CSCI 561 - Foundation for Artificial Intelligence

Week 8 Discussion Section

PROF WEI-MIN SHEN SHEN@ISI.EDU

Conjunctive Normal Form

A sentence in **Conjunctive Normal Form** is a conjunction of clauses, where each clause is a disjunction of literals.

Every sentence of first-order logic can be converted into an inferentially equivalent CNF sentence.

The CNF sentence will be unsatisfiable just when the original sentence is unsatisfiable.

Example: ***"Everyone who loves all animals is loved by someone."***

Original: $\forall x\ .\ (\forall y\ .\ \mathit{Animal}(y) \Rightarrow \mathit{Loves}(x, y)) \Rightarrow (\exists y\ .\ \mathit{Loves}(y, x))$

CNF:

$(\mathit{Animal}(F(x)) \vee \mathit{Loves}(G(x), x)) \wedge (\neg \mathit{Loves}(x, F(x)) \vee \mathit{Loves}(G(x), x))$

CNF Conversion

1. **Eliminate implications**
2. **Move ¬ inward**
3. **Standardize variables**
4. Skolemization **(removing existential quantifiers)**
5. **Drop universal quantifiers**
6. **Distribute ∨ over ∧**

1. Eliminate implications

Turn biconditionals into a conjunction of implications using the Biconditional Elimination rule.

Turn implications into disjunctions using the Implication Elimination rule.

0 $\forall x \,.\, (\forall y \,.\, Animal(y) \Rightarrow Loves(x, y)) \Rightarrow (\exists y \,.\, Loves(y, x))$

1 $\forall x \,.\, (\forall y \,.\, \neg Animal(y) \vee Loves(x, y)) \Rightarrow (\exists y \,.\, Loves(y, x))$

2 $\forall x \,.\, \neg(\forall y \,.\, \neg Animal(y) \vee Loves(x, y)) \vee (\exists y \,.\, Loves(y, x))$

2. Move ¬ inward

Apply the De Morgan laws to move negations inward down to the literals. Apply Double Negation Elimination as needed.

Apply these two tautologies for negated quantifiers:

$$\neg\forall x \,.\, p \iff \exists x \,.\, \neg p$$

$$\neg\exists x \,.\, p \iff \forall x \,.\, \neg p$$

2 $\forall x \,.\, \neg(\forall y \,.\, \neg Animal(y) \vee Loves(x, y)) \vee (\exists y \,.\, Loves(y, x))$

3 $\forall x \,.\, (\exists y \,.\, \neg(\neg Animal(y) \vee Loves(x, y))) \vee (\exists y \,.\, Loves(y, x))$

4 $\forall x \,.\, (\exists y \,.\, \neg\neg Animal(y) \wedge \neg Loves(x, y)) \vee (\exists y \,.\, Loves(y, x))$

5 $\forall x \,.\, (\exists y \,.\, Animal(y) \wedge \neg Loves(x, y)) \vee (\exists y \,.\, Loves(y, x))$

3. Standardize variables

Where the same variable name is used across different quantifiers, change the name of one of them to something unique.

For example, ***y*** is an animal in the first part, and a lover in the second. The scopes do not overlap. Just change the second ***y*** to something else.

5 $\forall x \,.\, (\exists y \,.\, Animal(y) \wedge \neg Loves(x, y)) \vee (\exists y \,.\, Loves(y, x))$

6 $\forall x \,.\, (\exists y \,.\, Animal(y) \wedge \neg Loves(x, y)) \vee (\exists z \,.\, Loves(z, x))$

4. Skolemization

Skolemization is the process of removing existential quantifiers by elimination.

Remember, we used **Skolem Constants** to remove existential quantifiers when the sentence was in a specific form:

$\exists x \,.\, P(x) \Leftrightarrow P(A)$ where A is a new constant symbol.

However, we **cannot** apply this rule to our current sentence:

6 $\forall x \,.\, (\exists y \,.\, Animal(y) \wedge \neg Loves(x, y)) \vee (\exists z \,.\, Loves(z, x))$

~~7~~ $\forall x \,.\, (Animal(A) \wedge \neg Loves(x, A)) \vee Loves(B, x)$

Wrong wrong wrong!

This reads: All objects do not love a particular animal (A), or they are loved by a particular object B. That is not what we want.

The problem is that y and z are **inside the scope** of x. We want the new Skolem Constants to depend on the value of x.

4. Skolemization

Now, we can replace all existentially quantified variables with **Skolem Constants** or **Skolem Functions**, as needed.

6 $\forall x \,.\, (\exists y \,.\, Animal(y) \wedge \neg Loves(x, y)) \vee (\exists z \,.\, Loves(z, x))$

7 $\forall x \,.\, (Animal(F(x)) \wedge \neg Loves(x, F(x))) \vee Loves(G(x), x)$

* The AIMA 3rd Ed. incorrectly skolemizes the first argument of ***Loves*** as ***G(z)***, when it should be ***G(x)***, on page 346-347, but is correct on page 349.

