National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 7: Inference in First-Order Logic

Issued: March 23, 2020 Due: Week 10, in the tutorial class

Important Instructions:

- *This tutorial is ungraded, but please bring a copy to class for the purpose of following the class discussion.*
- *You may discuss the content of the questions with your classmates. But work out and write up ALL the solutions by yourself.*

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) $S_1 \wedge S_2$ "S_1 and S_2"
   (b) $S_1 \vee S_2$ "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 $KB \vdash_{\mathcal{A}} \alpha$.

1. Given the following logical statements, use truth-table enumeration to show that $KB \models \alpha$. In other words, write down all possible true/false assignments to the variables, the ones for which KB is true and the one for which α is true, and see whether one is a subset of the other.

(a)

$$KB = (x_1 \vee x_2) \wedge (x_1 \Rightarrow x_3) \wedge \neg x_2$$
$$\alpha = x_3 \vee x_2$$

(b)

$$KB = (x_1 \vee x_3) \wedge (x_1 \Rightarrow \neg x_2)$$
$$\alpha = \neg x_2$$

Solution: For the first KB and α pair we have:

x_1	x_2	x_3	KB	α
True	*True*	*True*	*False*	*True*
True	*True*	*False*	*False*	*True*
True	*False*	*True*	*True*	*True*
True	*False*	*False*	*False*	*False*
False	*True*	*True*	*False*	*True*
False	*True*	*False*	*False*	*True*
False	*False*	*True*	*False*	*True*
False	*False*	*False*	*False*	*False*

Note that $\alpha = x_2 \vee x_3$ is false iff both x_2 and x_3 are false (rows 4 and 8 in the table above), and that KB evaluates to false in those cases as well; moreover, when $KB = True$ (row 3), α is true as well. Thus $KB \models \alpha$.

For the next KB and α pair, we have:

x_1	x_2	x_3	KB	α
True	*True*	*True*	*False*	*False*
True	*True*	*False*	*False*	*False*
True	*False*	*True*	*True*	*True*
True	*False*	*False*	*True*	*True*
False	*True*	*True*	*True*	*False*
False	*True*	*False*	*False*	*False*
False	*False*	*True*	*True*	*True*
False	*False*	*False*	*False*	*True*

In order to show that $KB \models \alpha$ we need to show that whenever KB is true, so is α. However this is not the case: note that in line 5 $KB = True$ but $\alpha = False$.

2. Given a graph $G = \langle V, E \rangle$, we say that a subset of vertices $X \subseteq V$ is an *independent set* if no two vertices in X share an edge.

   (a) Given a set of vertices $X \subseteq V$, write the constraint "no two vertices in X share an edge". You may only use Boolean variables of the form $x_v \in \{0, 1\}$ which indicate that x_v is part of the independent set. You may not refer to the set X in your solution, only the set V, and the edges in E. You can use basic arithmetic operators and basic logical operators.

   Solution: We can write the independence constraint as: $\sum_{v \in V} \sum_{u \in V : \{u,v\} \in E} x_u \times x_v = 0$.
   For the logical form it should be: for every $\{u, v\} \in E$ we have $\neg x_u \vee \neg x_v$.

   (b) Consider the graph in Figure 1. Write down the independent set constraints explicitly for this graph. In addition write down the constraint that the independent set size must be exactly 2. Then, run the AC3 algorithm assuming that we set $x_1 = 1$, and when we set $x_1 = 0$. Ignore the size constraint for this part (you can't write it in binary constraints when the size $k \geq 2$).

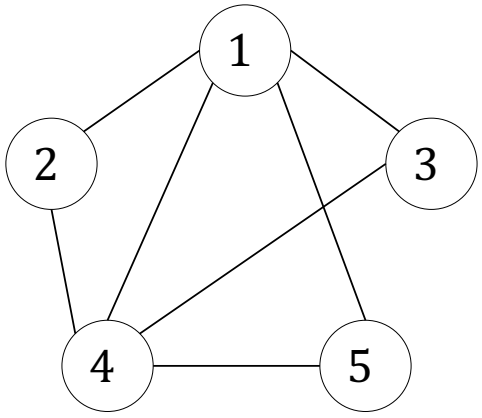

Figure 1: Independent set graph example

Solution: Independent set constraints:

$$x_1 \times x_2 = 0; x_1 \times x_3 = 0$$
$$x_1 \times x_4 = 0; x_1 \times x_5 = 0$$
$$x_2 \times x_4 = 0; x_3 \times x_4 = 0$$
$$x_4 \times x_5 = 0$$

