# Homework 3: Logic

Arya Farahi

02/20/2017

## 1 Logic Concepts and Manipulation

### 1.1 Validity and Satisfiability

**Use model checking via truth tables to determine whether the following logical statements are valid, satisfiable, or unsatisfiable. In your solution file, include both the truth tables and the statements validity/satisfiability.**

a. $(P \implies Q) \wedge (P \wedge \neg Q)$ : unsatisfiable

| P | Q | Sentence |
|---|---|---|
| TRUE | TRUE | FALSE |
| FALSE | TRUE | FALSE |
| TRUE | FALSE | FALSE |
| FALSE | FALSE | FALSE |

b. $(P \implies Q) \vee (Q \vee \neg P)$ : satisfiable

| P | Q | Sentence |
|---|---|---|
| TRUE | TRUE | TRUE |
| FALSE | TRUE | TRUE |
| TRUE | FALSE | FALSE |
| FALSE | FALSE | TRUE |

c. $(Q \implies (P \vee R)) \wedge \neg[(P \vee R) \vee \neg Q]$ : unsatisfiable

| P | Q | R | Sentence |
|---|---|---|---|
| TRUE | TRUE | TRUE | FALSE |
| FALSE | TRUE | TRUE | FALSE |
| TRUE | FALSE | TRUE | FALSE |
| TRUE | TRUE | FALSE | FALSE |
| FALSE | FALSE | TRUE | FALSE |
| FALSE | TRUE | FALSE | FALSE |
| TRUE | FALSE | FALSE | FALSE |
| FALSE | FALSE | FALSE | FALSE |

### 1.2 Conjunctive Normal Form

**Convert each of the following first-order logic statements into conjunctive normal form. Show every step of your conversions and proceed carefully, lest early mistakes propagate through later steps!**

a. $\forall y, z\ P(y) \iff Q(z)$

- $\forall y, z\ (P(y) \implies Q(z)) \wedge (Q(y) \implies P(y))$ (Translate Bidirectional)
- $\forall y, z\ (\neg P(y) \vee Q(z)) \wedge (\neg Q(z) \vee P(y))$ (Translate implication to disjunction)
- $(\neg P(y) \vee Q(z)) \wedge (\neg Q(z) \vee P(y))$ (Drop Universal Quantifiers)

Distribute and Associate into CNF

$S_1 : \neg P(y) \vee Q(z)$

$S_2 : \neg Q(z) \vee P(y)$

b. $\neg\forall x\ \neg[P(x) \vee Q(x)] \implies R(x)$

- $\neg\forall x\ \neg\neg[P(x) \vee Q(x)] \vee R(x)]$ (Translate implication to disjunction)
- $\neg\forall x\ [P(x) \vee Q(x) \vee R(x)]$ (Move Negation)
- $\exists x\ \neg[P(x) \vee Q(x) \vee R(x)]$ (Move Negation)
- $\exists x\ [\neg P(x) \wedge \neg Q(x) \wedge \neg R(x)]$ (Move Negation)
- $\neg P(X) \wedge \neg Q(X) \wedge \neg R(X)$ (Eliminate existential via. skolemization, where $X$ is a constant)

Distribute and Associate into CNF

$S_1 : \neg P(X)$

$S_2 : \neg Q(X)$

$S_3 : \neg R(X)$

($X$ is a Constant)

c. $\forall x\forall y\exists z[P(x, z) \implies Q(y, z)]$

- $\forall x\forall y\exists z[\neg P(x, z) \vee Q(y, z)]$ (Translate implication to disjunction)
- $\forall x\forall y[\neg P(x, f(x, y)) \vee Q(y, f(x, y))]$, (Eliminate existential via. skolemization, where $f(x, y)$ is a skolem function)
- $\neg P(x, Z) \vee Q(y, Z)$ (Drop Universal Quantifiers)

Distribute and Associate into CNF

$S_1 : P(x, f(x, y)) \vee Q(y, f(x, y))$

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

# 2 Translation

## 2.1 Propositional Logic

**Translate the following English sentences into propositional logic statements. You may use the following propositions about planets: BiggerThanEarth, GasPlanet, HasWater, HasRings, SupportsLife, and Terrestrial.**

