COSC1125/1127 Artificial Intelligence

School of Science (Computer Science)
RMIT University
Semester 1, 2018
A/Prof. Sebastian Sardina

Tutorial Sheet 6
First Order Logic

For some of the questions below, you may want to check the great slides by Torsten Hahmann (CSC 384 at University of Toronto) on Skolemization, Most General Unifiers, First-Order Resolution. It includes the Skolemization process to remove existential quantification, the algorithm for finding MGUs, and FOL resolution; all with great fully detailed examples!

1. Choose a vocabulary of predicates, constants and functions appropriate for representing the following information in First Order Logic, then represent each sentence in FOL.

   (a) "There is someone who is loved by everybody"

   (b) "All cats are lazy"

   (c) "Some students are clever"

   (d) "No student is rich"

   (e) "Every man loves a woman"
   (represent both meanings)

   (f) "Everything is bitter or sweet"

   (g) "Everything is bitter or everything is sweet"

   (h) "Martin has a new bicycle"

   (i) "Lynn gets a present from John, but she doesn't get anything from Peter"

Answer

(a) $(\exists y)(\forall x) loves(x, y)$

(b) $(\forall x)(cat(x) \Rightarrow lazy(x))$

(c) $(\exists x)(student(x) \wedge clever(x))$

(d) $(\neg\exists x)(student(x) \wedge rich(x))$
or
$(\forall x)(student(x) \Rightarrow \neg rich(x))$
or
$(\forall x)\neg(student(x) \wedge rich(x))$

(e) $(\forall x)(man(x) \Rightarrow (\exists y) woman(y) \wedge loves(x, y))$ (each man will have its lover)
or
$(\exists y)(woman(y) \wedge (\forall x)(man(x) \Rightarrow loves(x, y))$ (meaning that there is a woman that every man loves)

(f) $(\forall x)(bitter(x) \vee sweet(x))$

(g) $(\forall x) bitter(x) \vee \forall y sweet(y)$

(h) $(\exists x)(bike(x) \wedge new(x) \wedge has(Martin, x))$

(i) $(\exists x)(present(x) \wedge gets(Lynn, x, John)) \wedge$
$((\neg\exists y)(present(y) \wedge gets(Lynn, y, Peter)) \wedge (x \neq y)$

2. Do the same as for the previous question.

   (a) Anyone who is rich is powerful.

   (b) Anyone who is powerful is corrupt.

   (c) Anyone who is meek and has a corrupt boss is unhappy.

   (d) Not all students take both Computing-Theory and OO-Programming.

   (e) Only one student failed Computing-Theory.

   (f) Only one student failed both Computing-Theory and OO-Programming.

   (g) The best score in Computing-Theory was better than the best score in OO-Programming.

   (h) Every person who dislikes all donkeys is smart.

   (i) There is a woman who likes all men who are not footballers.

   (j) There is a barber who shaves all men who do not shave themselves.

   (k) No person likes a lecturer unless the lecturer is smart.

   (l) Politicians can fool some of the people all of the time, and they can fool all of the people some of the time, but they can't fool all of the people all of the time.

Answer

(a) $(\forall x)(rich(x) \Rightarrow powerful(x))$

(b) $(\forall x)(powerful(x) \Rightarrow corrupt(x))$

(c) $(\forall x)(\exists y)(meek(x) \wedge isBoss(y, x) \wedge corrupt(y) \Rightarrow unhappy(x))$

(d) $(\neg\forall x)(student(x) \Rightarrow (takes(x, CT) \wedge takes(x, OO)))$

(e) $(\exists x)student(x) \wedge fails(x, CT) \wedge (\forall y)student(y) \wedge fails(y, CT) \Rightarrow x = y$

In this example, an **equality symbol** is used to make a statment about the effect that two terms refer to the same object. For the above first sentence,

$$(\exists x)student(x) \wedge fails(x, CT) \wedge (\forall y)student(y) \wedge fails(y, CT)$$

would only assert that all instances of y and there is at least one x that failed CT, but nothing says that x and y are referring to the same object.
you can imagine that you run a loop going over all possible y values, and there is an x value where $x = y$ is true. When this is the case, the implication stands.
Note that in a knowledge base, the existential quantifier is eliminated by replacing it with specific instances. Based on this assumption, the above sentence will refer to only one specific student who failed CT.

