Computational Linguistics: Semantics

Raffaella Bernardi

KRDB, Free University of Bozen-Bolzano

P.zza Domenicani, Room: 2.28, e-mail: bernardi@inf.unibz.it

Contents

1. Exercise 1: Well formed formula

Let j be a constant of type e; M of type $e \to t$; S of type $((e \to t) \to (e \to t))$, and P of type $(e \to t) \to t$. Furthermore, x is a variable of type e, and Y a variable of type $(e \to t)$.

Determine which of the following is well-formed, give its type.

1. $(\lambda x.M(x))(P)$.
2. $(\lambda x.M(x))(j)$.
3. $\lambda x.M(j)$.
4. $S(\lambda x.M(x))$.
5. $(\lambda Y.Y(j))(M)$
6. $\lambda x.(M(x) \wedge M(j))$
7. $(\lambda x.M(x)) \wedge M(j))$

Solution

1. no (since the function is of type $e \longrightarrow t$ while the argument is of type $(e \longrightarrow t) \longrightarrow t$ (whereas it should be of type e.
2. yes, t
3. yes, $e \longrightarrow t$
4. yes, $e \longrightarrow t$
5. yes, t
6. yes, $e \longrightarrow t$
7. no. ($\wedge$ must connect expressions of type t)

2. Exercise 2: λ-conversion

Let j be a constant of type e; M of type $(e \rightarrow t)$, and A of type $e \rightarrow (e \rightarrow t)$. Furthermore, x and y are variables of type e, and Y is a variable of type $e \rightarrow t$. Reduce the following expression as much as possible by means of λ-conversion.

1. $\lambda x(M(x))(j)$
2. $\lambda Y(Y(j))(M)$
3. $\lambda x\lambda Y(Y(x))(j)(M)$
4. $\lambda x\forall y(A(x)(y))(j)$
5. $\lambda x\forall y(A(x)(y))(y)$
6. $\lambda Y(Y(j))\lambda x(M(x))$
7. $\lambda Y\forall x(Y(x))\lambda y(A(x)(y))$

Solution:

1. $M(j)$
2. $M(j)$
3. $M(j)$
4. $\forall y A(j)(y)$
5. $\forall z.A(y)(z)$
6. $M(j)$ (by replacing first Y with $\lambda x(M(x))$ and then x with j.
7. $\forall z.A(z)(x)$

Note, in 5 and 7 you have to rename variables. Direct λ-conversion is not possible: in 5. y is not free for x in $\forall y.(A(x)(y))$. Hence you have to rename the y by, e.g., z, so to be able λ-conversion. Similarly, in 7, you can rename x by z before apply λ-conversion.

3. Exercise 3: λ-calculus and NL

Given,

- new $\lambda Y_{e\to t}.\lambda x_e.(Y(x) \wedge new(x))_t : adj$
- book $\lambda x_e.(book(x))_t : n$
- student $\lambda x_e.student(x))_t : n$
- a $\lambda X_{(e\to t)}\lambda Y_{(e\to t)}(\exists x_e.X(x) \wedge Y(x)) : det$
- john $j : np$
- read $\lambda x_e.\lambda y_e.read(y, x) : tv$
- left $\lambda y_e.left(y) : iv$

build the meaning representation and the parse tree for

1. John read a book

2. A new student left

3. John read a new book

4. A student read a book

Use the following CFG to build the parse trees.

```
s ---> np vp
vp ---> iv
vp ---> tv np
np ---> det n
n ---> adj n
```

Solution:

1. John read a book

    - read u: $\lambda y.read(y, u)$
    - john read u: $read(j, u)$.

- john read: $\lambda z.read(j, z)$
- a book: $\lambda Y.\exists x.Book(x) \wedge Y(x)$
- john read a book: $\exists x.Book(x) \wedge read(j, x)$

```
         s
       /   \
      /     vp
     /     /   \
    /     /     np
   /     /     /   \
 np     tv   det    n
  |      |    |      |
 john read   a     book
```

2. A new student left

- new student: $\lambda y.Student(y) \wedge new(y)$

- a new student: $\lambda Y.\exists x.(Student(x) \wedge new(x)) \wedge Y(x)$
- a new student left: $\exists x.(Student(x) \wedge new(x)) \wedge left(x)$

```
        s
     /      \
    np       \
   /   \      \
  /     n     vp
 /    /   \    |
det  adj   n    iv
 |    |    |    |
a  new student left
```

3. John read a new book

- new book: $\lambda y.book(y) \wedge new(y)$
- a new student: $\lambda Y.\exists x.(book(x) \wedge new(x)) \wedge Y(x)$

- john read: $\lambda u.read(j, u)$ (as in 1.)
- john read a new book: $\exists x.(book(x) \wedge new(x)) \wedge read(j, x)$

```
          s
         / \
        /   vp
       /   /  \
      /   /   np
     /   /   /    \
    /   /   /      n
   /   /   /     /  \
  np   tv  det  adj  n
  |    |   |     |   |
 john read a    new  book
```

4. A student read a book

- u read a book: $\exists x.Book(x) \wedge Read(u, x)$ (see above)
- read a book: $\lambda u.\exists x.Book(x) \wedge Read(u, x)$
- a student: $\lambda Z.\exists y.Student(x) \wedge Z(u, y)$ (see above)
- a student read a book: $\exists y.Student(y) \wedge \exists x.Book(x) \wedge Read(y, x)$ (which is equivalent to $\exists y.\exists x.Student(y) \wedge (Book(x) \wedge Read(y, x)))$

```
            s
          /        \
         /          vp
        /          /   \
      np          /    np
     /  \        /    /   \
   det   n     tv   det     n
    |    |      |    |      |
   a student read   a     book
```

4. Exercise 3: λ-calculus and NL

You know that e.g.

"Every student left" can be represented as $\forall x.Student(x) \rightarrow Left(x)$; "No student left" as $\neg\exists x.Student(x) \rightarrow Left(x)$, John dind't leave as $\neg leave(j)$. Use them to give the lambda terms for the words below.

1. every
2. everybody
3. no
4. nobody
5. didn't
6. did
7. and
8. or

Solution

1. every: $\lambda X.\lambda Y.\forall z.X(z) \rightarrow Y(z)$
2. everybody: $\lambda Y.\forall z.Y(z)$
3. no: $\lambda X.\lambda Y.\neg\exists z.X(z) \rightarrow Y(z)$
4. nobody: $\lambda Y.\neg\exists z.Y(z)$
5. didn't: $\lambda Y.\neg Y$.
6. did: $\lambda Y.Y$.
7. and: $\lambda X.\lambda Y.Y \wedge X$
8. or: $\lambda X.\lambda Y.Y \vee X$