Language and Inference

Day 3: Building Meaning Representations

Johan Bos
johan.bos@rug.nl
university of groningen

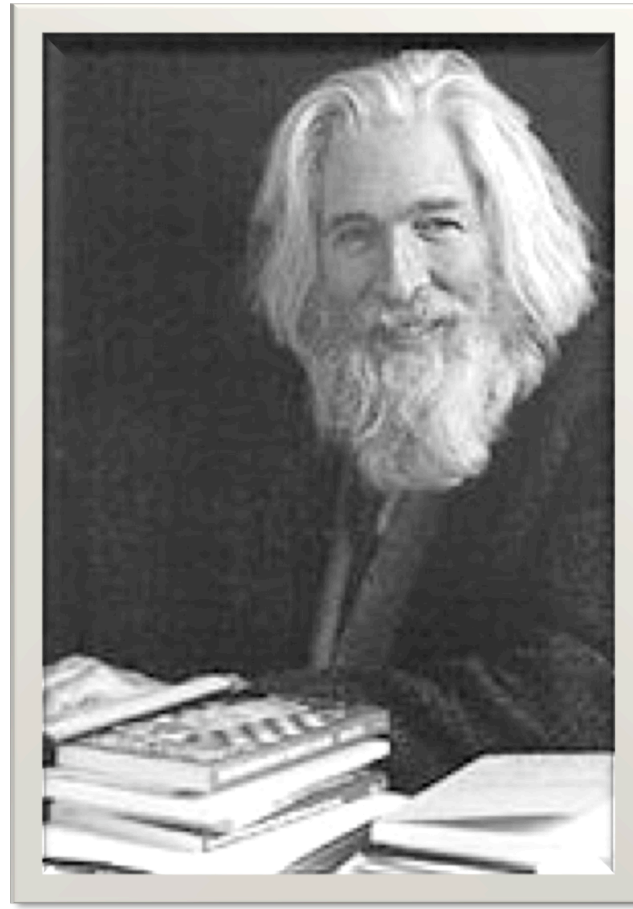
Today

- Introduce a method to build meaning representations from English text
- Use the grammar formalism introduced yesterday
- Specify the syntax-semantics interface

Bluebird

Starling

Thrush



Raymond Smullyan

$$\frac{X/Y \qquad Y}{X} >$$

$$\frac{Y \qquad X\backslash Y}{X} <$$

$$\frac{X/Y \qquad Y/Z}{X/Z} >B$$

$$\frac{Y\backslash Z \qquad X\backslash Y}{X\backslash Z} <B$$

$$\frac{X/Y \qquad Y\backslash Z}{X\backslash Z} >Bx$$

$$\frac{Y/Z \qquad X\backslash Y}{X/Z} <Bx$$

Rule schemata (1)

$$\frac{X}{Y/(Y\backslash X)} >T \qquad \frac{X}{X\backslash(Y/X)} <T$$

$$\frac{X \quad CONJ \quad X}{X} <>$$

$$\frac{(X/Y)/Z \quad Y/Z}{X/Z} >S \qquad \frac{Y/Z \quad (X\backslash Y)/Z}{X/Z} <Sx$$

Rule schemata (2)

- How do we construct DRSs from sentences (or texts) in a systematic way?
- We will let us guide by syntactic structure!
- What we will do is show how we can combine CCG with DRT

Compositional Semantics

- Use techniques from the lambda calculus to combine CCG with DRT
- Every word gets assigned a “partial” DRS
- Each combinatorial rule in CCG has a semantic interpretation consistent with lambda calculus

Combining CCG with DRT

Partial DRSs

We will add a couple of new ingredients:

λ @ ;

- The lambda operator λ signals missing information
- Function application is indicated by @
- The ; operator denotes a merge between two DRSs

CCG+DRT: lexical semantics

Category	Partial DRS	Example
N	λx. [| spokesman(x)]	*spokesman*
NP/N	λp. λq.([x |] ;(p@x);(q@x))	*a*
S\NP	λn.(n@λy. [e | lie(e) agent(e,y)])	*lied*

Type theory

- We will use two basic types:

  e (entity, i.e. discourse referents), and
  t (truth value, i.e. DRSs)

