National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 8: Logical Agents II

Issued: Week 10 Discussion in: Week 11

Important Instructions:

1. **Tutorial Assignment 8** consists of **Question 1** from this tutorial.
2. Your solution(s) must be TYPE-WRITTEN, though diagrams may be hand-drawn.
3. You must submit your solution(s) via **Canvas > Assignments**, satisfying the deadlines:
   - Pre-tutorial 8 submission by **Week 10 Sunday, 2359 hrs**.
   - Post-tutorial 8 submission by **Week 11 Friday, 2359 hrs**.
4. You must make both submissions for your assignment score to be counted.

Refer to **Appendix A** for notes on Knowledge Bases, and **Appendix B** for Propositional Logic Laws.

1. Consider an instance of the Vertex Cover problem given in Figure 1. In the Vertex Cover problem, we are given a graph $G = \langle V, E \rangle$. We say that a vertex v covers an edge $e \in E$ if v is incident on the edge e. We are interested in finding a *vertex cover*; this is a set of vertices $V' \subseteq V$ such that every edge is covered by some vertex in V'. In what follows, you may **only** use variables of the form x_v, where $x_v = 1$ if v is part of the vertex cover and is 0 otherwise.

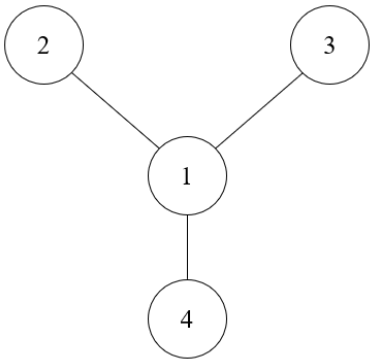


Figure 1: Graph for the Vertex Cover in Part (b)

Note: when writing the constraints, you may only use standard logical and set operators: $\forall$, $\exists$, $\vee$, $\wedge$ and $x \in X$, $X \subseteq Y$.

(a) Write down the vertex cover constraints as logical statements, as well as the size constraints in the case that the vertex cover is of size $k = 1$. Express your answers in CNF.

(b) Apply the resolution algorithm in order to prove that the vertex 1 must be part of the vertex cover; again, assume that the cover in Figure 1 must be of size $k = 1$.

2. **[AY23/24 S1 Final]** In this question, you are tasked to create a logical agent to solve a simplified version of the Minesweeper game (depicted on the right), which is a logic puzzle video game that is generally played on personal computers.

   The game features an n by m grid of covered cells, with k hidden ***mines*** scattered throughout the grid. The game rules are as follows.

   - When a player uncovers a cell at (r, c), where r is the row index and c is the column index, if that cell ***does not*** contain a mine, it is a ***safe*** cell, and will reveal how many mines are in the 8 cells adjacent to it (i.e., on uncovering a safe cell at (r, c), the number of mines in the cells at coordinates (r+1, c), (r+1, c+1), (r, c+1), (r–1, c+1), (r–1, c), (r–1, c–1), (r, c–1), and (r+1, c–1) will be revealed).
   - When a player uncovers a cell that is not a safe cell (i.e., it contains a mine), it will detonate the mine and cause the player to lose the game.
   - The goal of the game is to uncover every safe cell on the grid (note that there will always be $nm - k$ safe cells given k mines). Once all the safe cells are uncovered (without detonating a mine), the player wins the game.

   Consider the following example states on a 3 by 3 grid, where there is 1 mine.

   State s_1

A	B	C
D	1	E
F	G	H

   In this example state the player has uncovered the middle cell (only), which has revealed the value 1, meaning that there is exactly one (1) mine in the adjacent covered cells marked A, B, C, D, E, F, G, and H.

   State s_3

1	**B**	C
1	1	E
F	G	H

   If two other cells have been uncovered without losing. The resultant state is as shown. Here, we deduce that the mine is – as there must be 1 mine adjacent to A, but both the centre cell and D are safe – at B!

   For this simplified version of the game, we will also assume the following.

   - Only cells that the player uncovers will become uncovered; no additional cells will become uncovered except the one cell chosen to be uncovered by the player. (Note that in the original version of the game, certain cells would also be automatically uncovered for the player. However, this is not the case in the simplified version.)
   - When a cell is uncovered, maximally ***one*** neighbouring cell will contain a mine.
   - We do not know the value k (i.e., do not include any rules about k).
   - You may only use the Boolean variables $X_{i,j}$, which represent the cell at (i, j) – i.e., the cell with row index i and column index j. If a mine is present at (i, j), then we have $X_{i,j}$, but if (i, j) is safe, then we instead have $\neg X_{i,j}$.

(a) Consider the following state of the simplified Minesweeper game, where unmarked cells denote cells that are still covered, while cells with integer value denote the uncovered cells. (Also, note the indices for each cell given on the right for your convenience.)

1		
1	1	

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)

You are given an incomplete knowledge base (KB) for the logical agent applied to the above state. Complete the following specification of the KB. Ensure that all rules are in conjunctive normal form (CNF).