**a. Planets that support life have water and are terrestrial.**

$$\text{SupportsLife} \implies [\text{HasWater} \land \text{Terrestrial}]$$

**b. Terrestrial planets are not bigger than Earth.**

$$\text{Terrestrial} \implies \neg\text{BiggerThanEarth}$$

**c. Gas planets with rings are bigger than Earth.**

$$[\text{GasPlanet} \land \text{HasRings}] \implies \text{BiggerThanEarth}$$

**d. Gas planets have rings but other planets do not.**

$$[\text{GasPlanet} \implies \text{HasRings}] \land [\neg\text{GasPlanet} \implies \neg\text{HasRings}]$$

$$[\text{GasPlanet} \implies \text{HasRings}] \land [\text{HasRings} \implies \text{GasPlanet}]$$

$$\text{GasPlanet} \iff \text{HasRings}$$

**e. Planets are either terrestrial or gaseous (but not both).**

$$[\text{GasPlanet} \lor \text{Terrestrial}] \land \neg[\text{GasPlanet} \land \text{Terrestrial}]$$

$$[\text{GasPlanet} \lor \text{Terrestrial}] \land [\neg\text{GasPlanet} \lor \neg\text{Terrestrial}]$$

## 2.2 Definitions

**Consider the following English language sentences:**

1. **Rob fed Spot before Jim fed Rover.**
2. **Every dog chases a cat or a dog (or both).**
3. **There are at least two different cats.**
4. **Rob likes dogs that are bigger than any cat.**
5. **Jim fed some cat before Rover was fed.**

**Define any constants, predicates, and functions that you will use to best represent the five sentences in First-Order Logic. Some examples are Likes(x,y) and Fed(x,y,z) (with feeder x, one being fed y, and time of feeding z).**

constants:

- Rob
- Spot
- Jim
- Rover

predicates:

- Likes(x,y)
- Bigger(x, y) (x is bigger than y)

- Fed(x,y,z) (with feeder x, one being fed y, and time of feeding z).
- Dog(x)
- Cat(x)
- Time(x)
- Chase(x, y) (x chases y)
- Before(x, y) (x < y where x, y are time variables)

and functions: We can use existential quantifier so there is no need for functions but if we want to use functions then

- FedBy(x)
- ChasedBy(x)

are two functions that we need.

**b. Translate each of the 5 sentences into First-Order Logic using your symbols from part (a).**

1. $\exists x, y\ \text{Time}(x) \land \text{Time}(y) \land \text{Fed}(\text{Rob}, \text{Spot}, x) \land \text{Fed}(\text{Jim}, \text{Rover}, y) \land \text{Before}(x, y)$
2. $\forall x\ \text{Dog}(x) \implies [(\exists y\ \text{Chase}(x, y) \land \text{Cat}(y)) \lor (\exists z\ \text{Chase}(x, z) \land \text{Dog}(z) \land \neg(x = z))]$
3. $\exists x, y\ \text{Cat}(x) \land \text{Cat}(y) \land \neg(x = y)$
4. $\forall x\ \text{Like}(\text{Rob}, x) \land \text{Dog}(x) \implies [\forall y\ \text{Bigger}(x, y) \land \text{Cat}(y)]$
5. $[\exists x, y\ \text{Time}(x) \land \text{Time}(y) \land \text{Before}(x, y)] \land [(\exists z\ \text{Fed}(\text{Jim}, z, x) \land \text{Cat}(z)) \land (\exists w\ \text{Fed}(\text{Rover}, w, y))]$

# 3 Inference

## 3.1 Unification

**For each of the following pairs of legal first-order logic literals, find the most general unifier. If none exists, explain why not. You should assume that all variables are universally quantified. Note that your task is to execute the unification algorithm, not to interpret the semantics of the logical statements.**

**a. Hates**$(Jim, x)$ **and Hates**$(y, Bob)$

$\{y/Jim, x/Bob\}$

**b. Hates**$(Jim, x)$ **and Loves**$(y, z)$

Fails. Because Hates $\neq$ Loves. Unification must be done with the same predicate.

**c. Sees**$(Owner(x), x$, **Home**$(Mittens))$ **and Sees**$(Owner(y)$, **Spot, Home**$(y))$

Fails, because $\{x/Spot, y/Mittens\}$ But Home$(Spot) \neq$ Home$(Mittens)$. We do not have more information so we cannot unify them.