- The set of all types is recursively defined in the usual way:

  if α and β are types, then so is $<\alpha,\beta>$

Syntax of partial DRSs

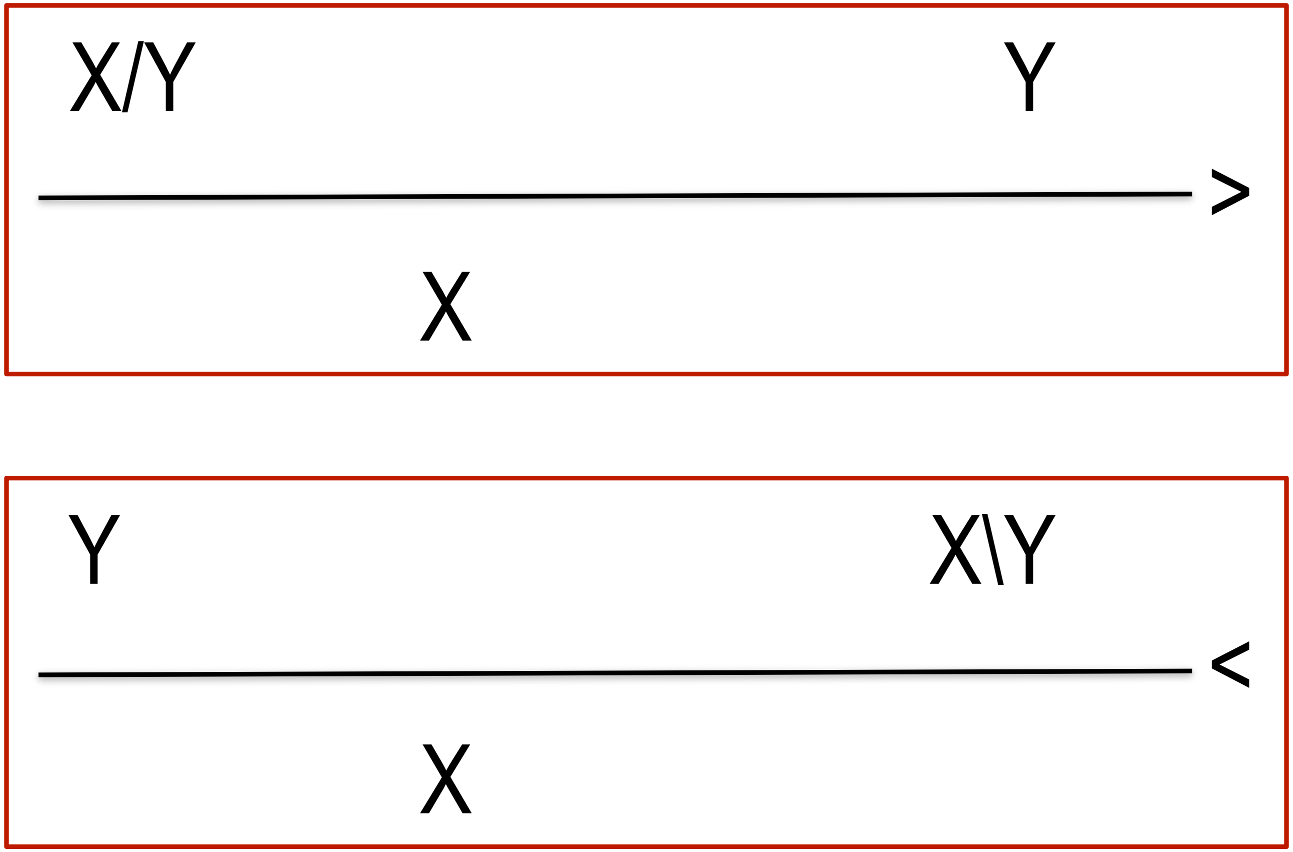
$<EXP_t>$ ::= [$<VAR_e>$* | $<CON>$*] | ($<EXP_t>$;$<EXP_t>$) | ($<EXP_{<\alpha,t>}>$@$<EXP_\alpha>$)

$<CON>$ ::= $<BASIC>$ | $<COMPLEX>$

$<BASIC>$ ::= $<SYM_1>$($<VAR_e>$) | $<SYM_2>$($<VAR_e>$,$<VAR_e>$) | ...

