

National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 7: Logical Agents II

SOLUTIONS

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.

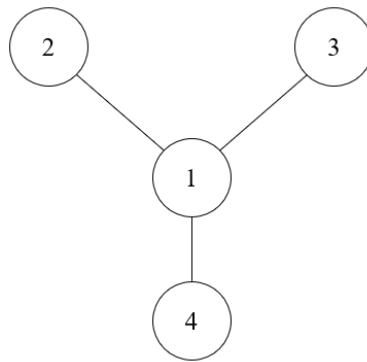


Figure 1: Graph for the Vertex Cover in Part (b)

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 $k = 1$.

Solution: We can write the edge cover constraints as follows (each relates to 1 edge).

$$x_1 \vee x_2; x_1 \vee x_3; x_1 \vee x_4$$

And the cardinality constraints as follows (each corresponds to a “at most 1 of” rule).

$$x_1 \Rightarrow \neg x_2; x_1 \Rightarrow \neg x_3; x_1 \Rightarrow \neg x_4; x_2 \Rightarrow \neg x_3; x_2 \Rightarrow \neg x_4; x_3 \Rightarrow \neg x_4$$

The cover constraints are in CNF, and the cardinality constraints translate into the following.

$$\neg x_1 \vee \neg x_2; \neg x_1 \vee \neg x_3; \neg x_1 \vee \neg x_4; \neg x_2 \vee \neg x_3; \neg x_2 \vee \neg x_4; \neg x_3 \vee \neg x_4$$

For resolution we need to show that $\text{KB} \models \alpha = x_1$. Thus, we take our $\text{KB} \wedge \neg \alpha \equiv \text{KB} \wedge \neg x_1$. Applying resolution (where \oplus is the resolution operation), we have:

$$\begin{aligned} \neg x_1 \oplus x_1 \vee x_2 &\Rightarrow x_2 \\ x_2 \oplus \neg x_2 \vee \neg x_3 &\Rightarrow \neg x_3 \\ \neg x_1 \oplus x_1 \vee x_3 &\Rightarrow x_3 \\ x_3 \oplus \neg x_3 &\Rightarrow \emptyset \end{aligned}$$

Since we have shown that $\text{KB} \wedge \neg x_1$ is unsatisfiable, it proves that $\text{KB} \models x_1$.

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_{ij} , 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_{ij} , but if (i, j) is safe, then we instead have $\neg X_{ij}$.

- (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).

Solution:

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: $\neg X_{0,0} \vee \neg X_{1,2}$

At most 1 mine in neighbouring cell of (1, 1):

R8: $\neg X_{0,1} \vee \neg X_{0,2}$

R9: $\neg X_{0,1} \vee \neg X_{1,0}$

R10: $\neg X_{0,1} \vee \neg X_{1,2}$

R11: $\neg X_{0,2} \vee \neg X_{1,0}$

R12: $\neg X_{0,2} \vee \neg X_{1,2}$

R13: $\neg X_{1,0} \vee \neg X_{1,2}$

At least 1 mine in neighbouring cell of (1, 1):

R14: $X_{0,0} \vee X_{0,1} \vee X_{0,2} \vee X_{1,0} \vee X_{1,2}$

At most 1 mine in neighbouring cell of (0, 0):

R15: $\neg X_{0,1} \vee \neg X_{1,1}$

R16: $\neg X_{1,0} \vee \neg X_{1,1}$

Note that $\neg X_{0,1} \vee \neg X_{1,0}$ is duplicated from R9 so it does not need to be considered.

At least 1 mine in neighbouring cell of (0, 0):

R17: $X_{0,1} \vee X_{1,0} \vee X_{1,1}$

At most 1 mine in neighbouring cell of (1, 0):

R18: $\neg X_{0,0} \vee \neg X_{1,1}$

Note that $\neg X_{0,0} \vee \neg X_{0,1}$ is duplicated from R4 $\neg X_{0,1} \vee \neg X_{1,1}$ is duplicated from R16 so it does not need to be considered.

At least 1 mine in neighbouring cell of (1, 0):

R19: $X_{0,0} \vee X_{0,1} \vee X_{1,1}$

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

Solution: Inference for cell (0, 1) containing a mine: $\neg\alpha \equiv \neg X_{0,1}$

Let R20: $\neg X_{0,1}$

Resolution Steps:

Resolve R19: $(X_{0,0} \vee X_{0,1} \vee X_{1,1})$ with R20: $(\neg X_{0,1}) \rightarrow$ R21: $X_{0,0} \vee X_{1,1}$

Resolve R01: $(\neg X_{0,0})$ with R21: $(X_{0,0} \vee X_{1,1})$ with \rightarrow R22: $X_{1,1}$

Resolve R03: $(\neg X_{1,1})$ with R22: $(X_{1,1}) \rightarrow$ R23: \emptyset

Therefore, since $KB \wedge \neg\alpha$ is unsatisfiable, and as such we infer that (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.

Solution: One possible way to perform inferences and, consequently, take actions would be to perform inference on cells where there is some information about their safety (i.e., cells that have neighbouring cells that are uncovered).

To do this, we will maintain 3 lists:

1. A list of cells that are uncovered (i.e., the cells that do not contain mines and have been uncovered), U .
2. A list of cells that have been identified to contain mines (i.e., based on inferences) and thus, have not been uncovered, M .
3. A list of active cells, A , where a cell, c , is considered active if (i) at least one of its neighbouring cells is uncovered, (ii) it does not contain a mine (i.e., not in M), and (iii) is itself not uncovered (i.e., not in U).