**d. Older**(**Father**$(x), x)$ **and Older**(**Father**$(y), Spot)$

$\{x/Spot, y/Spot\}$

**e. Loves(Friend(Spot), x) and Loves(y, y)**

$\{x/Friend(Spot), y/Friend(Spot)\}$

## 3.2 Resolution-Refutation

**The following information about University of Michigan classes can be represented in conjunctive normal form.**

1. Classes with thicker books are harder.
   S1 = $\neg ThickerBook(x, y) \vee HarderClass(x, y)$
2. Hardness of classes is transitive.
   S2 = $\neg HarderClass(u, v) \vee \neg HarderClass(v, w) \vee HarderClass(u, w)$
3. Programming classes have thicker books than some other classes.
   S3 = $\neg ProgrammingClass(z) \vee ThickerBook(z, SomeOtherClass(z))$
4. EECS 492 has a thicker book than EECS 376.
   S4 = $ThickerBook(EECS492, EECS376)$
5. EECS 376 is a harder class than ENGR 101.
   S5 = $HarderClass(EECS376, ENGR101)$
6. ENGR 101 is a programming class.
   S6 = $ProgrammingClass(ENGR101)$
7. ECON 101 is not a programming class.
   S7 = $\neg ProgrammingClass(ECON101)$

**a. Use resolution-refutation (p.345) with an answer literal to find some class that EECS 492 is harder than. In other words, find a c such that** $HarderClass(EECS492, c)$ **is entailed. Show all steps in your work, including which sentences get resolved and how variables are bound at each step.**

**Continue this resolution-refutation process to find the other two values of c for which $HarderClass(EECS492, c)$ is entailed. Show all steps in your work.**

KB $\models HarderClass(EECS492, c)$

We should prove that KB $\wedge \neg HarderClass(EECS492, c)$ is an empty set.

Let's assume sentence 8 is $\neg HarderClass(EECS492, c)$

9. $HarderClass(EECS492, EECS376)$ (from 1&4, $\{x/EECS492, y/EECS376\}$)

10. Empty Set. (from 9&8) so there exists a class.

Now let's assume that sentence 8 is $\neg HarderClass(EECS492, c) \vee \text{Ans}(c)$, where $c$ is the placeholder for the answer.

9. $HarderClass(EECS492, EECS376)$ (from 1&4, $\{x/EECS492, y/EECS376\}$)

10. $\text{Ans}(EECS376)$ (from 9&8, $\{c/EECS376\}$)

11. $HarderClass(EECS492, EENG101)$ (from 9&3, $\{x/EECS492, y/ENGR101\}$)

12. $\text{Ans}(EENG101)$ (from 11&8, $\{c/EENG101\}$)

13. $ThickerBook(ENGR101, SomeOtherClass(ENGR101))$ (from 6&3, $\{z/ENGR101\}$)

14. $HarderClass(ENGR101, SomeOtherClass(ENGR101))$
(from 13&1, $\{x/ENGR101, y/SomeOtherClass(ENGR101)\}$)

15. $\neg HarderClass(EECS492, ENGR101) \vee \neg HarderClass(ENGR101, w)$
$\neg HarderClass(EECS492, w)$
(from 11&2, $\{u/EECS492, v/ENGR101\}$)

16. $HarderClass(EECS492, SomeOtherClass(ENGR101))$
(from 15&14, $\{w/SomeOtherClass(ENGR101)\}$)

17. $\text{Ans}(SomeOtherClass(ENGR101))$ (from 16&8, $\{c/EENG101\}$)

Ans $= \{EECS376, ENGR101, SomeOtherClass(ENGR101)\}$

# 4 Prolog

**For this task you will need access to the Prolog programming language. If your system does not have Prolog installed, you can download SWI Prolog at `www.swi-prolog.org`. If youre having trouble getting started, a good Prolog tutorial can be found `www.learnprolognow.org`.**

**Consider the following First-Order Logic expressions,**

1. $\forall x\ University(x) \wedge Wealthy(x) \wedge GoodAt(x, Engr) \Rightarrow GoodAt(x, CS)$
2. $\forall x\ University(x) \wedge focusOn(x, Engr) \Rightarrow GoodAt(x, Engr)$
3. $\forall x\ University(x) \wedge focusOn(x, Football) \Rightarrow GoodAt(x, Football)$
4. $\forall x\ University(x) \wedge GoodAt(x, Football) \wedge focusOn(x, Engr) \Rightarrow Wealthy(x)$
5. $\forall x\ \forall y\ University(x) \wedge Beat(x, y, Football) \wedge University(y) \wedge GoodAt(y, Football) \Rightarrow GoodAt(x, Football)$
6. $\exists x\ University(x) \wedge Beat(x, Michigan, Football)$
7. $University(Michigan) \wedge University(MIT) \wedge University(OSU) \wedge University(Alabama)$
8. $focusOn(Alabama, Football) \wedge focusOn(MIT, Engr) \wedge focusOn(OSU, Football) \wedge focusOn(Michigan, Football)$
9. $beat(Michigan, OSU, Football)$

