Semantische Komposition und Satznegation

Frank Richter

Seminar für Sprachwissenschaft

Universität Tübingen
fr@sfs.uni-tuebingen.de

Semantische Grundannahmen

Lexicalized Flexible Ty2 (LF-Ty2)

1. Semantische Repräsentationen sind Ausdrücke der Two-sorted Type Theory (Ty2)
2. Das Grundgerüst:
   - Basic Translations
   - (Intensionale) Funktionale Applikation
3. Type Shifting
4. Anwendungsbeispiele:
   - Quantoren in Objektposition
   - Skopusambiguitäten

Das Grundgerüst

- Die logische Form eines Zeichens ist ein Ty2-Ausdruck. Wo zu Illustrationszwecken ausreichend arbeiten wir mit einem nichtintensionalen Fragment:

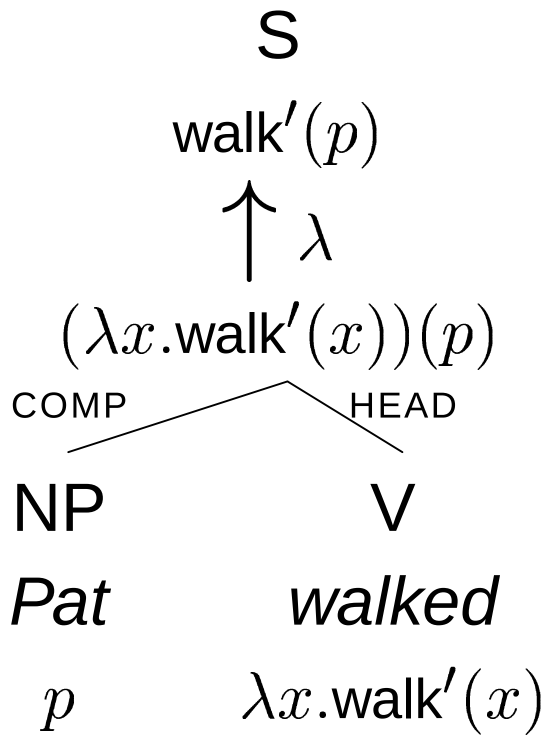
  1(a) *Pat walked* $\leadsto$ $\text{walk}'_{et}(p_t)$
  (b) *Everyone walked* $\leadsto$ $\forall x_e[\text{walk}'_{et}(x)]$
  (c) *Pat read something* $\leadsto$ $\exists x_e[\text{read}'_{e(et)}(p_e, x)]$

- Lexikalischen Elementen wird eine *basic translation* zugeordnet:

  2(a) *Pat* $\leadsto$ p
  (b) *walked* $\leadsto$ $\lambda x_e.\text{walk}'_{et}(x)$
  (c) *read* $\leadsto$ $\lambda y_e \lambda x_e.\text{read}'_{e(et)}(x, y)$
  (d) *everyone* $\leadsto$ $\lambda P.\forall x_e[P_{et}(x)]$

Das Grundgerüst

Die Logische Form einer Phrase ergibt sich durch (intensionale) *Funktionale Applikation* der Logischen Form einer syntaktischen Tochter auf die Logische Form der anderen syntaktischen Tochter.



Das Grundgerüst

Mutatis mutandis gilt Entsprechendes für einen Quantor in Subjektposition.

