Exercise 3: Logic and ANTLR

Formal Methods II, Fall Semester 2013

Solution Sheet

Propositional logic

1. (2 points)

   (a) The solution is stated using a compressed truth table.

p	q	p	$\rightarrow$	$(q$	$\rightarrow$	$p)$
T	T	T	T	T	T	T
F	T	F	T	T	F	F
T	F	T	T	F	T	T
F	F	F	T	F	T	F

   (b) The solution is stated again as a truth table:

p	q	r	$(p$	$\rightarrow$	$(q$	$\rightarrow$	$r))$	$\rightarrow$	$((p$	$\rightarrow$	$q)$	$\rightarrow$	$(p$	$\rightarrow$	$r))$
T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
F	T	T	F	T	T	T	T	T	F	T	T	T	F	T	T
T	F	T	T	T	F	T	T	T	T	F	F	T	T	T	T
F	F	T	F	T	F	T	T	T	F	T	F	T	F	T	T
T	T	F	T	F	T	F	F	T	T	T	T	F	T	F	F
F	T	F	F	T	T	F	F	T	F	T	T	T	F	T	F
T	F	F	T	T	F	T	F	T	T	F	F	T	T	F	F
F	F	F	F	T	F	T	F	T	F	T	F	T	F	T	F

   (c) Counter-example: For a true p and a true q, or a false p and a true q, the expression is false. Therefore, it cannot be a tautology.

   (d) Using a truth table, one can show that this expression is always false, no matter whether t is true or false:

p	$\neg$	$(p$	$\vee$	$\neg$	$p)$
T	F	T	T	F	T
F	F	F	T	T	F

2. (3 points) For this question, a critical equivalence to remember is $a \to b \leftrightarrow \neg a \vee b$. Another important equivalence is that $a \leftrightarrow b$ is equivalent to $(a \to b) \wedge (b \to a)$

   (a) $\neg c \vee (\neg b \vee a)$

   (b) $(b \wedge \neg a \vee c) \wedge (\neg c \vee \neg b \vee a)$. After applying de Morgan's law.

   (c) $(a \wedge \neg b) \vee (\neg a \wedge b)$. (Brackets for clarification).

3. (1 point) Yes, this is possible. Under the premise that all statements of propositional logic can be expressed using $\vee, \wedge$ and $\neg$, we have to come up with a way to "emulate" $\wedge$ with $\vee$ and $\neg$ only. This can be done using $\neg(\neg a \vee \neg b)$:

a	b	$a \wedge b$	$\neg$	$(\neg$	a	$\vee$	$\neg$	$b)$
T	T	T	T	F	T	F	F	T
F	T	F	F	T	F	T	F	T
T	F	F	F	F	T	T	T	F
F	F	F	F	T	F	T	T	F

   Unsurprisingly, applying de Morgan's law to $\neg(\neg a \vee \neg b)$ results in $a \wedge b$.

4. (2 points)

   Using the rule for rewriting the implication as mentioned above, followed by de Morgan's law, the distributive law and omission of vera ($\top$, "tautologies"):

$$\begin{aligned}(C \vee D) \to (C \wedge D) &\leftrightarrow \neg(C \vee D) \vee (C \wedge D) \\ &\leftrightarrow (\neg C \wedge \neg D) \vee (C \wedge D) \\ &\leftrightarrow \underbrace{(\neg C \vee C)}_{\top} \wedge (\neg C \vee D) \wedge (C \vee \neg D) \wedge \underbrace{(D \vee \neg D)}_{\top} \\ &\leftrightarrow (\neg C \vee D) \wedge (C \vee \neg D)\end{aligned}$$

Predicate logic

5. (4 points) The sample solutions stated below are not the only possibilities. Please check our corrections on your solutions for clarifications.

