

COMP3620/6320: Artificial Intelligence

Tutorial 2: KRR

Q1. CNF conversion, Horn KBs, resolution, and DPLL.

You are given the following propositional knowledge base (KB):

- $a \Rightarrow (b \Rightarrow c)$
- $a \Leftrightarrow b$
- $(a \wedge \neg c) \vee (b \wedge \neg c)$

Answer the following questions:

- Are any of these formulae already in CNF? For those that are not, convert them to CNF.
- Is the CNF KB in Horn form? If not, state which clauses are not in Horn form. Can you perform forward-chaining or backward-chaining as a means of inference in this KB?
- Is there a refutation of this clause set via resolution? If so, show it.
- Show how DPLL with unit propagation (but no pure literal elimination) using variable ordering a, b, c works on this clause set.

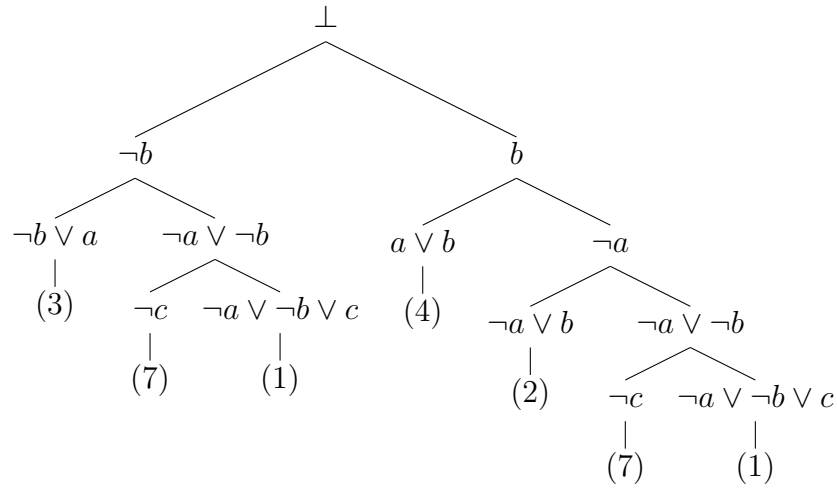
Solution:

(a) None of the formulae are in CNF, the CNF form of the KB is below:

- (1) $\neg a \vee \neg b \vee c$
- (2) $\neg a \vee b$
- (3) $\neg b \vee a$
- (4) $a \vee b$
- (5) $a \vee \neg c$
- (6) $b \vee \neg c$
- (7) $\neg c$

(b) Clause (4) is not in Horn form (Horn clauses have at most one positive literal), thus forward-chaining and back-chaining are not complete forms of inference for this KB.

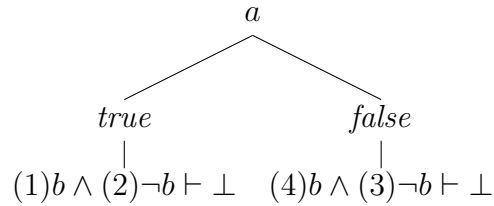
(c) Following is a resolution tree with clauses in the KB at the leaves. Each parent node is a resolvent of its children. The final resolvent of \perp shows that the KB is unsatisfiable.



(d) Before running DPLL, we first run unit propagation. Noting that $\neg c$ implies $c = \text{false}$ is any satisfiable model, we make this assignment and obtain the updated set of clauses:

- (1) $\neg a \vee \neg b$
- (2) $\neg a \vee b$
- (3) $\neg b \vee a$
- (4) $a \vee b$

With $c = \text{false}$, clause (1) is simplified and clauses (5), (6), and (7) are satisfied and thus removed from consideration. Now we begin the recursive DPLL model-building procedure:



At each node is the conjunction of simplified CNF formulae with their original clause number in parenthesis (satisfied clauses have been removed altogether). Note that on the first recursion, unit propagation is able to derive inconsistency (i.e., $\vdash \perp$) for both branches, thus showing the set of clauses to be unsatisfiable.

Q2. 2-SAT is Poly-time.

3-SAT is known to be NP-Complete (the input consists of a KB in CNF form where each clause has exactly 3 literals).

Show that there is an algorithm for solving 2-SAT that is polynomial time in the input size (each clause has a maximum of two literals).

Solution:

A polynomial time algorithm for 2-SAT proceeds as follows:

- (a) Perform unit propagation to get rid of unit clauses. Return UNSAT if a clause is unsatisfied. This is linear in the number of clauses.
- (b) Build a graph where any two variables participating in a clause have an edge between them. Partition the graph into its strongly connected components. Both of these operations are linear time.
- (c) For each strongly connected component, choose an arbitrary variable and assign it *true*. Note that this will lead to unit propagation that terminates with the clauses in the component being labeled satisfiable or unsatisfiable. Repeat by assigning the initial variable *false* and performing unit propagation. Do this for all components. If both assignments lead to unsatisfiability in any component, return UNSAT else return SAT. All of this takes linear time in the number of clauses.

