CS 161 HW5

Yu-Chen Lin 705315195 ericlin8545@cs.ucla.edu

Question 1.

(a) Valid

Smoke	Smoke => Smoke
T	**T**
F	**T**

(b) Neither

Smoke	Fire	Smoke => Fire
T	T	**T**
T	F	**F**
F	T	**T**
F	F	**T**

(c) Valid

Smoke	Fire	-Fire	Smoke v Fire v -Fire
T	T	F	**T**
T	F	T	**T**
F	T	F	**T**
F	F	T	**T**

(d) Neither

Smoke	Fire	Smoke=>Fire	-Smoke=>-Fire	(Smoke=>Fire) => (-Smoke=>-Fire)
T	T	T	T	**T**
T	F	F	T	**T**
F	T	T	F	**F**
F	F	T	T	**T**

(e) Neither
S: Smoke, F: Fire, H: Heat

S	F	H	S => F	S v H	(SvH) => F	(S=>F) => ((SvH) => F)
T	T	T	T	T	T	**T**
T	T	F	T	T	T	**T**
T	F	T	F	T	F	**T**
T	F	F	F	T	F	**T**
F	T	T	T	T	T	**T**
F	T	F	T	F	T	**T**
F	F	T	T	T	F	**F**
F	F	F	T	F	T	**T**

(f) Valid

S	F	H	S∧H	(S∧H)=>F	S=>F	H=>F	(S=>F)v(H=>F)	((S∧H)=>F)⇔((S=>F)v(H=>F))
T	T	T	T	T	T	T	T	**T**
T	T	F	F	T	T	T	T	**T**
T	F	T	T	F	F	F	F	**T**
T	F	F	F	T	F	T	T	**T**
F	T	T	F	T	T	T	T	**T**
F	T	F	F	T	T	T	T	**T**
F	F	T	F	T	T	F	T	**T**
F	F	F	F	T	T	T	T	**T**

Question 2.

(a) {x/A, y/B, z/B}
(b) Fail
(c) {y/John, x/John}
(d) Fail

Question 3.

(a) John likes all kinds of food.
A x, Food(x) => Like(John, x)

Apples are food.
A x, Apple(x) => Food(x)

Chicken is food.
A x, Chicken(x) => Food(x)

Anything anyone eats and isn't killed by is food.
A x, (A y, Eat(y, x) & -Kill(x, y)) => Food(x)

If you are killed by something, you are not alive.
A x, (E y, Kill(y, x)) => -Alive(x)

Bill eats peanuts and is still alive.
E x, Eat(Bill, x) & Alive(Bill) & Peanut(x)

Sue eats everything Bill eats
A x, Eat(Bill, x) => Eat(Sue, x)

(b) A x, Food(x) => Like(John, x)
-Food(x) v Like(John, x)

A x, Apple(x) => Food(x)
-Apple(x) v Food(x)

A x, Chicken(x) => Food(x)
-Chicken(x) v Food(x)

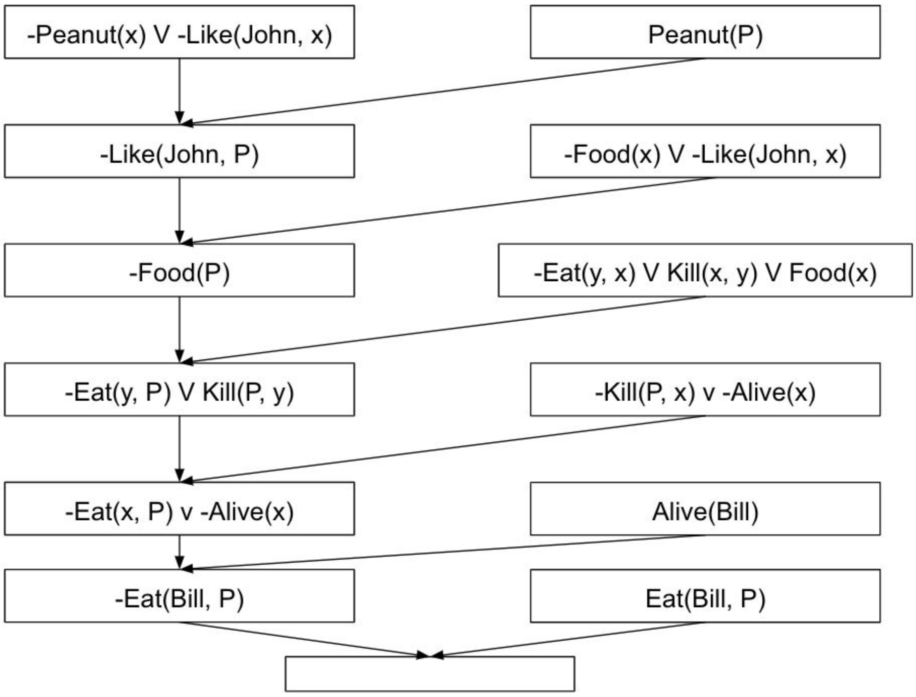
A x, (A y, Eat(y, x) & -Kill(x, y)) => Food(x)
-Eat(y, x) v Kill(x, y) v Food(x)

A x, (E y, Kill(y, x)) => -Alive(x)
-Kill(P, x) v -Alive(x)

E x, Eat(Bill, x) & Alive(Bill) & Peanut(x)
Eat(Bill, P) ∧ Alive(Bill) ∧ Peanut(P)

