(KB_1)$.

(D) Insufficient information given to determine the relationship between $\mathcal{M}(KB_1 \cup \{f_2\})$ and $\mathcal{M}(KB_1)$.

Solution A. f_2 is logically equivalent to f_1, so KB_1 entails f_2. Using the definition of entailment in terms of models, we can derive that $\mathcal{M}(KB_1 \cup \{f_2\}) = \mathcal{M}(KB_1)$.

c. (*8 points*) Suppose you undo adding f_2 to Alexar's knowledge base. In other words, Alexar's knowledge base is still KB_1. For your reference,

$$\begin{aligned} KB_1 = \{ & \mathit{Adult}(\texttt{Alice}), \\ & \mathit{Adult}(\texttt{Bob}), \\ & \mathit{Child}(\texttt{Carol}), \\ & \forall x.\big(\mathit{Adult}(x) \leftrightarrow \neg \mathit{Child}(x)\big), \\ & \forall x.\mathit{Fever}(x) \rightarrow \mathit{Flu}(x), \\ & \text{IF SOME ADULT HAS THE FLU, THEN EVERY ADULT HAS THE FLU}\}. \end{aligned}$$

From now on, every time you tell Alexar a new claim, Alexar will add the claim to the KB and further update the KB via the forward inference algorithm.

Hint: Keep in mind that we are building unit tests to verify the logical behavior of our AI system, and not the logical behavior of our own intelligence. Therefore, follow the algorithms taught in class instead of making logical derivations of your own.

(i) [4 points] In order to run forward inference on first-order logic, Alexar first performs propositionalization on all formulas in the knowledge base. Your task here is to **manually propositionalize formulas in** KB_1 in order to verify Alexar's behavior.

Note: For the sake of notational simplicity, we will only seek to eliminate quantifiers and variables. Predicates are acceptable as long as their arguments are not variables. For instance, $\mathit{Adult}(\texttt{Alice})$ is considered propositionalized; no need to rewrite it as $\mathit{AliceIsAdult}$. The same goes for $\mathit{Adult}(\texttt{Bob})$ and $\mathit{Adult}(\texttt{Carol})$.

[Example] $\forall x.\big(\mathit{Adult}(x) \leftrightarrow \neg \mathit{Child}(x)\big)$

$$\begin{aligned} & \big(\mathit{Adult}(\texttt{Alice}) \leftrightarrow \neg \mathit{Child}(\texttt{Alice})\big) \\ \wedge & \big(\mathit{Adult}(\texttt{Bob}) \leftrightarrow \neg \mathit{Child}(\texttt{Bob})\big) \\ \wedge & \big(\mathit{Adult}(\texttt{Carol}) \leftrightarrow \neg \mathit{Child}(\texttt{Carol})\big) \end{aligned}$$

Hint: Do *not* perform any logical derivations of your own. Follow the procedure of propositionalization.

① $\forall x.\mathit{Fever}(x) \rightarrow \mathit{Flu}(x)$

Solution

$\big(\mathit{Fever}(\texttt{Alice}) \rightarrow \mathit{Flu}(\texttt{Alice})\big) \wedge \big(\mathit{Fever}(\texttt{Bob}) \rightarrow \mathit{Flu}(\texttt{Bob})\big) \wedge \big(\mathit{Fever}(\texttt{Carol}) \rightarrow \mathit{Flu}(\texttt{Carol})\big)$

A common error we saw is not using parentheses (-1 point). Since $\wedge$ has higher precedence than $\rightarrow$. the parentheses around each implication are required.

② IF SOME ADULT HAS THE FLU, THEN EVERY ADULT HAS THE FLU

Solution

$$\big(\mathit{Adult}(\texttt{Alice}) \wedge \mathit{Flu}(\texttt{Alice})\big) \vee \big(\mathit{Adult}(\texttt{Bob}) \wedge \mathit{Flu}(\texttt{Bob})\big) \vee \big(\mathit{Adult}(\texttt{Carol}) \wedge \mathit{Flu}(\texttt{Carol})\big)$$
$$\rightarrow \big(\mathit{Adult}(\texttt{Alice}) \rightarrow \mathit{Flu}(\texttt{Alice})\big) \wedge \big(\mathit{Adult}(\texttt{Bob}) \rightarrow \mathit{Flu}(\texttt{Bob})\big) \wedge \big(\mathit{Adult}(\texttt{Carol}) \rightarrow \mathit{Flu}(\texttt{Carol})$$

A common error we saw is making logical derivations beyond the scope of propositionalization.

One such example is leaving $\mathit{Adult}(\cdot)$ and `Carol` out completely by incorporating existing formulas in the KB.

Another example is splitting the above expression into three implications joined with $\wedge$. This transformation requires inference: it requires converting the above expression into something called conjunctive normal form (CNF, not covered in class), and an inference rule called resolution (not covered in class, either).