Thus, the entire problem is linear time in the number of clauses and thus clearly polynomial time in the input size.

Q3. Translation to FOL and first-order resolution.

Translate the following sentences to first order logic (using only the predicate or function names provided in parenthesis) and use resolution to determine whether the query is entailed:

Knowledge base (KB):

- (a) A vegetarian pizza is defined as a pizza whose all toppings are vegetables ($VegetarianPizza(\cdot)$, $Pizza(\cdot)$, $Topping(\cdot, \cdot)$, $Vegetable(\cdot)$).
- (b) A non-vegetarian pizza is defined as a pizza with at least one non-vegetable topping ($NonVegetarianPizza(\cdot)$, $Pizza(\cdot)$, $Topping(\cdot, \cdot)$, $Vegetable(\cdot)$).
- (c) There exists a pizza with a Pepperoni topping and Pepperoni is known to not be a vegetable ($Pizza(\cdot)$, $Topping(\cdot, \cdot)$, $Vegetable(\cdot)$).
- (d) There exists a pizza with an Asparagus topping and Asparagus is known to be a vegetable ($Pizza(\cdot)$, $Topping(\cdot, \cdot)$, $Vegetable(\cdot)$).

Queries:

Query 1: Does there exist a vegetarian pizza?

Query 2: Does there exist a non-vegetarian pizza?

Note: The answer may not be what you initially think it should be... go through the mechanical process of resolution theorem proving to see what can be actually proved. Recall that first-order logic knowledge bases only provide constraints on the world and that these constraints rarely restrict the legal interpretations to a single model.

If humans tend to make additional inferences that cannot be proved via formal theorem proving, one explanation would be that we have additional implicit background assumptions that we use in our reasoning process. If you made different inferences than the theorem prover, how might you formalize the additional background assumptions that led to *your* inferences?

Solution:

First we translate natural language to first-order logic:

- (a) $\forall x \text{VegetarianPizza}(x) \Leftrightarrow \text{Pizza}(x) \wedge (\forall y \text{Topping}(x, y) \Rightarrow \text{Vegetable}(y))$
- (b) $\forall x \text{NonVegetarianPizza}(x) \Leftrightarrow \text{Pizza}(x) \wedge (\exists y \text{Topping}(x, y) \wedge \neg \text{Vegetable}(y))$
- (c) $\exists x \text{Pizza}(x) \wedge \text{Topping}(x, \text{Pepperoni}) \wedge \neg \text{Vegetable}(\text{Pepperoni})$
- (d) $\exists x \text{Pizza}(x) \wedge \text{Topping}(x, \text{Asparagus}) \wedge \text{Vegetable}(\text{Asparagus})$

To convert to CNF for first-order resolution, first we convert \Leftrightarrow to \Rightarrow and \Rightarrow to \vee . Then we put the formula in negation normal form (NNF), by “pushing-down” all negation to the literal level. Finally we pull all quantifiers out to the front of each formula. This yields:

- (a \Leftarrow) $\forall x, \exists y \text{VegetarianPizza}(x) \vee \neg \text{Pizza}(x) \vee [\text{Topping}(x, y) \wedge \neg \text{Vegetable}(y)]$
- (a \Rightarrow) $\forall x, \forall y \neg \text{VegetarianPizza}(x) \vee [\text{Pizza}(x) \wedge (\neg \text{Topping}(x, y) \vee \text{Vegetable}(y))]$
- (b \Leftarrow) $\forall x, \forall y \text{NonVegetarianPizza}(x) \vee \neg \text{Pizza}(x) \vee \neg \text{Topping}(x, y) \vee \text{Vegetable}(y)$
- (b \Rightarrow) $\forall x, \exists y \neg \text{NonVegetarianPizza}(x) \vee [\text{Pizza}(x) \wedge \text{Topping}(x, y) \wedge \neg \text{Vegetable}(y)]$
- (c) $\exists x \text{Pizza}(x) \wedge \text{Topping}(x, \text{Pepperoni}) \wedge \neg \text{Vegetable}(\text{Pepperoni})$
- (d) $\exists x \text{Pizza}(x) \wedge \text{Topping}(x, \text{Asparagus}) \wedge \text{Vegetable}(\text{Asparagus})$

Next we Skolemize, drop univocal quantifiers (all free variables are assumed universally quantified) and then complete the distribution of \vee over \wedge to obtain a set of first-order CNF clauses:

- (a1 \Leftarrow) $\text{VegetarianPizza}(x) \vee \neg \text{Pizza}(x) \vee \text{Topping}(x, f_1(x))$
- (a2 \Leftarrow) $\text{VegetarianPizza}(x) \vee \neg \text{Pizza}(x) \vee \neg \text{Vegetable}(f_1(x))$
- (a1 \Rightarrow) $\neg \text{VegetarianPizza}(x) \vee \text{Pizza}(x)$
- (a2 \Rightarrow) $\neg \text{VegetarianPizza}(x) \vee \neg \text{Topping}(x, y) \vee \text{Vegetable}(y)$
- (b \Leftarrow) $\text{NonVegetarianPizza}(x) \vee \neg \text{Pizza}(x) \vee \neg \text{Topping}(x, y) \vee \text{Vegetable}(y)$
- (b1 \Rightarrow) $\neg \text{NonVegetarianPizza}(x) \vee \text{Pizza}(x)$
- (b2 \Rightarrow) $\neg \text{NonVegetarianPizza}(x) \vee \text{Topping}(x, f_2(x))$
- (b3 \Rightarrow) $\neg \text{NonVegetarianPizza}(x) \vee \neg \text{Vegetable}(f_2(x))$
- (c1) $\text{Pizza}(c_1)$
- (c2) $\text{Topping}(c_1, \text{Pepperoni})$