$<COMPLEX>$ ::= ¬$<EXP_t>$ | $<EXP_t>$ => $<EXP_t>$ | $<VAR_e>$: $<DRS_t>$ | ...

$<EXP_{<\alpha,\beta>}>$ ::= $<VAR_{<\alpha,\beta>}>$ | λ$<VAR_\alpha>$.$<EXP_\beta>$ | ($<EXP_{<\gamma,<\alpha,\beta>>}>$@$<EXP_\gamma>$)



Application (> and <)

$$\frac{X/Y:\ \varphi \qquad\qquad Y:\ \psi}{X:\ (\varphi @ \psi)}\ >$$

$$\frac{Y:\ \psi \qquad\qquad X\backslash Y:\ \varphi}{X:\ (\varphi @ \psi)}\ <$$

Application (> and <)

Composition (>B and <B)

$$\frac{X/Y \qquad Y/Z}{X/Z}\ {>}B$$

$$\frac{Y\backslash Z \qquad X\backslash Y}{X\backslash Z}\ {<}B$$

X/Y: φ   Y/Z: ψ

——————————————— >B

X/Z: λx.(φ@(ψ@x))

Y\Z: ψ   X\Y: φ

——————————————— <B

X\Z: λx.(φ@(ψ@x))

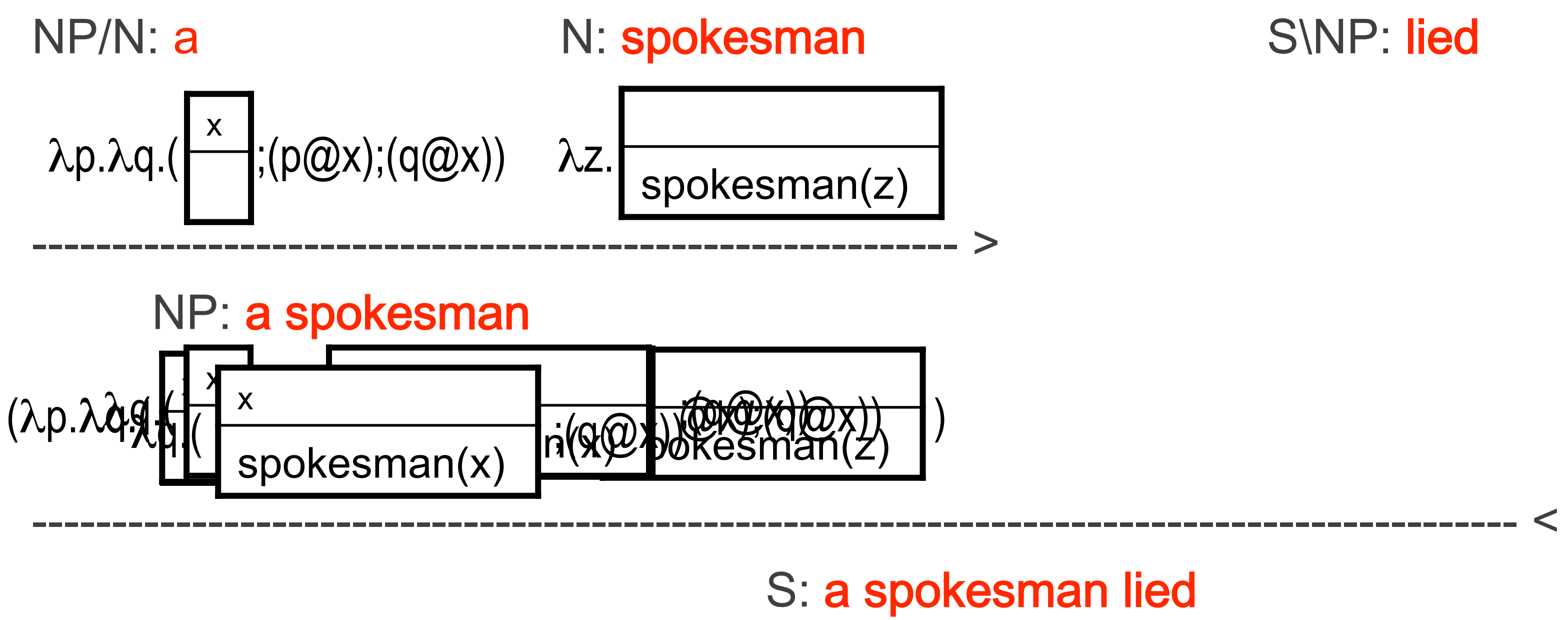
Composition (>B and <B)