(f) $(\exists x)(student(x) \wedge fails(x, CT) \wedge fails(x, OO)) \wedge$
$(\forall y)(student(y) \wedge fails(y, CT) \wedge fails(y, OO)) \Rightarrow x = y$

(g) $(\exists x)(\forall y)score(x, CT) \wedge score(y, OO) \Rightarrow better(x, y)$

(h) $(\forall x)person(x) \wedge ((\forall y)donkey(y) \Rightarrow dislikes(x, y)) \Rightarrow smart(x)$

(i) $(\exists x)woman(x) \wedge ((\forall y)man(y) \wedge \neg footballer(y)) \Rightarrow likes(x, y)$

(j) $(\exists x)barber(x) \wedge ((\forall y)man(y) \wedge \neg shaves(y, y)) \Rightarrow shaves(x, y)$

(k) $(\forall x)(\forall y)person(x) \wedge lecturer(y) \wedge \neg smart(y) \Rightarrow \neg likes(x, y)$

(l) $(\forall x)politician(x) \Rightarrow$
$((\exists y)(\forall t)person(y) \wedge fools(x, y, t)) \wedge$
$((\exists t)(\forall y)person(y) \wedge fools(x, y, y)) \wedge$
$\neg((\forall t)(\forall y)person(y) \Rightarrow fools(x, y, t))$

3. Consider the following story:

I married a widow (let's call her w) who has a grown up daughter (d). My father (f), who visited us quite often, fell in love with my step-daugher and married her. Hence my father became my son-in-law and my step-daughter became my mother. Some months later, my wife gave birth to a son (s_1), who became the brother-in-law of my father, as well as my uncle. The wife of my father, that is, my step-daughter, also had a son (s_2).

Using predicate calculus, create a set of expressions that represent the situation in the above story. Extend the kinship relations to in-laws and uncle. Use generalised modus ponens to prove 'I am my own grandfather'.

Answer

This works fine (with a little violation of the accepted semantics for family relationships). Here is an example showing how we can prove this; we can have the following rules (among others):

$$(\forall x)(\forall y) child(x, y) \iff parent(y, x) \quad (1)$$

$$(\forall x)(\forall y)(\exists z) grandparent(x, y) \iff parent(x, z) \wedge parent(z, y) \quad (2)$$

From the given story we can get the following facts (once again, among others):

$$parent(F, I) \text{ (since } F \text{ is my father)}, \quad (3)$$

$$child(F, I) \text{ (since my father becomes my son-in-law)}. \quad (4)$$

Some of the other rules are: $(parent(W, D), spouse(I, W), spouse(F, D)$, etc), but we don't really need them since we can already prove our conclusion.

From 1 and 4 (take $y = F$ and $x = I$) we derive that

$$parent(I, F) \quad (5)$$

Finally, from 3 and 5 we derive (take $x = I, y = I$ and $z = F$):

$$grandparent(I, I)$$

Thus, "I am my own grandfather"!

4. Give the most general unifier (MGU) for the following pairs of expressions, or say why it does not exist. In all of these expressions variables are upper case x, y, or z.

   (a) $p(a, b, b), p(x, y, z)$.

   (b) $q(y, g(a, b)), q(g(x, x), y)$.

   (c) $older(father(y), y), older(father(x), john)$.

   (d) $knows(father(x), y), knows(x, x)$.

   (e) $r(f(a), b, g(f(a))), r(x, y, z)$.

