Patrick Tan
204158646
Disscussion 1C

PROBLEM 1

A. P(A, B, B), P(x, y, z)

{x/A, y/B, z/B}

B. Q(y, G(A, B)), Q(G(x, x), y)

No unifier exists. y must bind to G(A,B) and x cannot bind to both A and B.

C. R(x,A,z), R(B,y,z)

{x/B, y/A} is the most general since z can bind to anything and be equivalent

D. Older (Father(y), y), Older (Father(x), John)

{x/John, y/John}

E. Knows(Father(y),y), Knows(x,x)

No unifier exists. x cannot bind to both Father(y) and y.

PROBLEM 2

A.

John likes all kinds of food.
1. Ax Food(x) => Like(John, x)

Apples are food.
2. Food(Apples)

Chicken is food.
3. Food(Chicken)

Anything anyone eats and isn't killed by is food.
4. Ex Ay Eats(x,y) ^ ~Killed(x,y) => Food(y)

If you are killed by something, you are not alive.
5. Ex Ay Killed(x,y) =>~Alive(x)

Bill eats peanuts and is still alive.*
6. Eat(Bill, Peanuts) & Alive(Bill)

Sue eats everything Bill eats.
7. Ax Eat(Bill,x) => Eat(Sue,x)

Patrick Tan
204158646
Disscussion 1C

B.

Ax Food(x) => Like(John, x)
Ax ~Food(x) | Like(John, x)
1. ~Food(x) | Like(John,x)

2. Food(Apples)

3. Food(Chicken)

Ex Ay Eats(x,y) & ~Killed(x,y) => Food(y)
Ex Ay ~(Eats(x,y) & ~Killed(x,y)) | Food(y)
Ex Ay ~Eats(x,y) | Killed(x,y) | Food(y)
Note: Standardize variables. In clause 1, x is the food, so we make x food and y the individual eating the food from here on.
Ax Ey ~Eats(y,x) | Killed(y,x) | Food(x)
Ax ~Eats(F(y), x) | Killed(F(y), x) | Food(x)
4. ~Eats(F(y), x) | Killed(F(y), x) | Food(x)

Ex Ay Killed(x, y) =>~Alive(x)
Ex Ay ~Killed(x,y) | ~Alive(x)
Note: Same as above, standardize variables.
Ax Ey ~Killed(y, x) | ~Alive(y)
Ax ~Killed(F(y), x) | ~Alive(F(y))
5. ~Killed(F(y), x) | ~Alive(F(y))

Eat(Bill, Peanuts) & Alive(Bill)
6. Eat(Bill, Peanuts)
7. Alive(Bill)

Ax Eats(Bill,x) => Eats(Sue,x)
Ax ~Eats(Bill,x) | Eats(Sue,x)
8. ~Eats(Bill,x) | Eats(Sue,x)

C.

Prove that John likes peanuts using resolution.

1. ~Food(x) | Like(John,x)
2. Food(Apples)
3. Food(Chicken)
4. ~Eats(F(y), x) | Killed(F(y), x) | Food(x)
5. ~Killed(F(y), x) | ~Alive(F(y))
6. Eat(Bill, Peanuts)
7. Alive(Bill)
8. ~Eats(Bill,x) | Eats(Sue,x)
9. ~Like(John, Peanuts) Negated assumption

Patrick Tan
204158646
Disscussion 1C

10. ~Food(Peanuts)	1, 9; {x/Peanuts}	
11. ~Eats(F(y), Peanuts) | Killed(F(y), Peanuts)		4, 10; {x/Peanuts}
12. Killed(Bill, Peanuts)	6, 11; {F(y)/Bill}	
13. ~Alive(Bill)	5, 12; {F(y)/Bill, x/Peanuts}	
14. Contradiction	7, 13	

Therefore, we prove that John likes peanuts using resolution since there is a contradiction.

D.

Ex Food(x) ^ Eats(Sue,x)
~(Ex Food(x) ^ Eats(Sue,x))
Ax ~(Food(x) ^ Eats(Sue,x))
Ax ~Food(x) | ~Eats(Sue,x)
~Food(x) | ~Eats(Sue,x)

1. ~Food(x) | Like(John,x)		
2. Food(Apples)		
3. Food(Chicken)		
4. ~Eats(F(y), x) | Killed(F(y), x) | Food(x)		
5. ~Killed(F(y), x) | ~Alive(F(y))		
6. Eat(Bill, Peanuts)		
7. Alive(Bill)		
8. ~Eats(Bill,x) | Eats(Sue,x)		
9. ~Food(x) | ~Eats(Sue,x)	Negated Assumption	
10. ~Eats(Bill,x) | ~Food(x)	8,9	
11. ~Food(Peanuts)	6,10; {x/Peanuts}	
12. ~Eats(F(y), Peanuts) | Killed(F(y), Peanuts)		4,11; {x/Peanuts}
13. Killed(Bill,Peanuts)	6,12; {F(y)/Bill}	
14. ~Alive(Bill)	5,13; {F(y)/Bill, x/Peanuts}	
15. Contradiction	7,14	

Therefore, we prove that Sue does not eat all foods, but we know that she eats some foods, including at least peanuts.

E.

If you don't eat, you die.
Ax Ey ~Eat(y,x) => Die(y)
Ax Ey Eat(y,x) | Die(y)
Ax Eat(F(y), x) | Die(F(y))
Eat(F(y),x) | Die(F(y))

If you die, you are not alive.
Ey Die(y) => ~Alive(y)
Ey ~Die(y) | ~Alive(y)
~Die(F(y)) | ~Alive(F(y))

Patrick Tan
204158646
Disscussion 1C

Bill is alive.
Alive(Bill)

1. ~Food(x) | Like(John,x)
2. Food(Apples)
3. Food(Chicken)
4. ~Eats(F(y), x) | Killed(F(y), x) | Food(x)
5. ~Killed(F(y), x) | ~Alive(F(y))
6. Eat(F(y),x) | Die(F(y))
7. ~Die(F(y)) | ~Alive(F(y))
8. Alive(Bill)
9. ~Eats(Bill,x) | Eats(Sue,x)
10. ~Food(x) | ~Eats(Sue,x)   Same negation as part D
11. ~Eats(Bill,x) | ~Food(x)   9, 10
12. Die(Bill) | ~Food(x)   6, 11; {F(y)/Bill}
13. ~Alive(Bill) | ~Food(x)   7, 12; {F(y)/Bill}
14. ~Food(x)   8, 13; {F(y)/Bill)
15. ~Eats(Bill, x) | Killed(Bill, x)   4, 14; {F(y)/Bill}
16. ~Eats(Bill, x) | ~Alive(Bill)   5, 15; {F(y)/Bill}
17. ~Eats(Bill, x)   8, 16; {F(y)/Bill}
18. Die(Bill)   6, 17; {F(y)/Bill}
19. ~Alive(Bill)   7, 18; {F(y)/Bill}
20. Contradiction   8, 20

By resolution, we know that Sue eats something that is food, but we do not have enough information to specify what kind of food she eats.