Semantics

Greg Kobele

June 18, 2018

Types

Basic Types

There are different kinds of things

semantics

propositions t

individuals e

trees T

Function Types

Starting with a finite number of basic types,

We allow for the type of *functions*
if

- α is a type
- β is a type

then

- $(\alpha\beta)$ is a type
- the type of functions from αs to βs

Trees

a tree: $a(b)(c(e)(f))(d)$

is not a function

- it is a basic object

it has type
T

Trees missing trees

a tree with a hole: $\lambda x.a(b)(x)(d)$

is a function
it takes a tree as input, and puts it in the hole

it has type
(TT)

Generalizing trees with holes

A tree with a hole

- is what is left over when you remove a subtree from a tree

It is called a

tree context

- it is the 'context' in which a subtree occurs

A context context

is what is left when you remove a *tree context* from a tree

Trees missing contexts

a context context $\lambda f.a(b)(f(e))(d)$

is also a function
it takes a context as input, and puts it in the hole

It has type
(TT)T

In general

- can play this game forever!

A (context context) context
is a tree which is missing a context context

- i.e. a tree, missing a tree missing a tree missing a tree
- of type ((TT)T)T

In general
$\alpha\beta$

- a β
- which is missing an α

Non-associativity of type formation

T(TT) is different from (TT)T

T(TT)

- takes two arguments to get a tree
- both arguments are trees

(TT)T

- takes one argument to get a tree
- only argument is a *function*

Example

$\lambda x.g(x)(x)$
constructs trees with identical daughters

$\lambda f.g(f(a))(f(b))$
can have ’differences’ between the daughters

- sameness is more abstract

Lambda Terms

Interpretation

$$\lambda x.M$$

1. an M with a hole named x
2. an M which is missing an x
3. a function which takes an argument (here called x), and outputs an M

Typing terms

Variables
a variable can have any type you like

Abstractions

- if M has type β
- and x has type α
- then $\lambda x.M$ has type $(\alpha\beta)$

Applications

- if M has type $(\alpha\beta)$
- and N has type α
- then $M\ N$ has type β

Typing contexts

$$\Gamma \vdash M : \alpha$$

- Γ assumptions about types of variables
- M a λ term
- α the type of M

We can use typing contexts
to help find the type of a term

Typing variables

Variables
a variable can have any type you like

Translation

- x has type α
- if you assume that x has type α

In pictures

$$\frac{}{\Gamma, x : \alpha \vdash x : \alpha}$$

Typing abstractions

Abstractions

- if M has type β
- and x has type α
- then $\lambda x.M$ has type $(\alpha\beta)$

Translation

- $\lambda x.M$ has type $(\alpha\beta)$
- if when you assume that x has type α
- M has type β

In pictures

$$\frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x.M : \alpha\beta}$$

Typing applications

Applications

- if M has type $(\alpha\beta)$
- and N has type α
- then $M\ N$ has type β

Translation

- $M\ N$ has type β
- if M has type $(\alpha\beta)$
- and N has type α

In pictures

$$\frac{\Gamma \vdash M : \alpha\beta \qquad \Gamma \vdash N : \alpha}{\Gamma \vdash MN : \beta}$$

Example

assume that

constant	type
a,c	TTT
b,d,e,f	T

- $a(b)(c(e)(f))$
- $\lambda x.a(b)(x)$
- $\lambda g.a(b)(g(e))$

Typed terms

Another way of writing λ terms includes typing information in the term (just write variables together with their types):

- $a(b)(c(e)(f))$
- $\lambda x^T.a(b)(x^T)$
- $\lambda g^{TT}.a(b)(g^{TT}(e))$

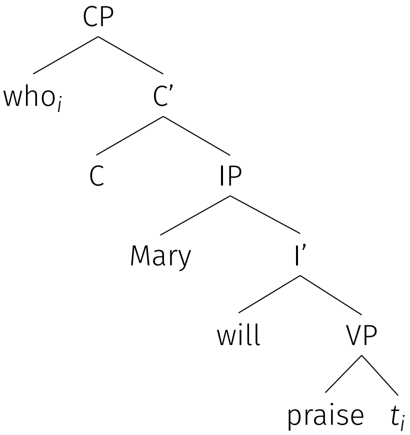
We can then drop the typing contexts

$$\frac{}{x^\alpha : \alpha} \qquad \frac{M : \beta}{\lambda x^\alpha.M : \alpha\beta}$$

$$\frac{M : \alpha\beta \qquad N : \alpha}{(M\ N) : \beta}$$

Interpreting Movement

Movement and traces



Why is there a trace in the VP?

1. because something moved out
2. because the VP is missing its object
3. because the VP has a hole

Interpreting traces

The basic idea

syntax	t_i
semantics	x_i

A slogan

traces are holes

Interpretation rules

landing sites ’I moved, and left behind a hole’

$$\left[\!\!\left[\begin{array}{ccc} & \bullet & \\ \diagup & & \diagdown \\ \alpha_i & & \beta \end{array} \right]\!\!\right] = [\![\alpha]\!]\,(\lambda x_i.\,[\![\beta]\!])$$

traces ’I am a hole’

$$[\![t_i]\!] = x_i$$

The types of traces

landing sites 'I moved, and left behind a hole'

$$\left[\!\!\left[\;[_{\bullet}\ \alpha_i \ \ \beta\,]\; \right]\!\!\right] = [\![\alpha]\!]\,(\lambda x_i.\,[\![\beta]\!])$$

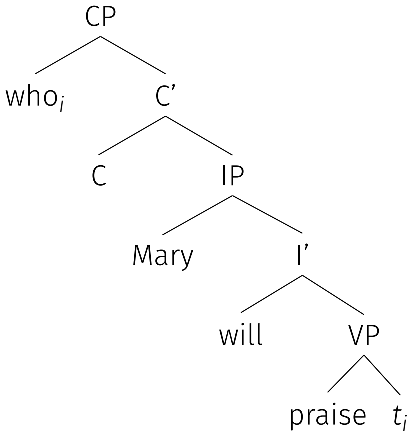
this requires that

- $[\![\alpha]\!]$ has type *(ab)c*
- $[\![\beta]\!]$ has type *b*
- x_i has type *a*

Example
a DP has type *(et)t*

- its trace must have type *e*

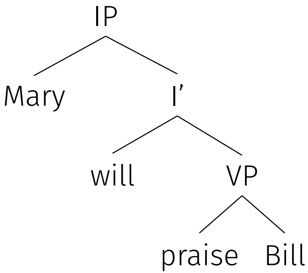
Example



- what should the interpretation look like?
- what should the ’truth conditions’ be?

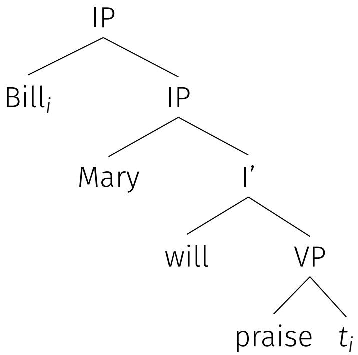
Transitivity

Revisiting a previous example



- how could we interpret this at all?

answer:
if the structure were different



Where should objects move?

answer:
at least to a place where the interpretation of the sister is of type t

VP internal subjects