- Consider the application: λx.φ@ψ
  - Here the functor is: λx.φ
  - And the argument is: ψ
- The process of replacing every free occurrence of x in φ by ψ is called
  β-conversion
  (or β-reduction, or λ-conversion)

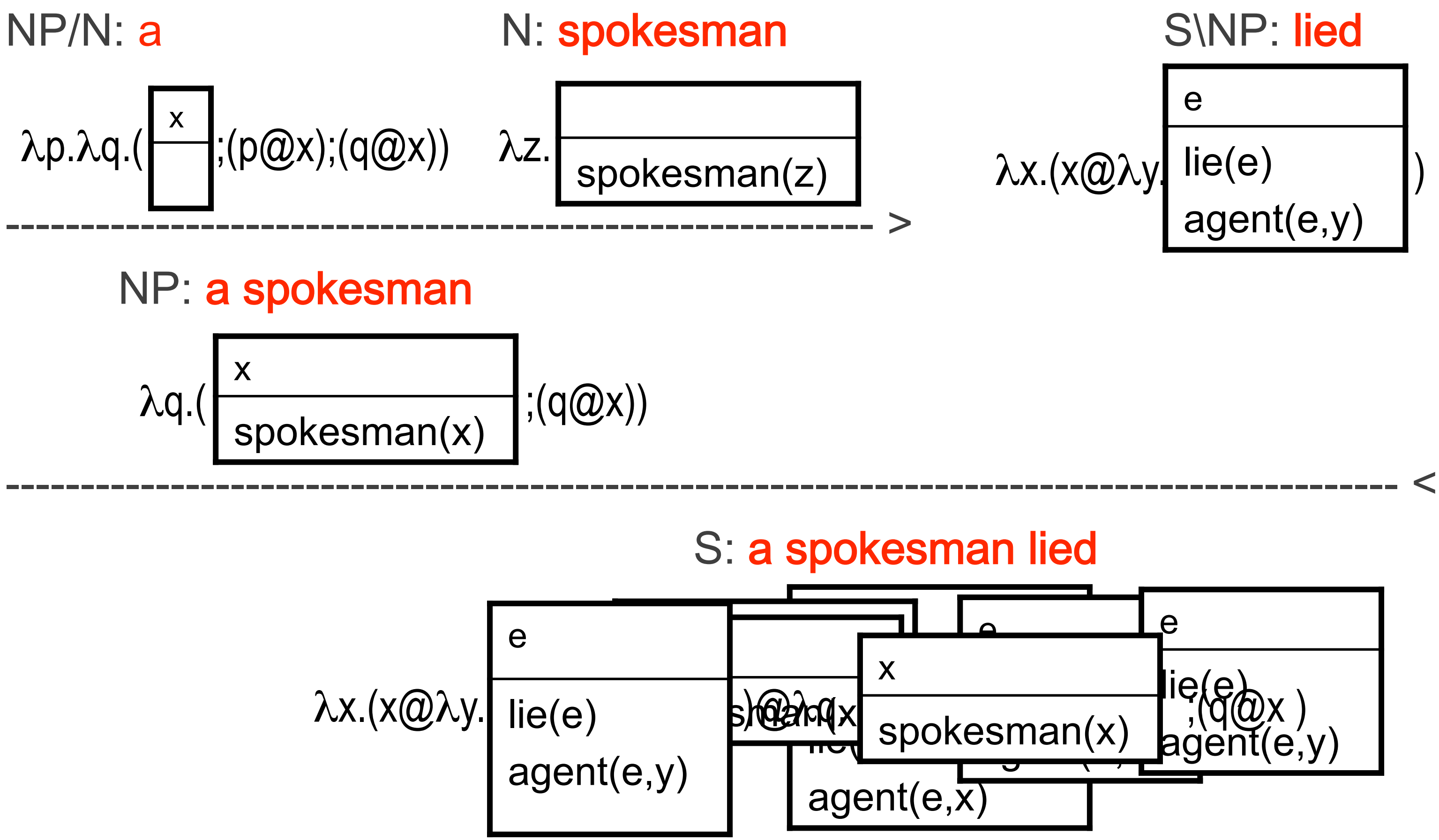
β-conversion

```
NP/N: a     N: spokesman                  S\NP: lied
------------------------------------ >
        NP: a spokesman
        -------------------------------------------------------------- <
                        S: a spokesman lied
```