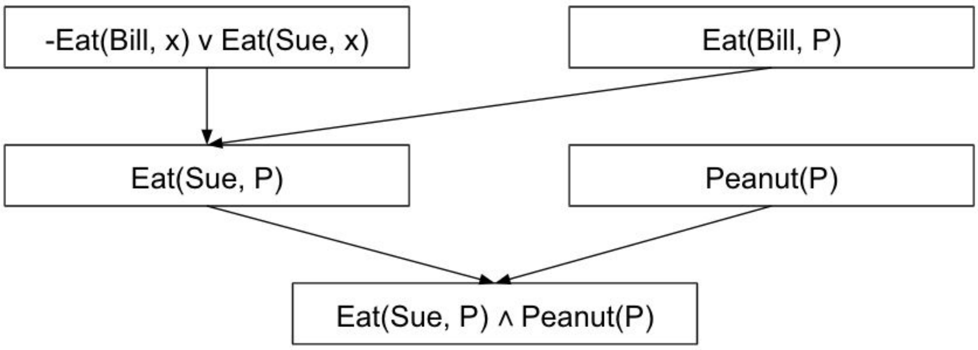
A x, Eat(Bill, x) => Eat(Sue, x)
-Eat(Bill, x) v Eat(Sue, x)

(c) The reverse of "John likes peanuts" is **A x, Peanut(x) => -Like(John, x)**, which can be converted as **-Peanut(x) V -Like(John, x)**



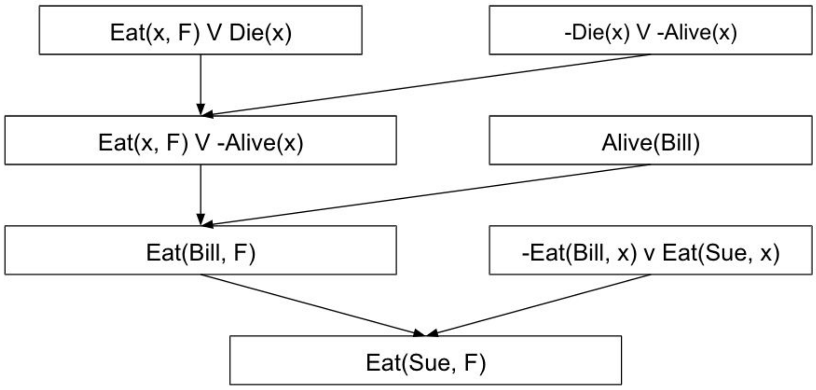
From the result, we can prove that John likes peanuts.

(d) Sue eats Peanuts



(e) We can only know Sue eats some food.

Original	If you don’t eat, you die.	If you die, you are not alive.	Bill is alive.
first-order logic	A x, (E y, -Eat(x, y)) => Die(x)	A x, Die(x) => -Alive(x)	Alive(Bill)
CNF	Eat(x, F) V Die(x)	-Die(x) V -Alive(x)	Alive(Bill)



Question 4.

mythical is true if the unicorn is mythical.
mortal is true if the unicorn is mortal.
mammal is true if the unicorn is a mammal.
horned is true if the unicorn is horned.
magical is true if the unicorn is magical.

(a) (mythical => -mortal) ∧ (-mythical => (mortal ∧ mammal)) ∧ ((-moral V mammal) => horned) ∧ (horned => magical)

(b) (-mythical V -mortal) ∧ (mythical V mortal) ∧ (mythical V mammal) ∧ (mortal V horned) ∧ (-mammal V horned) ∧ (-horned V magical)

(c) **mythical**

1. -mythical
2. -mythical V -mortal
3. mythical V mortal
4. mythical V mammal
5. mortal V horned
6. -mammal V horned
7. -horned V magical
8. mortal [1, 3]
9. -mythical [2, 8]
10. mammal [4, 9]
11. horned [6, 10]
12. magical [7, 11]

=> Can not find NIL
=> Can not prove the unicorn is mythical.

magical

1. -magical
2. -mythical V -mortal
3. mythical V mortal
4. mythical V mammal
5. mortal V horned
6. -mammal V horned

7. -horned V magical
8. -mortal V mammal [2, 4]
9. horned V mammal [5, 8]
10. horned [6, 9]
11. magical [7, 10]
12. unsatisfiable [1, 11]

=> The unicorn is magical.

<u>horned</u>

1. -horned
2. -mythical V -mortal
3. mythical V mortal
4. mythical V mammal
5. mortal V horned
6. -mammal V horned
7. -horned V magical
8. -mythical V horned [2, 5]
9. mammal V horned [4, 8]
10. horned [6, 9]
11. unsatisfiable [1, 10]

=> The unicorn is horned.

Question 5.

(a) => direction:

if α is valid, then M(α) = U, which is the universe of models. M(True) ⊆ U = M(α).

<= direction:

if True ⊨ α, then M(True) ⊆M(α). Since M(True) = U, if U ⊆ M(α), means that M(α) is also U. So α is valid.

(b) M(False) = ∅, no matter what α is, ∅ ⊆ M(α).

(c) => direction:

if α ⊨ β, means that M(α) ⊆ M(β). α => β is equivalent to -α V β. For models in M(β), the β in -α V β can guarantee the results to be True. And for models outside M(β), it's also models outside M(α), -α can guarantee the results to be True. So α => β will be valid.

<= direction:

If α => β is valid, -α V β is always true, which means (α, β) can be (True, True), (False, True), (False, False). For all the cases, M(α) ⊆ M(β), means α ⊨ β.

(d) => direction:

if α ⊨ β, means that M(α) ⊆ M(β). For models in M(α), they are also in M(β), so -β in α ∧ -β will let the results be False. And for models outside M(α), α in α ∧ -β will let the results be False. So the sentence will always be False, which means unsatisfiable.

<= direction:

If α ∧ -β is unsatisfiable, means (α, β) can be (True, True), (False, True), (False, False). For all the cases, M(α) ⊆ M(β), means α ⊨ β.