   (f) $older(father(y), y), older(father(y), john)$.

   (g) $f(g(a, h(b)), g(x, y)), f(g(z, y), g(y, y))$.

Answer

(a) $\{x/a, y/b, z/b\}$. Observe that $\theta' = \{x/a, y/b, z/y\}$ is not an MGU because it is not even a unification of both structures. In fact, $p(a, b, b)\theta' = p(a, b, c) \neq p(x, y, z)\theta' = p(a, b, y)$.

(b) No unifier (x cannot bind to both a and b).

(c) $\{y/john, x/john\}$.

(d) No unifier, because the occurs-check prevents unification of y with $father(y)$.

(e) $\{x/f(a), y/b, z/g(f(a))\}$.

(f) $\{y/john\}$.

(g) $\{x/y, z/a, y/h(b)\}$.

Some more (y, w, z, v, u are all variables):

(a) $p(f(y), w, g(z)), p(v, u, v)$.

(b) $p(f(y), w, g(z)), p(v, u, x)$.

(c) $p(a, x, f(g(y))), p(z, h(w), f(w))$.

(d) $p(z, h(w), g(z)), p(v, u, v)$.

(e) $p(a, h(w), f(g(y))), p(z, x, f(w))$.

Answer

(a) Impossible as v has to match both $f(y)$ and $g(z)$ and f and g are different symbols.

(b) $\{v/f(y), u/w, x/g(z)\}$.

(c) $\{z/a, x/h(w), w/g(y)\}$.

(d) Impossible as v has to unify with both z and $g(z)$.

(e) $\{z/a, x/h(w), w(g(y))\}$.

5. Confirm your answers to the previous question using Prolog predicate `unifiable/3`? Try it! Do you notice anything special in the fourth query? (Hint: read about occurs check).

Answer

(a)
```
?- unifiable(p(a,b,b),p(X,Y,Z),U).
U = [Z=b, Y=b, X=a].
```

Another way to check this is asking for query `?- p(a,b,b) = p(X,Y,Z).` This will give the following result:

```
X = a,
Y = Z, Z = b.
```

Note that the result states that for legibility SWI prints all the variables that are the same in one line. So, the line:

```
Y = Z, Z = b.
```

means that `Y = b` and `Z = b.`

To get the concrete MGU without any extra printing process, we use `unifiable/3`

(b)
```
?- unifiable(q(Y,g(a,b)), q(g(X,X),Y),U).
false.
```

(c)
```
?- unifiable(older(father(Y),Y), older(father(X),john),
U).
U = [Y=john, X=Y].
```

(d)
```
unifiable(knows(father(X),Y), knows(X,X), U).
U = [Y=father(X), X=father(X)].
```

Observe that this succeeds because SWI has occurs check disabled by default. We can unify with occurs check as follows:

```
?- unify_with_occurs_check(knows(father(X),Y),
knows(X,X)).
false.
```

Moreover, we can enable occurs check by default as follows:

```
?- set_prolog_flag(occurs_check,true).
true.
?- knows(father(X),Y) = knows(X,X).
false.
?- unifiable(knows(father(X),Y), knows(X,X), U).
false.
```

(e)
```
?- unifiable(r(f(a), b, g(f(a))), r(X, Y, Z), U).
U = [Z=g(f(a)), Y=b, X=f(a)].
```

(f)
```
?- unifiable(older(father(Y), Y), older(father(Y),
john), U).
U = [Y=john].
```

(g)
```
?- unifiable(f(g(a, h(b)), g(X, Y)), f(g(Z,Y), g(Y,
Y)), U).
U = [X=h(b), Y=h(b), Z=a].
```

6. Write down logical representations for the following sentences, suitable for use with generalised modus ponens. Pay special attention to implications and quantification.

   (a) Horses, cows and pigs are mammals.

   (b) An offspring of a horse is a horse.

   (c) Bluebeard is a horse.