CCG derivation



CCG+DRT derivation



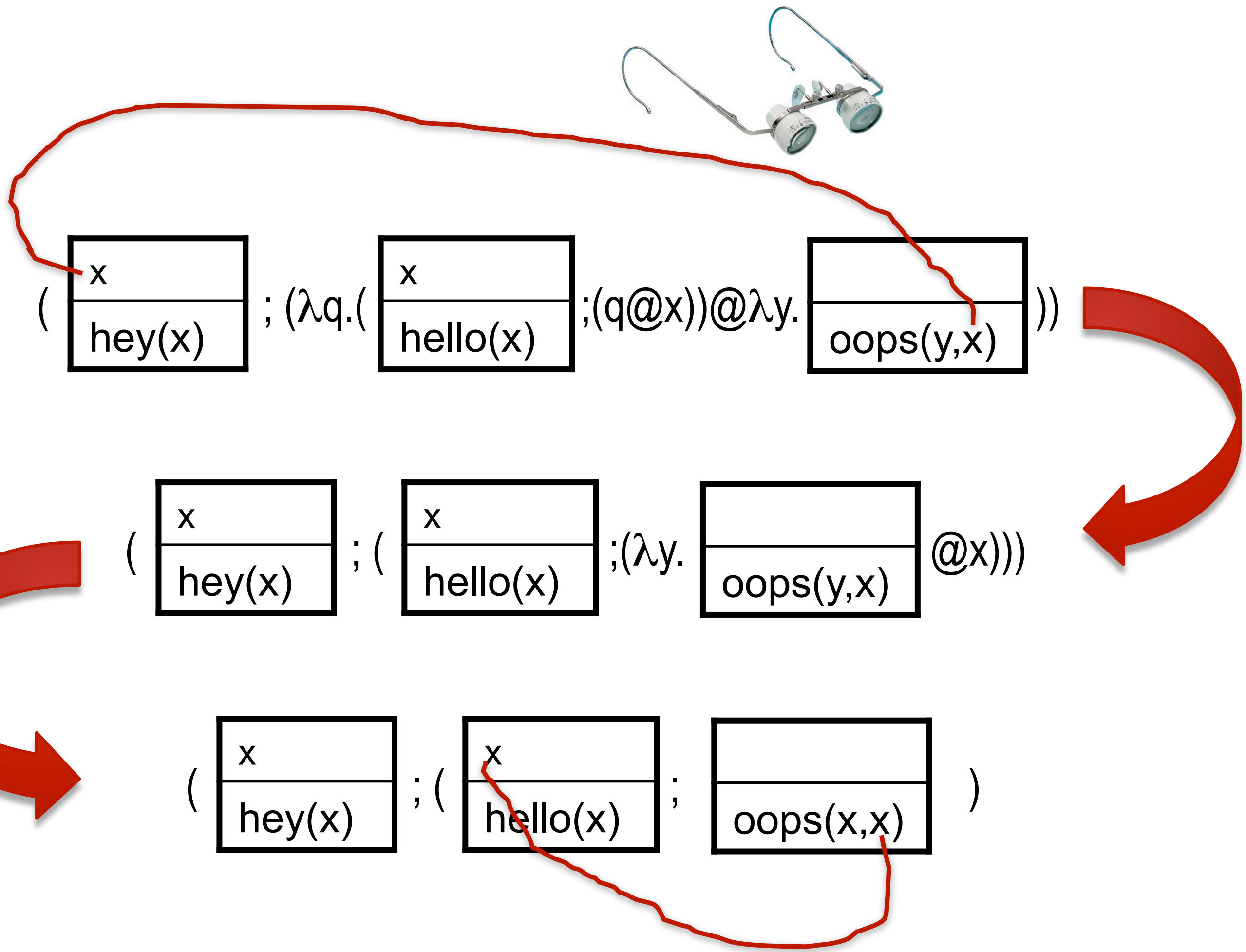
CCG+DRT derivation

What are the semantic types?