Size constraints:

$$\forall i, j, k : x_i + x_j + x_k \leq 2$$

(c) Write down the independent set constraints in Figure 1 in the language of propositional logic. Run inferences according to

i. Resolution
ii. Forward chaining
iii. Backwards chaining

to decide the following queries

i. The vertex 1 is not in the independent set (assume independent set size is $k = 2$). More formally, you need to show that if $x_1 = \mathit{True}$ then $x_2, x_3, x_4 = \mathit{False}$.
ii. The vertex 4 is not in the independent set (assume independent set size is $k = 3$). More formally, you need to show that if $x_4 \wedge x_j$ for some j, then the rest of the variables must be false.
iii. Feel free to think of other queries!

Solution: In terms of propositional logic we get

$$\neg x_1 \vee \neg x_2; \neg x_1 \vee \neg x_3$$
$$\neg x_1 \vee \neg x_4; \neg x_1 \vee \neg x_5$$
$$\neg x_2 \vee \neg x_4; \neg x_3 \vee \neg x_4$$
$$\neg x_4 \vee \neg x_5$$

For the size constraints, for $k = 2$ it will be

$$\forall i, j, k : x_i \wedge x_j \Rightarrow \neg x_k$$

Writing them in CNF form we have

$$\forall i, j : \neg x_i \vee \neg x_j$$
$$\forall i, j, k : \neg x_i \vee \neg x_j \vee \neg x_k$$

To prove that $\neg x_1$ we go through the following stages: first assume that $x_1 = True$ and that the knowledge base holds. By resolving x_1 with the constraints of the form $\neg x_1 \vee \neg x_j$ we get that $\neg x_j$ for all $j = \{2, 3, 4\}$.
The second part is obtained in a similar manner.

3. What is the problem in each the following first order logic statements? Suggest how these statements can be corrected.

   (a) $\forall x : Boy(x) \wedge Tall(x)$
   (Intended meaning: all boys are tall)

   (b) $\exists x : Boy(x) \Rightarrow Tall(x)$
   (Intended meaning: some boy is tall)

   Solution: This statement really means: everything under the universe is a boy and is tall. It excludes the possibility that something in the universe need not be a boy or need not be tall.

   Corrected statement: $\forall x : Boy(x) \Rightarrow Tall(x)$. Note that this statement holds true even if x is not a boy and is not tall. Thus, the correct way to express such "for all" statements is usually via an implication.

   The other statement is vacuously true as long as there is something in the universe that is not a boy. This is because the implication $Boy(x) \Rightarrow Tall(x)$ is satisfied whenever x is not a boy.

   Corrected statement: $\exists x : Boy(x) \wedge Tall(x)$. Note that this statement requires that there is something that is both a boy and is tall. Thus, the correct way to express such "there exists" statements is usually via a conjunction.

4. (Slightly modified from Question 9.24 of AIMA 3rd edition) Here are two sentences in the language of first-order logic:

   (a) $\forall x : \exists y : (x \geq y)$

   (b) $\exists y : \forall x : (x \geq y)$

   (i) Assume that the variables range over all the natural numbers $0, 1, 2, \ldots$ and that the "$\geq$" predicate means "is greater than or equal to". Under this interpretation, translate (a) and (b) into English.

   Solution: (a) is translated as "For every natural number x, there is a natural number y that is less than or equal to x". (b) is translated as "There is a natural number y that is less than or equal to all natural numbers".

   (ii) Is (a) true under this interpretation? Is (b) true under this interpretation?

Solution: Yes, (a) is true under this interpretation. You can always pick the number x itself to be the number y.

Yes, (b) is true under this interpretation. You can pick 0 to be the number y.

(iii) Does (a) logically entail (b)? Does (b) logically entail (a)? Justify your answers.

Solution: (a) does not logically entail (b). Consider the domain of integers

$$\{\ldots, -2, -1, 0, 1, 2, \ldots\},$$

and the interpretation where "$\geq$" maps to "is greater than or equal to". Then, (a) is true but (b) is false. So, (a) $\Rightarrow$ (b) is false. Therefore, (a) does not entail (b).

(b) logically entails (a). We will prove by resolution.