(c3) $\neg \text{Vegetable}(\text{Pepperoni})$

(d1) $\text{Pizza}(c_2)$

(d2) $\text{Topping}(c_2, \text{Asparagus})$

(d3) $\text{Vegetable}(\text{Asparagus})$

Now we need to actually make queries. Our queries can be written in FOL as the following:

Query 1: $\exists x \text{VegetarianPizza}(x)?$

Query 2: $\exists x \text{NonVegetarianPizza}(x)?$

However, to query the KB, we need to determine if $KB \wedge \neg \text{Query} \vdash \perp$. So we negate the above queries and convert them to CNF:

\neg *Query 1 CNF:* $\neg \text{VegetarianPizza}(x)$

\neg *Query 2 CNF:* $\neg \text{NonVegetarianPizza}(x)$

We start with \neg Query 2 and show a proof of first-order refutation via resolution. Note that all clauses that are unified have had their variables standardized apart.

Clause	Reason
[1] $\neg \text{NonVegetarianPizza}(z)$	Given (\neg Query 2 CNF)
[2] $\text{NonVegetarianPizza}(x) \vee \neg \text{Pizza}(x) \vee \neg \text{Topping}(x, y) \vee \text{Vegetable}(y)$	Given ($b \Leftarrow$)
[3] $\neg \text{Pizza}(x) \vee \neg \text{Topping}(x, y) \vee \text{Vegetable}(y)$	Resolved [1,2] $\theta = \{z/x\}$
[4] $\text{Pizza}(c_1)$	Given (c1)
[5] $\text{Topping}(c_1, \text{Pepperoni})$	Given (c2)
[6] $\neg \text{Vegetable}(\text{Pepperoni})$	Given (c3)
[7] $\neg \text{Topping}(c_1, y) \vee \text{Vegetable}(y)$	Resolved [3,4] $\theta = \{x/c_1\}$
[8] $\text{Vegetable}(\text{Pepperoni})$	Resolved [5,7] $\theta = \{y/\text{Pepperoni}\}$
[9] \perp	Resolved [6,8] $\theta = \emptyset$

From this refutation of \neg Query 2 CNF form, we can infer that Query 2 is entailed by our KB so we know that there is a non-vegetarian pizza.

Query 1 is a different story. Try as you might, but you'll never be able to derive a contradiction for \neg Query 1 CNF. Why is this? Note that while we've said there is Asparagus on a pizza, there is *no* additional constraint saying nothing else could be

on that pizza. We have constrained the type of pizza it may be, but we have not yet fully described everything that is true and false about it. Without such knowledge, a non-vegetarian pizza with both Asparagus and Pepperoni is completely consistent with our knowledge base. Thus we cannot prove the existence of a vegetarian pizza with *only* vegetable toppings from our current knowledge.

So why would a human tend to infer that there exists a vegetarian pizza given the KB content? One explanation is that humans often assume the minimal model (i.e., the minimal set of true atoms in an interpretation). So, we might *assume* that since we are only told that one pizza has Asparagus then there are no other toppings on this pizza.

Prolog can make such implicit assumptions using its so-called “closed-world assumption (CWA)” (what cannot be proven to be true from the given axioms is assumed to be false). However, the CWA can lead to semantic difficulties if restrictions are not placed on the use of negation. Alternately, the general field of *non-monotonic* logics look at additional “natural” assumptions one can make during inference. Perhaps one of the most well-known non-monotonic inference procedures aside from Prolog is a method called *circumscription* that tries to find the minimal model consistent with a set of axioms (i.e., it tries to circumscribe the models).

Note that both the CWA and circumscription are non-monotonic, because previous inferences may become false as a result of new information. On the other hand *FOL with the resolution rule for inference is monotonic*. This is easy to see from a model-theoretic standpoint because if $KB \models \alpha$ then certainly still $KB \wedge \beta \models \alpha$ (i.e., this follows from the properties of subsets and intersection and the two facts: (1) if $KB \models \alpha$ then $models(KB) \subseteq models(\alpha)$, and (2) the model-theoretic semantics of conjunction is defined as $models(KB \wedge \beta) = models(KB) \cap models(\beta)$). Since the resolution rule is *sound* and *complete* for inference, we know that the monotonicity for the model-theoretic semantics also extends to FOL inference via resolution.

You should be able to formalize such a proof of the monotonicity of resolution inference in FOL.