- Determine the semantic types of the partial DRSs of the previous example

- Consider again: $\lambda x.\varphi@\psi$
  - The functor is: $\lambda x.\varphi$
  - And the argument is: ψ
- β-conversion can only take place if the set of free variables in ψ is disjoint from the set of bound variables in φ. Why?

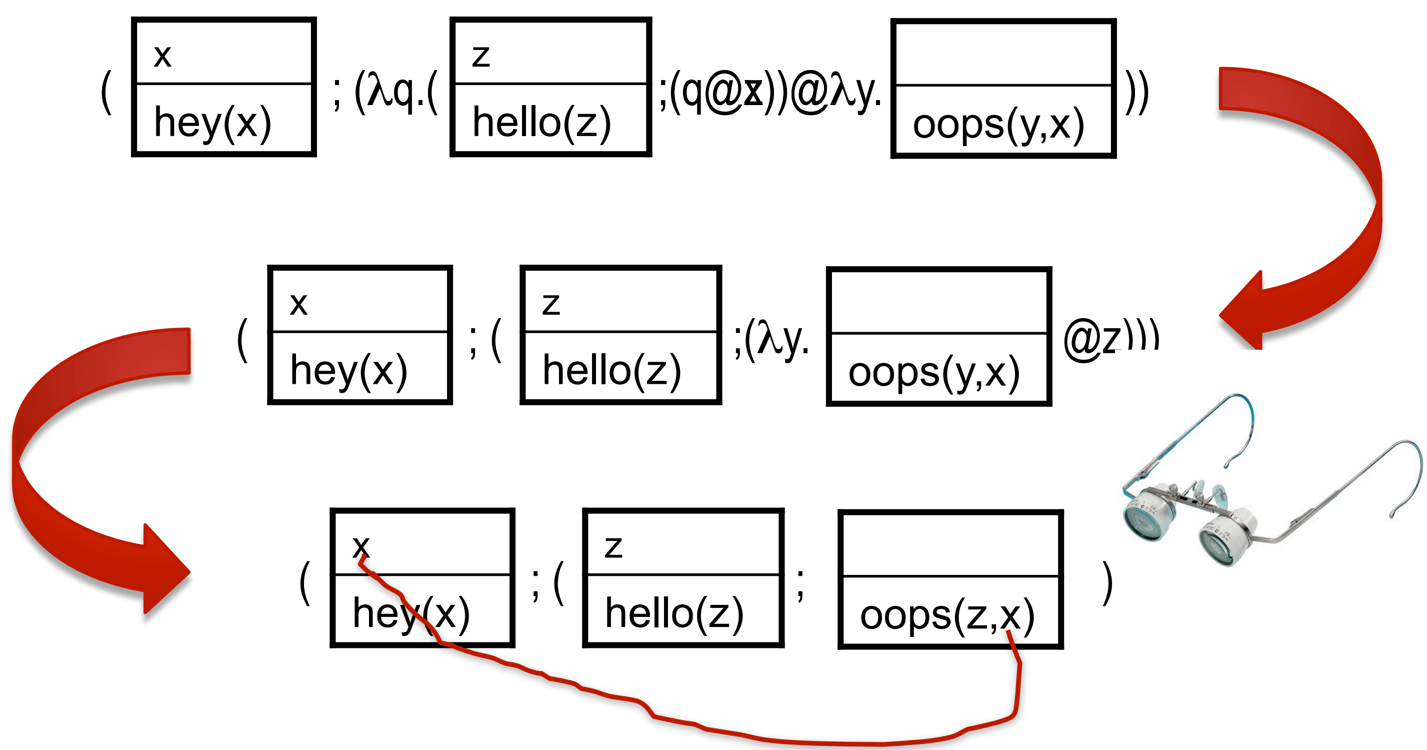
Constraints on β-conversion



Accidentally capturing free variables

- **α-conversion** is the process of replacing bound occurrences of a variable in an expression by a new (unused) variable
- If we do this with the functor for every application before we perform β-conversion, we won't capture free variables anymore