**a. Create a file where you convert the preceding expressions as exactly as possible into Prolog syntax. You can create one or more functions as necessary. Include this file (or a paste of it) with your assignment submission.**

```
goodAt(X,cs) :- university(X), wealthy(X), goodAt(X,engr).

goodAt(X,engr) :- university(X), focusOn(X,engr).

goodAt(X,football) :- university(X), focusOn(X,football).

wealthy(X) :- university(X), goodAt(X,football), focusOn(X,engr).

goodAt(X,football) :- university(X), beat(X,Y,football), university(Y), goodAt(Y,football).

university(universityBeatenBy(michigan,football)).

beat(universityBeatenBy(michigan,football),michigan,football).

university(michigan).

university(mit).

university(osu).

university(alabama).

focusOn(alabama,football).

focusOn(mit,engr).

focusOn(osu,football).

focusOn(michigan,football).

beat(michigan,osu,football).
```

**b. Use Prolog to answer the queries: Who is good at football? and Who is good at Engr? Extract all answers. Include the output of Prolog with your assignment submission (cut and paste is fine).**

Good at Football: Michigan, OSU, ALABAMA, UniversityBeatenBy(Michigan, Football)

```
?- goodAt(X1, football).
X1 = michigan
X1 = osu
X1 = alabama
X1 = universityBeatenBy(michigan, football)
X1 = universityBeatenBy(michigan, football)
X1 = michigan
false.
```

Good at Engr: MIT

```
?- goodAt(X1, engr).
X1 = mit
false.
```

**c. Now say we want to correct or change a statement in our knowledge base using assert and retract. Retract the statement** $focusOn(Michigan; Football)$ **add the statement** $focusOn(Michigan, Engr)$ **to your KB using assert and retract. Now extract all of the answers of asking rerunning the commands from (b). What new answers did you get and (briefly) why? What answers from part b did you no longer get and (briefly) why? Provide your prolog statements as well.**

Good at Football: Michigan, OSU, ALABAMA, UniversityBeatenBy(Michigan, Football)

```
?- goodAt(X,football).
X1 = osu
X1 = alabama
X1 = universityBeatenBy(michigan, football)
X1 = michigan
false.
```

Good at Engr: MIT, Michigan

```
?- goodAt(X,engr).
X = michigan
X = mit
false.
```

Well, we added a sentence to the KB which leads to Michigan is good at Engr so we expect it finds it, by combining the new sentence and sentence 2. Also, a University which s going to focus at Football would be good at football, and still it is in our KB (sentence 8 and 3), even though we removed the explicit sentence which says Michigan is good at football (result of focusOn and sentence 3). Now it finds it once because the fact that we removed Michigan is good at football. There are two different ways to get Michigan is good at Football, one is using 8 and 3. The other is by using sentence 9, 8, 3, 5. Here we remove one of them. The same thing is happening for UniversityBeatenBy(UM, Football). There were two way of getting it, now there is only one way.

**d. What happens when you invert the order of premises for expression 5? Change (a copy of) your file from part (a), load it into a new instance of Prolog and run the same queries as in part b. Explain how (if at all) this changes what Prolog tells you, and why. Include the output of Prolog with your assignment submission (you don't need to include the slightly modified Prolog code).**

Good at Football: Michigan, OSU, ALABAMA, UniversityBeatenBy(Michigan, Football)

```
?- goodAt(X1, football).
X = michigan
X = osu
X = alabama
X = universityBeatenBy(michigan, football)
```

```
X = michigan
X = universityBeatenBy(michigan, football)
ERROR: [Thread 2] Out of local stack
```

Good at Engr: MIT

```
?- goodAt(X,engr).
X = mit
false.
```

The good at engr is like before. Because the expression we changed had nothing to do with engineering question and search.

The second one finds all solution in different order, though at the very end it enters an inf. loop. It uses expression 5 which is good(X,football) and they by unifying x/y it has to prove that it is goodAt(X,football) and it goes through an inf. loop