Theoretical homework #4, TTTV 2017

By: Deborah Lambregts (11318643) & Bram Otten (10992456)
Group: G
TA: Douwe van der Wal

Date: May 7, 2017

Exercise 1

(a) Angus likes somebody and somebody likes Betty
$\exists x \exists y[Likes(a, Person(x)) \wedge Likes(Person(y), b)]$

(b) Angus loves a dog who loves him
$\exists x[Loves(a, Dog(x)) \wedge Loves(Dog(x), a)]$

(c) Nobody greets Carl
1) $\neg \exists x[Greets(Person(x), c)]$
2) $\forall x[\neg\ Greets(Person(x), c)]$

(d) Somebody coughs and sneezes
$\exists x[Person(x) \wedge Coughs(x) \wedge Sneezes(x)]$

(e) Nobody coughs or sneezes
1) $\neg \exists x[Person(x) \wedge Coughs(x) \wedge Sneezes(x)]$
2) $\forall x[Person(x) \rightarrow \neg\ Coughs(x) \wedge \neg\ Sneezes(x)]$

(f) Only one dog barks
$\exists x[Dog(x) \wedge Barks(x)] \wedge \neg \exists y[Dog(y) \wedge y \neq x \wedge Barks(y)]$

Exercise 2

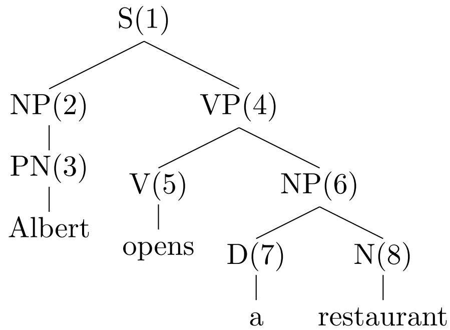
$Restaurant(md)$
$ServesVeg(md)$
$ServesMeat(md)$
$\forall x[Restaurant(x) \wedge \neg ServesMeat(x) \rightarrow Vegetarian(x)]$
$\forall x.\forall y[Person(x) \wedge Restaurant(y) \wedge ServesVeg(y) \rightarrow CanEatAt(x, y)]$

Exercise 3

(a) Sem(S) = Sem(NP) @ Sem(VP)
Sem(NP) = Sem(D) @ Sem(N)
Sem(NP) = Sem(PN)
Sem(VP) = Sem(V) @ Sem(NP)

D → a: $\lambda x.[\lambda y.[\exists z.[x@z \wedge y@z]]]$
D → every: $\lambda x.[\lambda y.[\forall z.[x@z \wedge y@z]]]$
N → restaurant: $\lambda x.Restaurant(x)$
N → menu: $\lambda x.Menu(x)$
PN → Albert: $\lambda x.[x@a]$
V → has: $\lambda x.[\lambda y.[x@\lambda y.Has(x, y)]]$
V → opens: $\lambda x.[\lambda y.[x@\lambda y.Opens(x, y)]]$

(b) (i) $\exists x(Opens(a, x) \wedge Restaurant(x))$



Before β-conversion

(1) $\lambda x.[x@a]$ @ $\lambda x.[\lambda y.[x@\lambda y.Opens(x, y)]]$ @ $\lambda x.[\lambda y.[\exists z.[x@y \wedge y@z]]]$ @ $\lambda x.Restaurant(x)$

(2) $\lambda x.[x@a]$

(3) $\lambda x.[x@a]$

(4) $\lambda x.[\lambda y.[x@\lambda y.Opens(x, y)]]$ @ $\lambda x.[\lambda y.[\exists z.[x@y \wedge y@z]]]$ @ $\lambda x.Restaurant(x)$

(5) $\lambda x.[\lambda y.[x@\lambda y.Opens(x, y)]]$

(6) $\lambda x.[\lambda y.[\exists z.[x@y \wedge y@z]]]$ @ $\lambda x.Restaurant(x)$

(7) $\lambda x.[\lambda y.[\exists z.[x@y \wedge y@z]]]$

(8) $\lambda x.Restaurant(x)$

After β-conversion

(1) $\lambda x.[x@a]@\lambda y.Opens(x, y).\exists z.(\lambda x.Restaurant(x)@y \wedge y@z)$
$\rightsquigarrow \lambda y.Opens(x, y).\exists z.(\lambda x.Restaurant(x)@y \wedge y@z)@a$
$\rightsquigarrow Opens(x, a).\exists z.(\lambda x.Restaurant(x)@a \wedge a@z)$

(2) x

(3) x

(4) $\lambda x.[\lambda y.[x@\lambda y.Opens(x, y)]]@\lambda y.\exists z.(\lambda x.Restaurant(x)@y \wedge y@z)$
$\rightsquigarrow \lambda y(\lambda y.\exists z.(\lambda x.Restaurant(x)@y \wedge y@z)@\lambda y.Opens(x, y)$
$\rightsquigarrow \lambda y.Opens(x, y).\exists z.(\lambda x.Restaurant(x)@y \wedge y@z)$

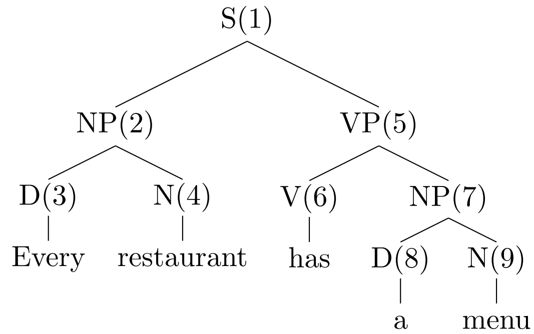
(5) x

(6) $\lambda x.[\lambda y.[\exists z.[x@y \wedge y@z]]]$ @ $\lambda x.Restaurant(x)$
$\rightsquigarrow \lambda y.\exists z.(\lambda x.Restaurant(x)@y \wedge y@z)$

(7) x

(8) x

(ii) $\forall x \exists y (Restaurant(x) \rightarrow Menu(y) \wedge Has(x, y)$



β-conversion

(1)

(2)

(3) $\lambda x.[\lambda y.[\forall z.[x@y \wedge y@z]]]$

(4) $\lambda x.Restaurant(x)$

(5) $\lambda x.[\lambda y.[x@\lambda y.Has(x, y)]]@\lambda y.\exists z.(\lambda x.Menu(x))@y \wedge y@z)$
$\rightsquigarrow$

(6) $\lambda x.[\lambda y.[x@\lambda y.Has(x, y)]]$

(7) $\lambda x.[\lambda y.[\exists z.[x@y \wedge y@z]]]@\lambda x.Menu(x)$
$\rightsquigarrow \lambda y.\exists z.(\lambda x.Menu(x))@y \wedge y@z)$

(8) $\lambda x.[\lambda y.[\exists z.[x@y \wedge y@z]]]$

(9) $\lambda x.Menu(x)$

Exercise 4

(a) The scope of the sentence is ambiguous: either restaurants can have unique menus but must have one, or all restaurant have the same menu.

(b) (i) $\forall x \exists y (Restaurant(x) \rightarrow Menu(y) \land Has(x, y))$

(ii) $\exists x \forall y \neg \exists z ((Menu(x) \land Restaurant(y) \rightarrow Has(y, x)) \land \neg(Menu(z) \land Restaurant(y) \land x \neq z \rightarrow Has(y, z)))$

(c) This is not possible, because the only rule for 'a' is

```
Det[SEM=<\P \Q.exists x.(P(x) & Q(x))>] -> 'a'
```

which has the meaning of 'a' used in logic formula (i).