Partial KB:

R1: $\neg X_{0,0}$
R2: $\neg X_{1,0}$
R3: $\neg X_{1,1}$
R4: $\neg X_{0,0} \vee \neg X_{0,1}$
R5: $\neg X_{0,0} \vee \neg X_{0,2}$
R6: $\neg X_{0,0} \vee \neg X_{1,0}$
R7: $\neg X_{0,0} \vee \neg X_{1,2}$

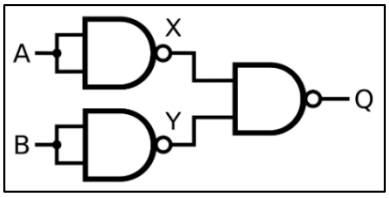
(b) Using resolution, infer that cell (0, 1) contains a mine.

(c) Assuming an agent function that utilises a KB and inference engine (IF) similar to those used in Part (a) and Part (b)**,** describe how the agent function can formulate queries to send to the IF, and explain how these queries are linked to actions that may be taken in the simplified Minesweeper game.

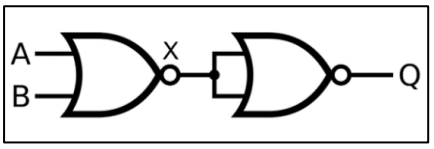
3. **[AY23/24 S2 Final]** You are a chief designer in a semiconductor company that has been making the following digital circuits for years now.

In this circuit (***Circuit* 1** – depicted on the right), three ***NAND*** gates are used. It is represented as follows:

- ***NAND***(A, A) = X
- ***NAND***(B, B) = Y
- ***NAND***(X, Y) = Q



One day, an intern that you personally hired suggests that it is possible to make the same digital circuit using fewer logic gates. The intern suggests that ***NOR*** gates be used instead of ***NAND*** gates.



In this circuit (***Circuit* 2** – depicted on the left), two ***NOR*** gates are used. It is represented as follows:

- ***NOR***(A, B) = X
- ***NOR***(X, X) = Q

As the chief designer, you would like to prove (or disprove) the intern's claim.

(a) Use **Model Checking** to prove or disprove the claim. You may label additional points in the circuits and use these points in your table.

(b) Populate a **knowledge base** (**KB**) with the rules that describe ***Circuit* 1** and ***Circuit* 2**. Make sure that each rule is in **Conjunctive Normal Form** (**CNF**). You may suffix the variables with 1 or 2 to reference ***Circuit* 1** and ***Circuit* 2**.

(c) Recall that our objective is to show that ***Circuit* 1** and ***Circuit* 2** are equivalent. State the **query** to the **KB** that would prove this. You may use the following Boolean functions: ***AND***, ***OR***, ***NOT***, ***XOR***, and ***XNOR***.

Appendix A: Notes on Knowledge Bases

A knowledge base (KB) corresponds to a set of logical rules that models what the agent knows. These rules are written using a certain language (or *syntax*) and uses a certain truth model (or *semantics*) to determine when a certain statement is *True* or *False*. In propositional logic sentences are defined as follows.

1. Atomic Boolean variables are sentences.
2. If S is a sentence, then so is $\neg S$.
3. If S_1 and S_2 are sentences, then so are:
    a. $S_1 \wedge S_2$, i.e., "S_1 and S_2".
    b. $S_1 \vee S_2$, i.e., "S_1 or S_2".
    c. $S_1 \Rightarrow S_2$, i.e., "S_1 implies S_2".
    d. $S_1 \Leftrightarrow S_2$, i.e., "S_1 holds if and only if S_2 holds".

We say that a logical statement α models β ($\alpha \models \beta$) if β holds whenever α holds. In other words, if $M(\alpha)$ is the set of all value assignments to variables in α for which α holds *True*, then $M(\alpha) \subseteq M(\beta)$.

An inference algorithm $\boldsymbol{A}$ is one that takes as input a knowledge base (**KB**) and a query α and decides whether α is derived from **KB**, written as $\mathbf{KB} \vdash_{\boldsymbol{A}} \alpha$. $\boldsymbol{A}$ is *sound* if $\mathbf{KB} \vdash_{\boldsymbol{A}} \alpha$ implies that $\mathbf{KB} \models \alpha$; $\boldsymbol{A}$ is *complete* if $\mathbf{KB} \models \alpha$ implies that $\mathbf{KB} \vdash_{\boldsymbol{A}} \alpha$.

Appendix B: Propositional Logic Laws

De Morgan's Laws	$\neg(p \vee q) \equiv \neg p \wedge \neg q$	$\neg(p \wedge q) \equiv \neg p \vee \neg q$
Idempotent laws	$p \vee p \equiv p$	$p \wedge p \equiv p$
Associative laws	$(p \vee q) \vee r \equiv p \vee (q \vee r)$	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
Commutative laws	$p \vee q \equiv q \vee p$	$p \wedge q \equiv q \wedge p$
Distributive laws	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
Identity laws	$p \vee \mathit{False} \equiv p$	$p \wedge \mathit{True} \equiv p$
Domination laws	$p \vee \mathit{True} \equiv \mathit{True}$	$p \wedge \mathit{False} \equiv \mathit{False}$
Double negation law	$\neg(\neg p) \equiv p$	
Complement laws	$p \vee \neg p \equiv \mathit{True} \vee \neg \mathit{False} \equiv \mathit{True}$	$p \wedge \neg p \equiv \mathit{False} \wedge \neg \mathit{True} \equiv \mathit{False}$
Absorption laws	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
Conditional identities	$p \Rightarrow q \equiv \neg p \vee q$	$p \Leftrightarrow q \equiv (p \Rightarrow q) \wedge (q \Rightarrow p)$