   (d) Bluebeard is Charlie's parent.

   (e) Offspring and parent are inverse relations.

   (f) Every mammal has a parent.

Answer

$\forall x[Horse(x) \supset Mammal(x)]$
$\forall x[Cow(x) \supset Mammal(x)]$
$\forall x[Pig(x) \supset Mammal(x)]$

$\forall x, y[Offspring(x, y) \wedge Horse(y) \supset Horse(x)]$

$Horse(bluebeard)$

$Parent(bluebeard, charlie)$

$\forall x, y[Offspring(x, y) \supset Parent(y, x)]$
$\forall x, y[Parent(x, y) \supset Offspring(y, x)$

$\forall x, \exists y[Mammal(x) \supset Parent(y, x)]$

7. Using the logical expressions from the previous question do the following:

   (a) Draw the proof tree using backward chaining (that is, starting from your query) for the query $(\exists x)Horse(x)$.

   (b) How many solutions for H are there?

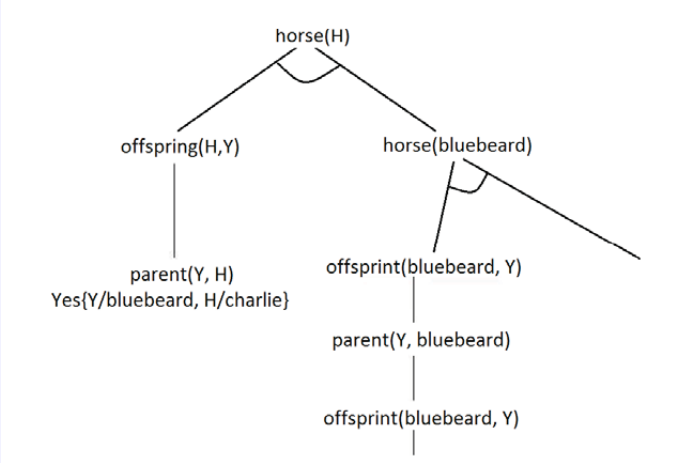
   (c) Repeat the proof using forward chaining (that is, start with the facts and derive further conclusions). Do you get the the same results as before?

   (d) Translate (a)-(f) logic formulas into Prolog.

   (e) Translate $(\exists x)Horse(x)$ into a Prolog query. What answers do you expect Prolog to return and how? Try it!

Answer

(a) The proof tree is shown below. The branch with $Offspring(bluebeard, y)$ and $Parent(y, bluebeard)$ repeats indefinitely, so the rest of the proof is never reached.



(b) Both $bluebeard$ and $charlie$ are horses.

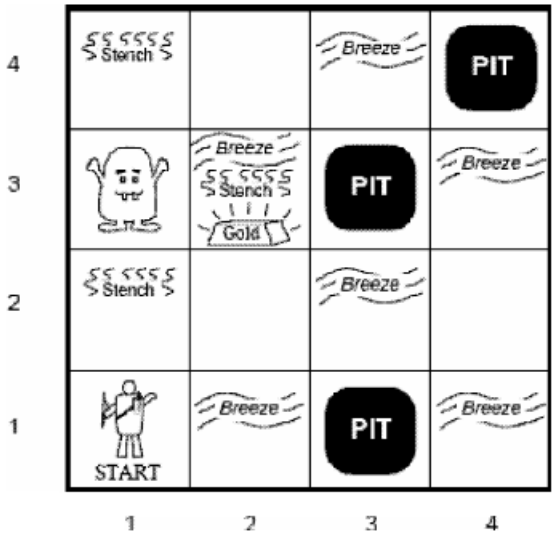
(c) yes.

8. A popular children's riddle is *"Brothers and sisters I have none, but that man's father is my father's son"*. Use the rules of the kinship domain to work out who that man is.

Answer

The son (of the man speaking).