For each cell c in A , we perform two inferences:

- i. c contains a mine.
- ii. c does not contain a mine.

Consequently, if we can infer that c contains a mine, then we mark it as a mine by removing it from A and adding it to M . Otherwise, if we can infer that c does not contain a mine, then we can uncover c and add it to U . If c is uncovered, then any cell neighbouring c that has not already in A , M , or U , is then added to A .

When all cells are in M and U . Then the game has been cleared.

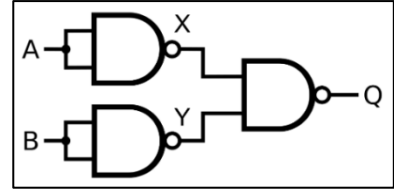
If the game has not been cleared, but A is empty or we have already unsuccessfully attempted to make inferences on all cells in A , then we must make guesses about which cells are safe.

/* When making guesses, we can use probabilities to make safer guesses. */

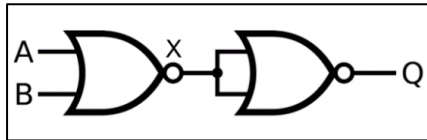
3. [AY23/24 S2 Final] You are a chief designer in a semiconductor company that has been making the following digital circuits for years now.

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.

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

Solution:

- Comparing the truth tables, we see that the models for both circuits are the same.

Circuit 1

A	B	X	Y	Q
T	T	F	F	T
T	F	F	T	T
F	T	T	F	T
F	F	T	T	F

Circuit 2

A	B	X	Q
T	T	F	T
T	F	F	T
F	T	F	T
F	F	T	F

- The KB includes the following rules.

$R_{01}: A_1 \vee X_1$
 $R_{02}: \neg A_1 \vee \neg X_1$
 $R_{03}: B_1 \vee Y_1$
 $R_{04}: \neg B_1 \vee \neg Y_1$
 $R_{05}: X_1 \vee Q_1$
 $R_{06}: Y_1 \vee Q_1$

$R_{07}: \neg X_1 \vee \neg Y_1 \vee \neg Q_1$
 $R_{08}: A_2 \vee B_2 \vee X_2$
 $R_{09}: \neg A_2 \vee \neg X_2$
 $R_{10}: \neg B_2 \vee \neg X_2$
 $R_{11}: X_2 \vee Q_2$
 $R_{12}: \neg X_2 \vee \neg Q_2$

Circuit 1

- $NAND(A, A) \rightarrow X$
- $\equiv AND(A, A) \vee X$
- $\equiv (A \vee X) \wedge (A \vee X)$
- $\equiv A \vee X$
- $X \rightarrow NAND(A, A)$
- $\equiv \neg X \vee NAND(A, A)$
- $\equiv \neg X \vee \neg(A \wedge A)$
- $\equiv \neg X \vee \neg A \vee \neg A$
- $\equiv \neg X \vee \neg A$
- $NAND(B, B) \rightarrow Y$
- $\equiv B \vee Y$
- $Y \rightarrow NAND(B, B)$
- $\equiv \neg Y \vee \neg B$
- $NAND(X, Y) \rightarrow Q$
- $\equiv (X \vee Q) \wedge (Y \vee Q)$
- $Q \rightarrow NAND(X, Y)$
- $\equiv \neg Q \vee \neg(X \wedge Y)$
- $\equiv \neg Q \vee \neg X \vee \neg Y$

Circuit 2

- $NOR(A, B) \rightarrow X$
- $\equiv A \vee B \vee X$
- $X \rightarrow NOR(A, B)$
- $\equiv \neg X \vee \neg(A \vee B)$
- $\equiv \neg X \vee (\neg A \wedge \neg B)$
- $\equiv (\neg X \vee \neg A) \wedge (\neg X \vee \neg B)$
- $NOR(X, X) \rightarrow Q$
- $\equiv X \vee Q$
- $Q \rightarrow NOR(X, X)$
- $\equiv \neg X \vee \neg Q$

(c) We wish to show that “ $(A_1 = A_2 \wedge B_1 = B_2) \Leftrightarrow Q_1 = Q_2$ ”.

Thus, the query, α , is $XNOR((A_1 = A_2 \wedge B_1 = B_2), Q_1 = Q_2)$ // i.e., since **IFF** \equiv **XNOR**.

- Given that the negation is **XOR** $((A_1 = A_2 \wedge B_1 = B_2), Q_1 = Q_2)$.

The query, α , is thus: $XOR(XNOR(A_1, A_2) \wedge XNOR(B_1, B_2), XNOR(Q_1, Q_2))$.

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 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 $\mathbf{KB} \vdash_A \alpha$. A is *sound* if $\mathbf{KB} \vdash_A \alpha$ implies that $\mathbf{KB} \models \alpha$; A is *complete* if $\mathbf{KB} \models \alpha$ implies that $\mathbf{KB} \vdash_A \alpha$.

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	$p \vee \text{False} \equiv p$	$p \wedge \text{True} \equiv p$
Domination laws	$p \vee \text{True} \equiv \text{True}$	$p \wedge \text{False} \equiv \text{False}$
Double negation law	$\neg(\neg p) \equiv p$	
Complement laws	$p \vee \neg p \equiv \text{True} \vee \neg \text{False} \equiv \text{True}$	$p \wedge \neg p \equiv \text{False} \wedge \neg \text{True} \equiv \text{False}$
Absorption laws	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
Conditional identities	$p \Rightarrow q \equiv \neg p \vee q$	$p \Leftrightarrow q \equiv (p \Rightarrow q) \wedge (q \Rightarrow p)$