Tristan Adams
4/11/16
CMSC 471: Intro to AI
HW 4

1. Checking Validity
    a. P v ¬P
        i. `>>> tt_true(expr('P | ~P'))`
           `True`
    b. P → P
        i. `>>> tt_true(expr('P >> P'))`
           `True`
    c. P→ (P v Q)
        i. `>>> tt_true(expr('P >> (P | Q)')`
           `True`
    d. (P v Q) → P
        i. `>>> tt_true(expr('(P | Q) >> P')`
           `False`
    e. ((A ∧B) →C) ↔ (A → (B → C))
        i. `>>> tt_true(expr('((A & B) >> C) <=> (A >> (B >>`
           `C))'))`
           `True`
    f. ((A →B) →A) →A
        i. `>>> tt_true(expr('((A >> B) >> A) >> A'))`
           `True`
2. Satisfiability
    a. P ∧ Q
        i. `>>> dpll_satisfiable(expr('P & Q'))`
           `{P: True, Q: True}`
    b. ALIVE→ ¬DEAD ∧ ¬ALIVE∧ ¬DEAD
        i. `>>> dpll_satisfiable(expr('ALIVE >> ~DEAD &`
           `~ALIVE & ~DEAD'))`
           `{ALIVE: False, DEAD: False}`
    c. P → ¬ P v P
        i. `>>> dpll_satisfiable(expr('P >> ~P | P'))`
           `{P: True}`
    d. ~ (P v ¬ P)
        i. `>>> dpll_satisfiable(expr('~(P | ~P)'))`
           `False`
3. Propositional Consequence
    a. P ∧ Q ⊨ P
        i. True
    b. P ⊨ P ∧ Q
        i. False
    c. P ⊨ P v Q
        i. True

d. $P \models \neg \neg P$
    i. True
e. $P \rightarrow Q \models \neg P \rightarrow \neg Q$
    i. False
f. $\neg P \models P \rightarrow Q$
    i. True
g. $\neg Q \models P \rightarrow Q$
    i. False
h. $P \wedge (P \rightarrow Q) \models Q$
    i. True
i. $(\neg P) \wedge (Q \rightarrow P) \models \neg Q$
    i. True

4. English to FOL
    a. There is no largest prime number.
        i. ~(Ex number(x) ^ prime(x) ^ (Ay number(y) ^ prime(y) -> x>=y))
    b. Everything is either dead or alive. Is either meaning that everything is dead OR alive (so they can be both) or everything is dead XOR alive, where something can't be both?
        i. Ax dead(x) XOR alive(x)
        ii. Ax dead(x) v alive(x)
    c. Dead things are not animate.
        i. Ax (dead(x) => ~animate(x))
    d. Zombies are not alive but they are animate.
        i. Ax (zombie(x)) => (animate(x) ^ ~alive(x))
    e. Good food is not cheap and cheap food is not good.
        i. Af ((good(f) => ~cheap(f)) ^ (cheap(f) => ~good(f)))
    f. John has exactly two brothers.
        i. Ex (John(x)) => (brothers(x) = 2)
    g. No person can have two mothers.
        i. ~(Ap (person(p) ^ (mothers(p) = 2)))
    h. If John has a sister, she is smart.
        i. (Ex (John(x) ^ (sister(x) = 1))) => (Ey (Johns_Sister(y) ^ smart(y)))
    i. Every person is either male or female and no person can be both male and female.
        i. Ap (male(p) v female(p)) ^ ~(male(p) ^ female(p))
        ii. This could also be simplified using XOR:
            1. Ap (male(p) XOR female(p))
    j. The enemy of your enemy is your friend.
        i. Ap (Ee (Ef ((f => enemy(e)) ^ e => enemy(p)))) => (f => friend(p))
    k. An ancestor of your ancestor is your ancestor.
        i. Ap (Ea (Eb (b => ancestor(a) ^ a => ancestor(p)))) => (b => ancestor(p))

5. CNF and Horn Clauses
    a. ∀x knows(x, x) ∧ likes(x, x)
        i. For all people, they know and like their self
        ii. this can be rewritten as a horn clause
        iii. set of clauses: [knows(x, x), likes(x, x)]
    b. ∀x ∀y married(x, y) → loves(x, y) v hates(x, y)
        i. For all people, those who are married love or hate their spouse, and possibly both.
        ii. Cannot be rewritten as horn clause
        iii. ~married(x, y) v (loves(x, y) v hates(x, y))
    c. ∀x ∀y loves(x, y) ↔ loves(y, x)
        i. For all people, the person they love loves them back.
        ii. Can be rewritten as horn clause
        iii. (~loves(x, y) v loves(y, x)) ^ (~loves(y, x) v loves(x, y))
    d. ∀x ∀y dating(x, y) v engaged(x, y) → knows(x, y) ∧ likes(x, y)
        i. For all people, if one are dating and/or engaged to somebody else, they know and like that other person
        ii. Can be rewritten as horn clause
        iii. ~dating(x, y) v ~engaged(x, y) v ~(knows(x, y) v likes(x, y))
    e. ∀x ∀y loves(x, y) → ¬ hates(x, y)
        i. For all people who love another, they can't hate that person.
        ii. Can be rewritten as horn clause
        iii. ~loves(x, y) v ~hates(x, y)
    f. ∀x ∀y ¬ knows(x, y) → ¬ likes(x, y)
        i. For all people, if they don't know another person they can't like them.
        ii. Can be rewritten as horn clause
        iii. knows(x, y) v ~likes(x, y)
    g. ∀x ∃y knows(x, y) ∧ hates(x, y)
        i. For all people there is some person they know who they also hate.
        ii. Can be rewritten as horn clause
        iii. ~(~knows(x, y) v ~hates(x, y))
    h. ∃y ∀x knows(x, y) ∧ hates(x, y)
        i. For some people, everybody who knows them hates them.
        ii. Can be rewritten as a horn clause
        iii. ~(~knows(x, y) v ~hates(x, y))
    i. ¬ (∀x loves(x, x))
        i. Not all people love themselves.
        ii. Cannot be written as a horn clause
        iii. ~loves(x, x)
    j. ¬ (∃x ∀y knows(x, y))
        i. There aren't some people that everybody knows.
        ii. Cannot be rewritten as a horn clause
        iii. ~knows(x, y)