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:

- **Assignment 8** consists of **Question 1** from this tutorial.
- Your solution(s) must be TYPE-WRITTEN, though diagrams may be hand-drawn.
- You are to submit your solution(s) during your **Tutorial Session in Week 11**.

Note: you may discuss the assignment question(s) with your classmates, but you must work out and write up your solution individually. Solutions that are plagiarised will be heavily penaltised.

Please 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.

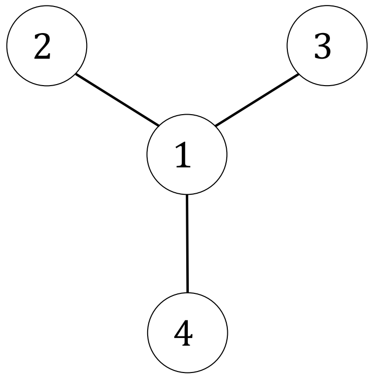

Figure 1: Graph for Vertex Cover CSP part (b)

When writing the constraints you may **only use**

- Standard logical and set operators ($\forall, \exists, \vee, \wedge$ and $x \in X, X \subseteq Y$)

(i) 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.

(ii) 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. Suppose that we are maintaining a knowledge base with propositional logical statements involving Boolean variables $x_1, \dots, x_n$. Given a logical formula q, let $M(q)$ be the set of all truth assignments to variables for which q is true. Recall that an inference algorithm $\mathcal{A}$ is sound if whenever a statement q is inferred by a knowledge base KB, it must be the case that $M(KB) \subseteq M(q)$. An inference algorithm $\mathcal{A}$ is *complete* if whenever $M(KB) \subseteq M(q)$, then eventually q will be inferred by $\mathcal{A}$. Formally prove that the resolution algorithm is sound (you've seen a sketch in class). You may also try to show completeness, but this is a longer proof (hint: you can use induction).

3. Recall that a CNF formula ϕ is defined over a set of variables $X = \{x_1, \dots, x_n\}$; a truth assignment assigns true/false (or equivalently 1 or 0) to every variable $x_i \in X$. Thus, it is useful to think of ϕ as a mapping from $\{0,1\}^n$ to $\{0,1\}$. We say that an assignment $\vec{a} \in \{0,1\}^n$ satisfies ϕ if $\phi(\vec{a}) = 1$. A k-CNF formula is one where each clause contains at most k literals. For example,

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_3 \vee x_4)$$

is a 3-CNF formula, since the size of each clause is no more than 3.

Show that every CNF formula can be converted to a 3-CNF formula. More formally, suppose that we are given a k-CNF formula ϕ with $k \geq 3$, over n variables $X = \{x_1, \dots, x_n\}$. You need to show that there exists some 3-CNF formula ϕ', whose domain is X and additional variables $Y = \{y_1, \dots, y_m\}$, such that for every truth assignment $\vec{a} \in \{0,1\}^n$, there exists a truth assignment to $y_1, \dots, y_m$, say $\vec{b} \in \{0,1\}^m$ such that $\phi(\vec{a}) = 1$ if and only if $\phi'(\vec{a}; \vec{b}) = 1$.

Hint: We have seen that the resolution algorithm can iteratively reduce the size of clauses e.g.

$$\frac{(x_1 \vee a) \wedge (x_2 \vee x_3 \vee \cdots \vee x_m \vee \neg a)}{(x_1 \vee x_2 \vee x_3 \vee \ldots x_m)}.$$

The trick in this question is to run the algorithm 'in the other direction' by adding dummy variables (whose value will be determined later as in the resolution algorithm), in order to reduce the size of the clauses.

4. Show that the resolution procedure for CNF formulas described in class yields a polynomial time algorithm for deciding whether a 2-CNF formula is satisfiable.

   Hint: Note that any clause containing exactly two variables can be written in terms of a conditional, i.e.:

$$\begin{array}{rr} x \Rightarrow y & x \Rightarrow \neg y \\ \neg x \Rightarrow y & \neg x \Rightarrow \neg y \end{array}$$

   Next, look at the directed graph whose nodes are variables (and their negations), and think what happens if there is some cycle containing a variable x and its negation $\neg x$.

Appendix A: Notes on Knowledge Bases

A knowledge base KB is a set of logical rules that model what the agent knows. These rules are written using a certain language (or *syntax*) and use a certain truth model (or *semantics* which say 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 is:
   (a) $S_1 \land S_2$ "S_1 and S_2"
   (b) $S_1 \lor S_2$ "S_1 or S_2"
   (c) $S_1 \Rightarrow S_2$ "S_1 implies S_2"
   (d) $S_1 \Leftrightarrow S_2$ "S_1 holds if and only if S_2 holds"

We say that a logical statement a models b ($a \models b$) if b holds whenever a holds. In other words, if $M(q)$ is the set of all value assignments to variables in a for which a holds true, then $M(a) \subseteq M(b)$.

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

Appendix B: Propositional Logic Laws

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