First, convert (b) to CNF:

$$\begin{aligned} \exists y :& \forall x : (x \geq y) \\ \forall x :& (x \geq C) \qquad\qquad \text{(Skolemization)} \\ & x \geq C \end{aligned}$$

Next, convert ¬(a) to CNF:

$$\begin{aligned} & \neg(\forall x : \exists y : (x \geq y)) \\ \exists x :& \forall y : \neg(x \geq y) \\ \forall y :& \neg(D \geq y) \qquad\qquad \text{(Skolemization)} \\ & \neg(D \geq y) \end{aligned}$$

The two clauses unify with MGU $\{x \leftarrow D, y \leftarrow C\}$ to derive the empty clause.

This can be written in intuitive natural language as well: suppose by contradiction that (b) does not entail (a). For simplicity we write $\neg(A \geq B)$ as $A < B$. By (b), there is some constant C such that for every x, $x \geq C$. Now, suppose for contradiction that there exists some D such that for every y, $D < y$ (both steps thus far were Skolemization steps). This, in particular, holds for C, thus $D < C$; moreover, $D \geq C$ by (b) (this is the MGU step!). Putting them together we obtain $D \geq C > D$, a contradiction (this is the empty clause resolution step). Intuitively, any resolution can be written as a proof by contradiction.

5. Suppose that we are maintaining a knowledge base with propositional logical statements involving Boolean variables $x_1, \ldots, x_n$. 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).

Solution: Suppose that we obtained α from KB by running a sequence of resolution operations. Our proof is by induction on the number of resolutions we executed before obtaining α. Let $\vec{x} = (x_1, \ldots, x_n)$; suppose that KB is in CNF form. For the first resolution step we have that there exist two OR clauses in KB of the form $P(\vec{x}) \vee x$ and $Q(\vec{x}) \vee \neg x$. Applying resolution we get $P(\vec{x}) \vee Q(\vec{x})$. Note that if $\vec{t} \in \{True, False\}^n$ is a satisfying truth assignment for KB then it must be that both $P(\vec{x}) \vee x$ and $Q(\vec{x}) \vee \neg x$ are true. In particular, if $x = True$ under $\vec{t}$, then $Q(\vec{t}) = True$; if $x = False$ then $P(\vec{t}) = True$; in either case the expression $P(\vec{t}) \vee Q(\vec{t}) = True$. Since this is true for any resolvent reachable after 1 resolution step (there was nothing special about the one we picked), we have shown the case for resolvents that are reached after one step. For the inductive step, suppose that any resolvent q that is achievable after r resolution steps satisfies $M(KB) \subseteq M(q)$; we show the claim holds for $r + 1$. However, this is simply a repetition of the proof for the one-step case, with the KB being KB^r: the set of all resolvents reachable from KB after r resolution steps.

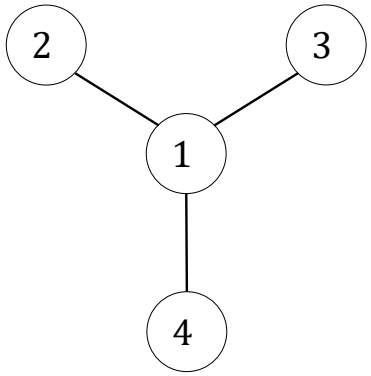

Figure 2: Graph for Vertex Cover CSP part (b)

6. Consider an instance of the VERTEXCOVER problem given in Figure 2. 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. 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 $k = 1$.

(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 2 must be of size $k = 1$.

Solution: We can write the edge cover constraints as

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

and the cardinality constraints as

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

Which translate to CNF as

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

For resolution we need to show that $KB \models \alpha = x_1$. Thus we take our $KB \wedge \neg\alpha \equiv KB \wedge \neg x_1$. Applying resolution we have

$$\neg x_1 \oplus x_1 \vee x_2 \Rightarrow x_2$$
$$x_2 \oplus \neg x_2 \vee \neg x_3 \Rightarrow \neg x_3$$
$$\neg x_1 \oplus x_1 \vee x_3 \Rightarrow x_3$$
$$x_3 \oplus \neg x_3 \Rightarrow []$$

Here, $\oplus$ is the resolution operation.

7. Consider the no regret learning setting we have studied in class. We saw that in order to achieve a low regret, we need a sufficiently large number of rounds. What happens when the number of rounds is small? Prove that if there are n experts and T time steps, such that $T < \lfloor \log_2 n \rfloor$, then for any randomized algorithm $\mathcal{A}$, there is some randomized assignment of losses[1] in $\{0, 1\}$ to the experts (which can depend on what $\mathcal{A}$ does) such that

(a) $\mathcal{A}$ incurs a loss of at least $\frac{T}{2}$.