9. For the following Wumpus world:



(a) Develop a notation capturing the important aspects of this domain in first order logic.

(b) How would you express in a first order logic sentence:

i. If a square has no smell then the Wumpus is not in this square or any of the adjacent squares?

ii. If there is stench in a square there must be a Wumpus in this square or in one of the adjacent squares?

(c) Suppose the agent has traversed the path $(1,1) \rightarrow (1,2) \rightarrow (2,2) \rightarrow (1,2)$. How can the agent deduce that the Wumpus is in square [1,3] using the laws of inference of first order logic?

Answer

(a) Similar to previous practical for propositional case but now stated using first order.

(b) i. $\forall x[\neg Smell(x) \supset \forall y((Adjacent(x,y) \vee x = y) \supset \neg WumpusAt(y))]$.

ii. $\forall x[Smell(x) \supset \exists y((Adjacent(x,y) \vee x = y) \wedge WumpusAt(y))]$.

(c) It follows from (ii) above and the fact that the knowledge base includes or entails:

$Adjacent((1,3),(1,2))$
$Smell((1,2))$
$\neg WumpusAt((1,1))$
$\neg Smell((2,2))$(or directly $\neg WumpusAt((2,2))$)

10. **Resolution Proofs.** Consider the following sentences:

(a) Marcus was a man.

(b) Marcus was a Roman.

(c) All men are people.

(d) Caesar was a ruler.

(e) All Romans were either loyal to Caesar or hated him (or both).

(f) Everyone is loyal to someone.

(g) People only try to assassinate rulers they are not loyal to.

(h) Marcus tried to assassinate Caesar.

Carry out the following tasks (refer to book and/or slides from Lecture 5):

(a) Convert each of these sentences into first-order logic and then convert each formula into clausal form. Indicate any Skolem functions or constants used in the conversion.

(b) Convert the negation of the question "Who hated Caesar?" into causal form (with an answer literal)

(c) Derive the answer to this question using resolution. Give the answer in English. In the proof use the notation developed in class.

Answer

First-order sentences:

(a) $Man(marcus)$

(b) $Roman(marcus)$

(c) $\forall x.Man(x) \supset Person(x)$.

(d) $Ruler(caesar)$.

(e) $\forall x.Roman(x) \supset Loyal(x, caesar) \vee Hate(x, caesar)$.

(f) $\forall x.\exists y.Loyal(x, y)$

(g) $TryKill(x, y) \supset Ruler(y) \wedge \neg Loyal(x, y)$

(h) $TryKill(marcus, caesar)$.

Conversions to CNF:

(a) $Man(marcus)$

(b) $Roman(marcus)$

(c) $\neg Man(x) \vee Person(x)$.

(d) $Ruler(caesar)$.

(e) $\neg Roman(x) \vee Loyal(x, caesar) \vee Hate(x, caesar)$.

(f) $Loyal(x, f(x))$ ($f(x)$ is a Skolem function)

(g) $(\neg TryKill(x, y) \vee Ruler(y))$ and $(\neg TryKill(x, y) \vee \neg Loyal(x, y))$.

(h) $TryKill(marcus, caesar)$.

Query: $\exists z.Hate(z, caesar)$.
Query with answer predicate: $\exists z.[Hate(z, caesar) \wedge \neg A(z)]$.
Negation of query with answer predicate: $\neg Hate(z, caesar) \vee A(z)$

The resolution follows easily until we get a clause $A(marcus)$: Marcus hated Caesar.

11. Determine whether the following sentence is valid (i.e., a tautology) using FOL Resolution:

$$\exists x \forall y \forall z((P(y) \Rightarrow Q(z)) \Rightarrow (P(x) \Rightarrow Q(x))).$$

Answer

You do it! Convert this first to CNF to get a set of clauses (with no existential quantification), then do resolution...

12. You say more examples and exercises on resolution? Check here:

https://www.cs.utexas.edu/users/novak/reso.html