α-conversion



Avoiding capturing free variables

- Theoretical work
  - Associating syntactic categories with a semantic type
  - Follow CCG's principle of type transparency

- Practical work
  - Produce a lexical DRS for each lexical category, obeying type restrictions
  - Lot of work: all lexical categories found in CCGbank

Building the Semantic Lexicon

S	Sentence
NP	Noun Phrase
N	Noun
PP	Prepositional Phrase

Basic Syntactic Categories

e	entity (discourse referent)
t	truth value (box)

Basic Semantic Types

The category N

- Nouns express properties
- Hence it makes sense to associate the category N with the semantic type *<e,t>*
- The semantic type *<e,t>* denotes functions from entities to truth values (properties)

squirrel

$$\text{N: } \langle e,t\rangle\text{: } \lambda\text{x. } \begin{array}{|l|}\hline \\ \hline \text{squirrel(x)} \\ \hline \end{array}$$

red

$$\text{N/N: } \langle\langle e,t\rangle,\langle e,t\rangle\rangle\text{: } \lambda\text{p.}\lambda\text{x.(} \begin{array}{|l|}\hline \\ \hline \text{red(x)} \\ \hline \end{array} \text{;(p@x))}$$

- Prepositional phrases (PPs) also express properties
- Hence it makes sense to associate the category PP with the semantic type *<e,t>* too

The category PP

at a table

$$\text{PP: } \langle e,t \rangle\text{: } \lambda x_1 . \begin{array}{|c|}\hline x_2 \\ \hline \text{table}(x_2) \\ \text{at}(x_1,x_2) \\ \hline \end{array}$$

wife

$$\text{N/PP: } \langle\langle e,t \rangle, \langle e,t \rangle\rangle\text{: } \lambda p . \lambda x_1 . (\begin{array}{|c|}\hline x_2 \\ \hline \text{person}(x_1) \\ \text{wife}(x_2) \\ \text{role}(x_1,x_2) \\ \hline \end{array} ;(p@x_2))$$

The category NP

- Noun phrases denote entities
  - Therefore, the category NP is usually associated with the type *e*
- But we deviate from this approach
  - instead, we give a type-raised analysis to NP
  - The type we give to NP is *<<e,t>,t>*, that is, a function from properties to truth values

someone

$$\text{NP: } \langle\langle e,t\rangle,t\rangle\text{: } \lambda\text{p.}\left(\begin{array}{|c|}\hline \text{x} \\ \hline \text{person(x)} \\ \hline \end{array}\;;(\text{p@x})\right)$$

Examples (NP)

The category S

- Sentence denote truth values
  - Therefore, S would be associated with the semantic type *t*
- But once again we deviate from this view
  - Instead we pair S with the type *<<e,t>,t>*
  - Motivation: compositional neo-Davidsonian semantics

- Approach: Method of Continuation
- Basic ideas:
  - Discourse referents for events get introduced in the lexicon
  - Abstraction over potential modifiers of event discourse referents
  - Each modifier introduce a new abstraction over potential modifiers ("continuation")

Compositional neo-Davidsonian

smoke

$$(\mathrm{S[dcl]}\backslash\mathrm{NP}):\ \langle\langle\langle e,t\rangle,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle:\ \lambda \mathrm{n}_1.\lambda \mathrm{p}_2.(\mathrm{n}_1@\lambda \mathrm{x}_3.(\begin{array}{|c|}\hline \mathrm{e}_4\\ \hline \mathrm{smoke}(\mathrm{e}_4)\\ \mathrm{agent}(\mathrm{e}_4,\mathrm{x}_3)\\ \hline \end{array};(\mathrm{p}_2@\mathrm{e}_4)))$$

Example (S\NP)

(i) $$\lambda p.\left(\begin{array}{|l|}\hline e \\ \hline \mathbf{V}(e) \\ \mathbf{M}_1(e) \\ \hline\end{array};(p@e)\right)$$

(ii) $$\mathbf{(}\lambda v.\lambda p'.(v@\lambda e.\left(\begin{array}{|l|}\hline \\ \hline \mathbf{M}_2(e) \\ \hline\end{array};(p'@e)\right)\ \mathbf{@}\ \lambda p.\left(\begin{array}{|l|}\hline e \\ \hline \mathbf{V}(e) \\ \mathbf{M}_1(e) \\ \hline\end{array};(p@e)\right)\mathbf{)}$$