(b) There exists at least one action that has a loss of 0.

[1] Here we consider losses instead of rewards, but the idea is similar.

Solution: We analyze the case where $n = 2^k$ with $k > T$ for ease of exposition. The idea is as follows: we maintain a list of 'flagged' actions. When an action is flagged at time t, it loses from time t onwards. At every time step t, choose a random set of $\frac{n}{2^t}$ unflagged actions, and flag them; the expected loss of any algorithm on this randomly generated sequence of actions is $\geq \frac{1}{2}$ at each time step. In more detail, at time $t = 1$ we choose a set of $\frac{n}{2}$ actions uniformly at random and set their loss to 1; all other actions incur a loss of 0. Since every action has an expected loss of $\frac{1}{2}$, the expected loss of the algorithm $\mathcal{A}$ is $\frac{1}{2}$; in particular, there exists some set A_1^* for which setting the losses of A_1^* to 1 and the rest of the losses to 0 will result in $\mathcal{A}$ losing at least $\frac{1}{2}$ in round 1. We assume inductively that at times $1, \ldots, t-1$ we could choose sets $A_1^* \subseteq A_2^* \subseteq \cdots \subseteq A_{t-1}^*$ such that for all $q \in \{1, \ldots, t-1\}$:

(a) $|A_q^*| = \left(1 - \frac{1}{2^q}\right) n$.

(b) An action in A_q^* incurs a loss of 1 at times $q, \ldots, t-1$

(c) All actions in $N \setminus A_q^*$ incur a loss of 0

(d) The expected loss of $\mathcal{A}$ on A_q^* at time q is $\geq \frac{1}{2}$ (note that this randomization is over the randomness of $\mathcal{A}$, the sets A_q^* are chosen deterministically).

We choose a set of $\frac{n}{2^t}$ actions in $N \setminus A_{t-1}^*$ uniformly at random. At round t, every action in the set $N \setminus A_{t-1}^*$ has an expected loss of $\frac{1}{2}$, and every action in A_{t-1}^* has a (deterministic) loss of 1 at time t (we made it so that once an action is flagged, it always loses). Therefore, the expected loss of $\mathcal{A}$ from our randomized choice of $\frac{n}{2^t}$ losing actions is at least $\frac{1}{2}$; therefore, there exists some choice of a set of $\frac{n}{2^t}$ actions in $N \setminus A_{t-1}^*$, call it B_t^* such that setting the actions in $B_t^* \cup A_{t-1}^*$ to be losing at round t results in $\mathcal{A}$ having an expected loss of at least $\frac{1}{2}$. Therefore setting $A_t^* = A_{t-1}^* \cup B_t^*$ yields the desired result: A_t^* satisfies all of the properties listed above (check!).

Since we terminate at time $T < \lfloor \log n \rfloor$, $|A_T^*| = \left(1 - \frac{1}{2^T}\right) n < n$. In particular, there is at least one action a^* that was never flagged, and therefore never exhibited any loss. In this case, the expected loss of $\mathcal{A}$ would be at least $\frac{T}{2}$, but there is some action a^* that has a total loss of 0.

8. Recall that the RANDGREEDY algoritm picks at time t an action uniformly at random from S_t, the set of best-performing actions at time t. Formally prove that the randomized greedy algorithm discussed in class achieves a $\mathcal{O}(\log n)$ regret when compared to the best action. In other words, that for every loss of the best action, RANDGREEDY loses at most $\mathcal{O}(\log n)$ times. Recall the setup: we look at the best action, call it b, the specific time-steps that it incurred a loss. Suppose that after T time steps, b loses at steps $t_1, \ldots, t_k$ for a total loss of k. Formally prove that the expected loss of RANDGREEDY in between consecutive losses is $\mathcal{O}(\log n)$. We have gone over most of the steps in class, but not written down a formal proof. The general idea is this: assume that the for every $t \in t_j + 1, \ldots, t_{j+1}$, S_t is the set of best actions at time t, and that $r(t)$ is the number of actions in S_t that incurred a loss at time t.

What happens to the size of S_t? What is the expected loss of RANDGREEDY? How can we bound the loss of RANDGREEDY between t_j and t_{j+1}?