Predicate Logic

(SE 212 Tutorial 5)

Amin Bandali[1]

Wed Oct 9, 2019

[1] bandali@uwaterloo.ca

Today's plan

- do some semantics questions from homework 4
- do some ND questions from homework 5

Today's plan

- do some semantics questions from homework 4
- do some ND questions from homework 5

h04q05

Provide a counterexample to show that the following argument is not valid and demonstrate that your answer is correct.

```
forall y : M . exists x : N . p(g(x), y)
|=
exists z : M . p(z, z)
```

```
Domain:
    M = {m1, m2}
    N = {n1, n2}

Mapping:
    Syntax   | Meaning
    --------------------------
      g(.)   |  G(n1)  := m1
             |  G(n2)  := m2
    --------------------------
    p(., .)  | P(m1, m1)  := F
             | P(m1, m2)  := T
             | P(m2, m1)  := T
             | P(m2, m2)  := F
```

```
Premise:
        [forall y : M . exists x : N . p(g(x), y)]
     = [exists x: N. p(g(x), ^m1)] AND
       [exists x: N . p(g(x), ^m2)]
     = (P(G(n1), m1) OR P(G(n2), m1)) AND
       (P(G(n1), m2) OR P(G(n2), m2))
     = (P(m1, m1) OR P(m2, m1)) AND
       (P(m1, m2) OR P(m2, m2))
     = (F OR T) AND (T OR F)
     = T
```

h04q05 (cont'd)

```
Conclusion:
        [exists z: M . p(z, z)]
     = P(m1, m1) OR P(m2, m2)
     = F OR F
     = F
```

h04q06

Express the following sentences in predicate logic. Use types in your formalization. Is the set of formulas consistent? Demonstrate that your answer is correct using the semantics of predicate logic.

```
All programmer like some computers.
Some programmers use MAC.
Therefore, some people who like some computers use MAC.
```

h04q06 (cont'd)

All programmer like some computers.
Some programmers use MAC.
Therefore, some people who like some computers use MAC.

```
Formalization:
    programmer(x) means x is a programmer
    usesmac(x) means x uses MAC
    likes(x, y) means x likes y

forall x: Person . programmer(x) =>
           exists y: Computer . likes(x, y),
exists x: Person . programmer(x) & usesmac(x)
|-
exists x: Person .
  (exists y: Computer . likes(x, y) & usesmac(x))
```

h04q06 (cont'd)

These sentences are *consistent*. Here is an interpretation in which all the formulas are T:

```
Domain:
    People = {John}
    Computer = {MacPro}

Mapping:
    Syntax          | Meaning
    -------------------------------------------
    programmer(.)   | programmer(John) = T
    likes(.,.)      | likes(John, MacPro) = T
    usesmac(.)      | usesmac(John) = T
```

h04q06 (cont'd)

```
formula 1:
      [forall x: Person . programmer(x) =>
       exists y: Computer . likes(x, y)]
    = [programmer(^John) =>
       exists y: Computer . likes(^John, y)]]
    = programmer(John) IMP likes(John, MacPro)
    = T IMP T
    = T

formula 2:
      [exists x: Person . programmer(x) & usesmac(x)]
    = programmer(John) AND usesmac(John)
    = T AND T
    = T
```

```
formula 3:
      [exists x: Person . (exists y: Computer .
       likes(x, y) & usesmac(x))]
    = [exists y: Computer .
       likes(^John, y) & usesmac(^John)]
    = likes(John, MacPro) AND usesmac(John)
    = T AND T
    = T
```

h05q01a

If the following arguments are valid, use natural deduction AND semantic tableaux to prove them; otherwise, provide a counterexample.

```
forall x . s(x) | t(x),
forall x . s(x) => t(x) & k(c, x),
forall x . t(x) => m(x)
|-
m(c)
where c is a constant
```

h05q01a (cont'd)

```
#check ND
forall x . s(x) | t(x),
forall x . s(x) => t(x) & k(c, x),
forall x . t(x) => m(x)
|-
m(c)
```

```
1) forall x . s(x) | t(x) premise
2) forall x . s(x) => t(x) & k(c, x) premise
3) forall x . t(x) => m(x) premise
4) s(c) | t(c) by forall_e on 1
5) s(c) => t(c) & k(c, c) by forall_e on 2
6) t(c) => m(c) by forall_e on 3
7) case s(c) {
    8) t(c) & k(c, c) by imp_e on 5, 7
    9) t(c) by and_e on 8
    10) m(c) by imp_e on 6, 9
}
11) case t(c) {
    12) m(c) by imp_e on 6, 11
}
13) m(c) by cases on 4, 7-10, 11-12
```

h05q01b

Is this formula a tautology?

```
|- (exists x . p(x)) => forall y . p(y)
```

No, this formula is not a tautology. Interpretation:

```
1) Domain = {a, b}

2) Mapping:
    Syntax   | Meaning
    ----------------------
    p(.)     | P(a) = T
             | P(b) = F

Conclusion:
  [(exists x. p(x)) => forall y. p(y)]
= (P(a) OR P(b)) IMP (P(a) AND P(b))
= (T OR F) IMP (T AND F)
= T IMP F
= F
```

h05q01d

Is this argument valid?

```
forall x . p(x) | q(x),
forall x . !p(x)
|-
forall x . q(x)
```

```
#check ND

forall x . p(x) | q(x), forall x . !p(x) |- forall x . q(x)

1) forall x . p(x) | q(x) premise
2) forall x . !p(x) premise
3) for every xg {
    4) p(xg) | q(xg) by forall_e on 1
    5) case p(xg) {
        6) !p(xg) by forall_e on 2
        7) q(xg) by not_e on 5, 6
    }
    8) case q(xg) {}
    9) q(xg) by cases on 4, 5-7, 8-8
}
10) forall x. q(x) by forall_i on 3-9
```

Announcements

- no tutorial next week (Oct 16) (reading week)
- no tutorial the week after (Oct 23) (midterm marking)