(iii) $$\lambda p'.\left(\begin{array}{|l|}\hline e \\ \hline \mathbf{V}(e) \\ \mathbf{M}_1(e) \\ \mathbf{M}_2(e) \\ \hline\end{array};(p'@e)\right)$$

Continuation at work

Syntactic Category	Semantic Type
S	<<e,t>,t>
NP	<<e,t>,t>
N	<e,t>
PP	<e,t>

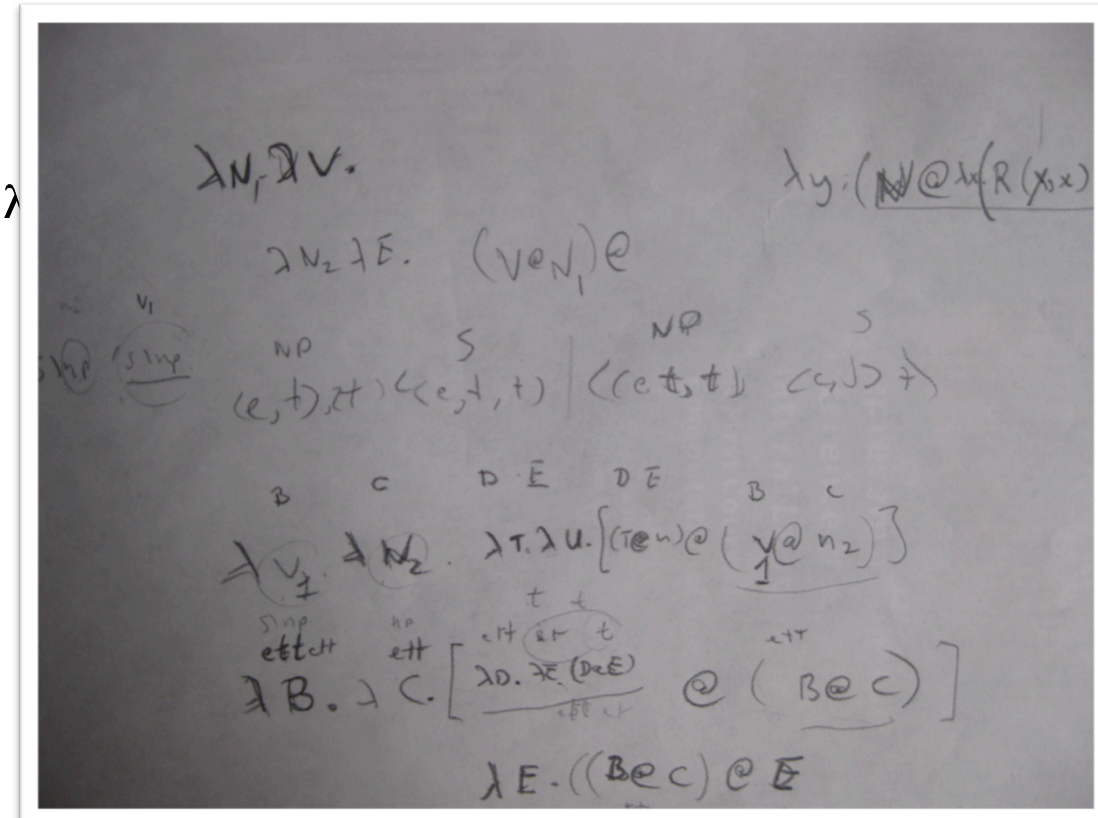
Mapping syntax to semantics

- For each category in the lexicon, we need to provide a fitting partial DRSs
- Manual work, ca. 500 categories
  - Some categories are straightforward
  - Others categories are far from trivial

Producing lexical DRSs

promise: $((S_{dcl}\backslash NP)/(S_{to}\backslash NP))/NP$

$\lambda n_1. \lambda v. \lambda$ [illegible] ;(E@e))



Example Partial DRS (lexicon)

- This is all implemented as the **Boxer** system
  A semantic parser based on CCG and DRT

- The **Groningen Meaning Bank**
  Semantically annotated corpus (CCG + DRT)

Implementation