5. Drop universal quantifiers

At this point, all remaining variables are universally quantified.

Furthermore, the current sentence is equivalent to one where all of the quantifiers are moved to the left (**Prenex Normal Form**).

Accordingly, we can drop all of the universal quantifiers, as long as we remember that these variables are all still universally quantified.

7 $\forall x . (Animal(F(x)) \land \neg Loves(x, F(x))) \lor Loves(G(x), x)$

8 $(Animal(F(x)) \land \neg Loves(x, F(x))) \lor Loves(G(x), x)$

6. Distribute ∨ over ∧

Use **Distributivity of ∨ over ∧** in order to push ∨ downward and ∧ upward, just as we did in propositional logic.

This step may also require "flattening out" nested conjunctions and disjunctions, removing unnecessary parentheses using **Associativity** tautologies.

8 ***(Animal(F(x)) ∧ ¬Loves(x, F(x))) ∨ Loves(G(x), x)***

9 ***(Animal(F(x)) ∨ Loves(G(x), x)) ∧ (¬Loves(x, F(x)) ∨ Loves(G(x), x)***

Done! The resulting sentence is in first-order Conjunctive Normal Form. It is unsatisfiable exactly when the original sentence is unsatisfiable.

Example 9.4

For each pair of atomic sentences, give the most general unifier if it exists:

a) P(A,B,B), P(x,y,z).

{x/A, y/B, z/B} (or some permutation of this).

b) Q(y,G(A,B)),Q(G(x,x),y).

No unifier (x cannot bind to both A and B).

c) Older(Father(y),y),Older(Father(x),John).

{y/John,x/John}.

d) Knows(Father(y),y), Knows(x,x).

No unifier (prevents unification of y with Father(y)).

Example 9.13

In this exercise, use the sentences you wrote in Exercise 9.6 to answer a question by using a backward-chaining algorithm.

$\text{Horse}(x) \Rightarrow \text{Mammal}(x)$

$\text{Cow}(x) \Rightarrow \text{Mammal}(x)$

$\text{Pig}(x) \Rightarrow \text{Mammal}(x)$

$\text{Offspring}(x,y) \wedge \text{Horse}(y) \Rightarrow \text{Horse}(x)$

$\text{Horse}(\text{Bluebeard})$

$\text{Parent}(\text{Bluebeard}, \text{Charlie})$

$\text{Offspring}(x,y) \Rightarrow \text{Parent}(y,x)$

$\text{Parent}(x,y) \Rightarrow \text{Offspring}(y,x)$

$\text{Mammal}(x) \Rightarrow \text{Parent}(G(x), x)$

Example 9.13

Draw the proof tree generated by an exhaustive backward-chaining algorithm for the query ∃*h* Horse(*h*), where clauses are matched in the order given.

