CMSC 471 HOMEWORK FOUR Sprint 2016

Please do not distribute or post on the Web or Internet.

Total points: 100

1. Checking Validity (10 points, 2 each for 1.2-1.6)

2. p → p
>>> tt_true(expr('P >> P'))
True

3. p → (p v q)
>>> tt_true(expr('P >> (P | Q)'))
True

4. (p v q) → p
>>> tt_true(expr(' (P | Q) >> P'))
False

5. ((A ∧B) →C) ↔ (A → (B → C))
>>> tt_true(expr(' ((A & B) >> C) <=> (A >> (B >> C)) '))
True

6. ((a → b) → a) → a
>>> tt_true(expr(' ((A >> B) >> A) >> A '))
True

2. Satisfiability (9 points, 3 each)

2. ALIVE→ ¬DEAD ∧ ¬ALIVE∧ ¬DEAD
>>> dpll_satisfiable(expr('Alive>>~Dead&~Alive&~Dead'))
{Alive: False, Dead: False}

3. P → ¬ P v P
>>> dpll_satisfiable(expr('P>>~P|P'))
{P: True}

4. ~ (P v ¬ P)
>>> dpll_satisfiable(expr('~(P|~P)'))
False

3. Propositional Consequence (24 points, 3 each)

2. p |= p ^ q
False

3. p |= p v q
True

4. p |= ~~p
True

5. p-->q |= ~p-->~q
False

6. ~p |= p-->q
True

7. ~q |= p-->q
False

8. p, p-->q |= q
True

9. ~p, q-->p |= ~q
True

4. English to FOL (30 points, 3 each)

There are usually several reasonable ways to express natural language sentences in logic. One source of variation is what to leave implicit and what to make explicit, e.g., omitting obvious types as in rendering 'every person loves their mother' as Ax mother(x,y) => loves(y,x). Another comes from the application of standard tautologies, e.g., you can encode 'no man is an island' as ~Ex man(x) ^ island(x) or as Ax man(x) => ~island(x).

2. Everything is either dead or alive.

Here are two ways

Ax alive(x) v dead(x)
~Ex ~alive(x) ^ ~dead(x)

3. Dead things are not animate.

Here are two ways to do it.

Ax dead(x) => ~ animate(x)
~Ex dead(x) ^ animate(x))

4. Zombies are not alive but they are animate

Ax zombie(x) => ~ alive(x) ^ animate(x)

5. Good food is not cheap and cheap food is not good.

Here are three reasonable answers:

Ax ((food(x) ^ good(x)) => ~cheap(x)) ^ ((food(x) ^ cheap(x)) => ~good(x)))
Ax food(x) => (good(x) => ~ cheap(x)) ^ (cheap(x) => ~good(x))

But, this is redundant, since P=>Q = ~Q=>~P, so we can also write it as

Ax food(x) => (good(x) => ~ cheap(x))

8. If John has a sister, she is smart.

Two simple ways to say this

Ax hasSister(John,x) => smart(x)
~Ex hasSister(john,x) ^ ~ smart(x)

9. Every person is either male or female and no person can be both male and female.

Here are some variations:

Ax person(x) => ((male(x) ^ ~female(x)) v (~male(x) ^ female(x)))

Ax person(x) => ((male(x) v female(x)) ^ (male(x) => ~female(x)))

Alternatively we might assume that male and female apply only to persons and then have

Ax ((male(x) v female(x)) ^ (male(x) => ~female(x)))

10. The enemy of your enemy is your friend.

Ax,y,z enemy(x,y) ^ enemy(y,z) => friend(x,z)

11. An ancestor of your ancestor is your ancestor.

Ax,y,z ancestor(x,y) ^ ancestor(y,z) => ancestor(x,z)

5. CNF and horn clauses (27 points, 3 each for 5.2-10)

1. ∀x knows(x x) ^ likes(x, x)

(a) Everyone knows and likes himself;
(b) this can be rewritten as a horn clause;
(c) set of clauses: [knows(x, x), likes(x, x)]

2. ∀x ∀y married(x, y) → loves(x, y) v hates(x, y)

(a) if you are married to someone, you either love them or hate them
(b) cannot be written as horn clauses
(c) [loves(x,y) V ~married(x,y) V hates(x,y))]

3. ∀x ∀y loves(x, y) ↔ loves(y, x)

(a) Everybody is loved by everyone they love
(b) can be written as 2 horn clause statements
(c) [(~loves(x,y) v loves(y,x)) , (~loves(y,x) v loves(x,y))]

4. ∀x ∀y dating(x, y) v engaged(x, y) → knows(x, y) ^ likes(x, y)

(a) You know and like the people you date or are engaged to.
(b) can be written as 4 horn statements
(c) [(~dating(x, y) v knows(x, y)), (~engaged(p, q) v knows(p, q)) ,(~dating(r, s) v likes(r, s)), (~engaged(t, u) v likes(t, u))]

5. ∀x ∀y loves(x, y) → ¬ hates(x, y)

(a) You don't hate someone you love
(b) can be written as horn clause
(c) [~loves(x,y) v ~hates(x,y)]

6. ∀x ∀y ¬ knows(x, y) → ¬ likes(x, y)

(a) You don't like someone you don't know.
(b) can be written as horn clause
(c) [knows(x,y) v ~likes(x,y)]

7. ∀x ∃y knows(x, y) ^ hates(x, y)

(a) Everyone has someone they know and hate
(b) can be written as horn clause
(c) [knows(x, hated(x)), hates(x, hated(x))]

Note: hated is a Skolem function that refers to one of the people who its argument knows and hates

8. ∃y ∀x knows(x, y) ^ hates(x, y)

(a) There is someone that everyone knows and everyone hates
(b) can be written as horn clause
(c)) [knows(x,A), hates(x, A)]

Note: A is a Skolem constant

9. ¬ (∀x loves(x, x))

(a) There is someone who does not love himself.
(b) can be written as horn clause
(c) [~loves(x,A)]

Note: A is a Skolem constant

10. ¬ (∃x ∀y knows(x, y))

(a) There is no one who knows everyone

(b) can be written as horn clause
(c) [~knows(x, stranger(x))]

stranger is a Skolem function