(ii) [4 points] At this point, you tell Alexar: "Alice has a fever." Alexar translates the sentence into f_3 and adds f_3 to KB_1. Alexar then runs forward inference on the propositionalized KB, with propositional modus ponens as its only inference rule, and yields an updated knowledge base KB_2.

Which of the following formulas do you expect to see in KB_2? Answer yes/no for each formula.

Note: If the KB contains a formula of the form $p \wedge q$, the KB contains both formulas p and q.

Hint: Do *not* perform any logical derivations of your own. Follow the procedure of forward inference algorithm using modus ponens. Pay attention to the propositionalized formulas.

(A) $\mathit{Fever}(\texttt{Alice})$

(B) $\mathit{Flu}(\texttt{Alice})$

(C) $\mathit{Fever}(\texttt{Bob})$

(D) $\mathit{Flu}(\texttt{Bob})$

Solution Yes for (A) and (B); no for (C) and (D).

(A) is the correct translation of "Alice has a fever".

From (i①), we see the KB contains $\mathit{Fever}(\texttt{Alice}) \rightarrow \mathit{Flu}(\texttt{Alice})$. By modus ponens, we derive (B).

It might be tempting to select (D) since it indeed holds true, but propositional modus ponens actually can't derive (D). To see this, we examine our result from (i②):

$$\big(\mathit{Adult}(\texttt{Alice}) \wedge \mathit{Flu}(\texttt{Alice})\big) \vee \big(\mathit{Adult}(\texttt{Bob}) \wedge \mathit{Flu}(\texttt{Bob})\big) \vee \big(\mathit{Adult}(\texttt{Carol}) \wedge \mathit{Flu}(\texttt{Carol})\big)$$
$$\rightarrow \big(\mathit{Adult}(\texttt{Alice}) \rightarrow \mathit{Flu}(\texttt{Alice})\big) \vee \big(\mathit{Adult}(\texttt{Bob}) \rightarrow \mathit{Flu}(\texttt{Bob})\big) \vee \big(\mathit{Adult}(\texttt{Carol}) \rightarrow \mathit{Flu}(\texttt{Carol})\big)$$

As we've seen on p.26 of `inference-rules`, although we do have $\mathit{Adult}(\texttt{Alice}) \wedge \mathit{Flu}(\texttt{Alice})$ in the KB, modus ponens can't derive the consequent given the antecedent is connected with $\vee$'s. Therefore, we wouldn't be able to derive (D).

d. (*7 points*) Alexar now has KB_2 as its knowledge base. You now ask Alexar: "Does everyone have the flu?" Alexar translates the sentence "everyone has the flu" into f_4. Determine Alexar's expected response by considering satisfiability:

(i) [3 points] Is $KB_2 \cup \{\neg f_4\}$ satisfiable? Answer yes/no.

If you answered yes, show that there exists a model that satisfies $KB_2 \cup \{\neg f_4\}$ by completing the following table with 0 or 1. Otherwise, leave it blank.

x	$Adult(x)$	$Child(x)$	$Fever(x)$	$Flu(x)$
`Alice`	1			
`Bob`	1			
`Carol`		1		

Solution Yes.

x	$Adult(x)$	$Child(x)$	$Fever(x)$	$Flu(x)$
`Alice`	**1**	0	1	1
`Bob`	**1**	0	0 or 1	1
`Carol`	0	**1**	0 or 1	0

Most wrong answers draw incorrect conclusions on satisfiability.

(ii) [3 points] Is $KB_2 \cup \{f_4\}$ satisfiable? Answer yes/no.

If you answered yes, show that there exists a model that satisfies $KB_2 \cup \{f_4\}$ by completing the following table with 0 or 1. Otherwise, leave it blank.

x	$Adult(x)$	$Child(x)$	$Fever(x)$	$Flu(x)$
`Alice`	1			
`Bob`	1			
`Carol`		1		

Solution Yes.

x	$Adult(x)$	$Child(x)$	$Fever(x)$	$Flu(x)$
`Alice`	**1**	0	1	1
`Bob`	**1**	0	0 or 1	1
`Carol`	0	**1**	0 or 1	1

Most wrong answers draw incorrect conclusions on satisfiability.

(iii) [1 point] Select the statement that correctly characterizes the relationship between KB_2 and f_4 as well as Alexar's expected response:

(A) KB_2 entails f_4; Alexar is expected to say "Yes, everyone has the flu."

(B) KB_2 contradicts f_4; Alexar is expected to say "No, not everyone has the flu."

(C) f_4 is contingent; Alexar is expected to say "I'm not sure."

Solution (C). We follow the decision tree on p.36 of `propositional-logic-semantics`.