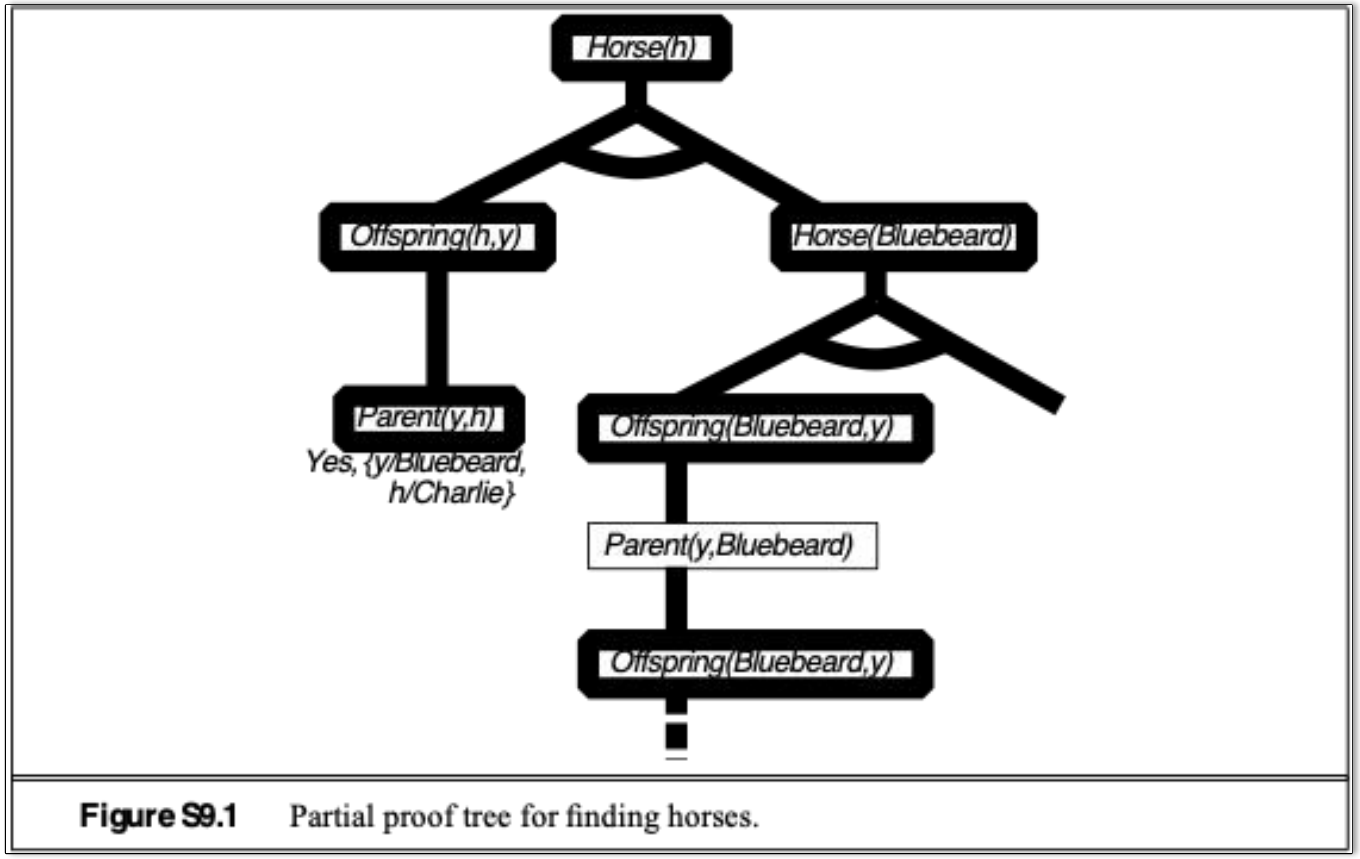

Figure S9.1 Partial proof tree for finding horses.

Club Example (Problem Statement)

Tony, Claude and Ellen belong to the Zymba Club. Every member of the Zymba Club is either a skier or a mountain climber or both. No mountain climber likes rain, and all skiers like snow. Ellen dislikes whatever Tony likes and likes whatever Tony dislikes. Tony likes rain and snow.

We want to answer the following queries:

· Is there a member of the Zymba Club who is a mountain climber but not a skier?

· Who is it?

a. Translate the problem into FOL Sentences

(∀x) S(x) v M(x)

~(Ex) M(x) ^ L(x, Rain)

(∀x) S(x) => L(x, Snow)

(∀y) L(Ellen, y) <=> ~L(Tony, y)

L(Tony, Rain)

L(Tony, Snow)

Query: (Ex) M(x) ^ ~S(x)

Negation of the Query: ~(Ex) M(x) ^ ~S(x)

b. Convert to Conjunctive Normal Form

1. S(x1) v M(x1)
2. ~M(x2) v ~L(x2, Rain)
3. ~S(x3) v L(x3, Snow)
4. ~L(Tony, x4) v ~L(Ellen, x4)
5. L(Tony, x5) v L(Ellen, x5)
6. L(Tony, Rain)
7. L(Tony, Snow)
8. Negation of the Query: ~M(x7) v S(x7)

c. Produce an answer to the first query using Proof by Contradiction

1. S(x1) v M(x1)
2. ~M(x2) v ~L(x2, Rain)
3. ~S(x3) v L(x3, Snow)
4. ~L(Tony, x4) v ~L(Ellen, x4)
5. L(Tony, x5) v L(Ellen, x5)
6. L(Tony, Rain)
7. L(Tony, Snow)
8. Negation of the Query: ~M(x7) v S(x7)

Clause 1	Clause 2	Resolvent	MGU
8	1	9. S(x1)	{x7/x1}
9	3	10. L(x1, Snow)	{x3/x1}
10	4	11.~L(Tony, Snow)	{x4/Snow,x1/Ellen}
11	7	False	{}

answer: Ellen

What you should know

- How does first-order logic differ from propositional logic? How are objects, relations, functions, variables, quantifiers and equality used in first-order logic?
- Know how to translate from English to logic
- Know how to translate from logic into English
- What is Unification? How is it different than pattern matching? When is to used? What is the algorithm? Why is it important for Generalized Modus Ponens?
- What is Forward Chaining? What is Backward Chaining? What is the differences between them?
- Know how to translate to CNF and Horn clauses

Want More?

- Check out some of these exercises in the book:

8.1-3, 8.6, 8.9-10, 8.14, 8.17, 8.28

9.3, 10, 15, 20, 23