   (a) Note how b is used to ensure that m and z are the highest buildings of the respective cities. $>$ and $\leq$ refer to the building sizes.

$$\begin{aligned}&(\exists m, z)(\text{isBuilding}(m) \wedge \text{inManhattan}(m) \wedge \text{isBuilding}(z) \wedge \text{inZurich}(z) \wedge \\ &\quad (\forall b)(\text{isBuilding}(b) \wedge \text{inManhattan}(b) \to b \leq m) \wedge \\ &\quad (\forall b)(\text{isBuilding}(b) \wedge \text{inZurich}(b) \to b \leq z) \wedge m > z)\end{aligned}$$

   (b) $(\exists p)(\exists c)(\text{person}(p) \wedge \text{car}(c) \wedge \text{young}(p) \wedge \text{fast}(c) \wedge \text{safe}(c) \wedge \text{owns}(p, c))$

(c) $(\exists p)(\text{person}(p) \wedge (\forall m)(\text{moped}(m) \wedge (\neg\text{slow}(m)) \rightarrow \text{likes}(p, m)))$

(d) $\neg(\exists p, \exists c)(\text{person}(p) \wedge \text{car}(c) \wedge \text{slow}(c) \wedge \text{likes}(p, c))$

Bonus $(\forall p)(\text{person}(p) \rightarrow (\text{saves}(p, p) \veebar \text{saves}(\text{ChuckNorris}, p)))$

Another nice solution that was handed in:
$(\forall p)(\text{person}(p) \wedge (\neg\text{saves}(p, p) \leftrightarrow \text{saves}(\text{ChuckNorris}, p)))$

A common mistake was that "if and only if" ("genau dann wenn") was modelled using an implication.

6. (2 points)

(a) The formula was formed from the atomic formulas (x = 0) and (x = S(y)) by using the following rules:

$$\neg(x = 0) \quad \text{rule (ii) from the Formula def. in the script}$$

$$(\exists y)(x = S(y)) \quad \text{(iii)}$$

$$\neg(x = 0) \rightarrow (\exists y)(x = S(y)) \quad \text{(ii)}$$

$$(\forall x)(\neg(x = 0) \rightarrow (\exists y)(x = S(y))) \quad \text{(iii)}$$

(b) The meaning of the formula is that every nonzero natural number has its predecessor.

Note: Merely restating the formula in natural language (e.g. "for all x not equal to zero there exists an y for which holds x = y + 1") was not enough to get points for this subtask.

7. (3 points)

(a)

$$\begin{aligned} &\neg\exists x(\forall y(P(x, y))) \\ \leftrightarrow &\forall x(\neg\forall y(P(x, y))) \\ \leftrightarrow &\forall x(\exists y(\neg P(x, y))) \end{aligned}$$

(b) Counter example: $P(x) := x > 0, \quad Q(x) := x < 0$

(c) First we substitute y for x in the subformula $\exists x(Q(x))$, which is a legal operation, giving:

$\forall x(P(x)) \wedge \exists y(Q(y))$
since y does not affect $P(x)$ we can write
$\forall x(\exists y(P(x) \wedge Q(y)))$

ANTLR Flashback

8. (3 points) The following changes to the provided grammar are required:

   (a) In order to fix the grammar, the rule for the conjunction has to be defined such that it actually is able to consume a conjunction. Assuming that the rule for the disjunction is correct and understood, a similar rule for the conjunction can be implemented. Note that the order of the rules is important because it takes care of the precedence.

   (b) The rules for the implication and equivalence operators can be implemented similarly to the disjunction and the (fixed) conjunction rules. Again, the order is important to take care of the precedence. Note that the `entry` rule has to be modified accordingly. This steps also requires an adaption of the `parenthesis` rule. Without this modification, `<->` and `->` could not be used inside of parenthesis.

   (c) Once the grammar is fixed, the `atom` rule can be refined such that it also accepts a variable. A lexer rule for `VARIABLE` defines the syntax of such a variable.

   Once all of the above is taken care of, the result may look like this:

```
1  grammar boolean;
2
3  entry               : equivalence EOF ;
4  equivalence         : implication
5                        ( '<->' implication)* ;
6  implication         : disjunction
7                        ( '->' disjunction)* ;
8  disjunction         : conjunction
9                        ( '||' conjunction)* ;
10 conjunction         : negation
11                       ( '&&' negation)* ;
12 negation            : '!' parenthesis
13                       | parenthesis ;
14 parenthesis         : '(' equivalence ')'
15                       | atom ;
16 atom                : 'T' | 'F' | VARIABLE ;
17 VARIABLE            : ('a'..'z')+ ;
```