S

$$\forall x[\text{walk}'(x)]$$

$$\uparrow \lambda$$

$$\forall x[(\lambda y.\text{walk}'(y))(x)]$$

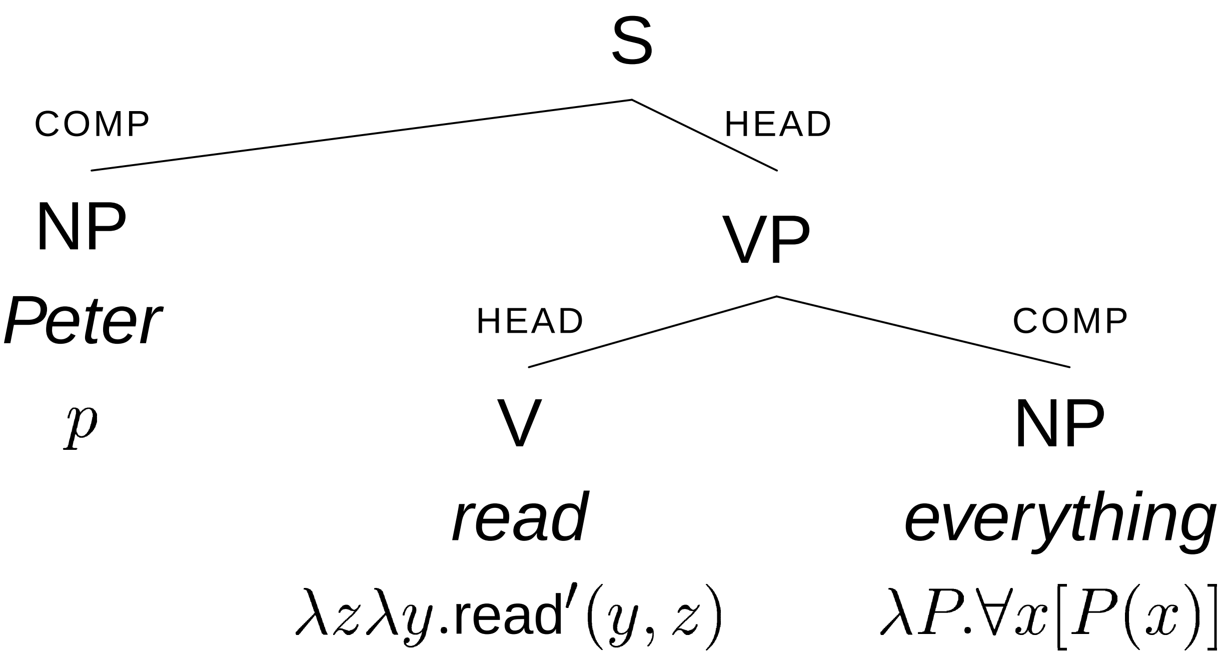
$$\uparrow \lambda$$

$$(\lambda P.\forall x[P(x)])(\lambda y.\text{walk}'(y))$$

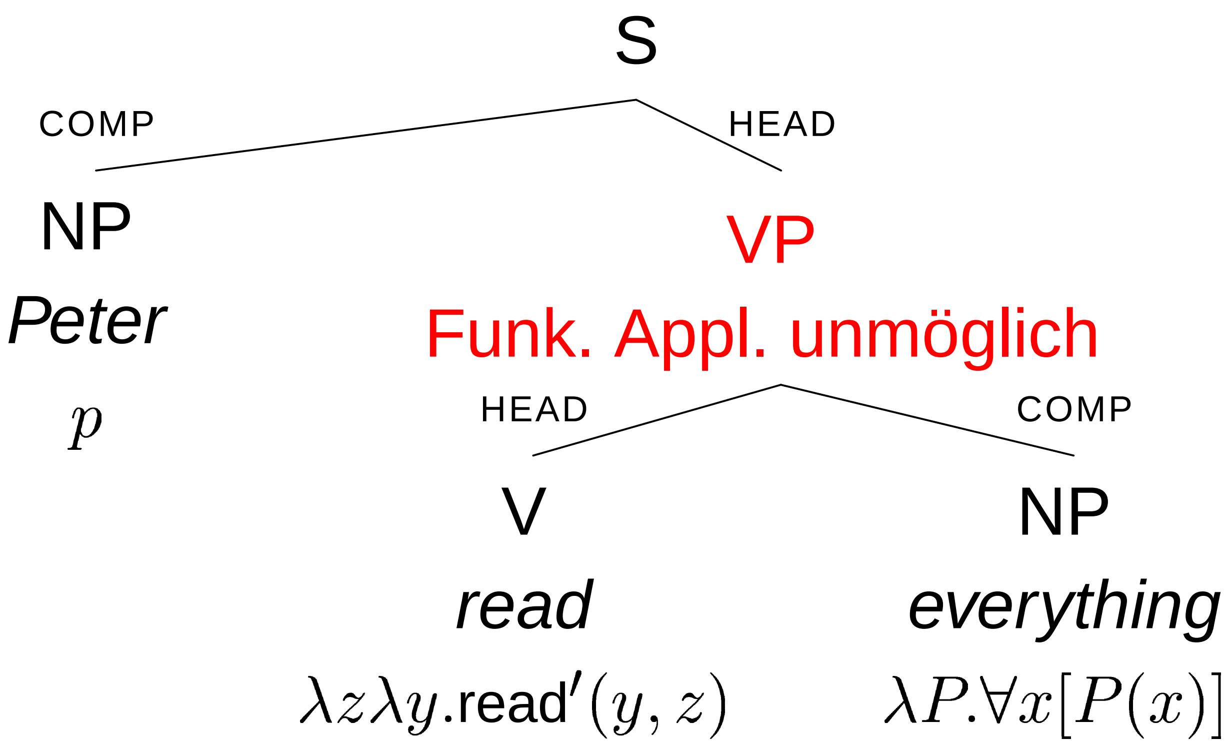
COMP HEAD

NP	V
Everyone	*walked*
$\lambda P.\forall x[P(x)]$	$\lambda y.\text{walk}'(y)$

Flexible Typenzuweisung

- *Peter read everything.*
- Basic translations:
  Peter $\rightsquigarrow p$
  everything $\rightsquigarrow \lambda P.\forall x[P(x)]$
  read $\rightsquigarrow \lambda z \lambda y.\text{read}'(y, z)$
- 

Flexible Typenzuweisung

- *Peter read everything.*
- Basic translations:
  Peter $\rightsquigarrow p$
  everything $\rightsquigarrow \lambda P.\forall x[P(x)]$
  read $\rightsquigarrow \lambda z \lambda y.\text{read}'(y, z)$
- 

Type Shifting Regeln

Argument Raising (extensional):

For each $i \in \mathbb{N}$, AR_i is a relation between two expressions α and β such that
if α is of type $(a_1(\ldots((a_i)(\ldots(a_n b)\ldots)))))$
then β is an expression of the form

$$\lambda x_{a_1,1} \ldots \lambda X_{((a_i)b)b,i} \ldots \lambda x_{a_n,n}.X(\lambda x_{a_i,i}.\alpha(x_1)\ldots(x_i)\ldots(x_n)).$$

Type Shifting Regeln

Argument Raising (extensional):

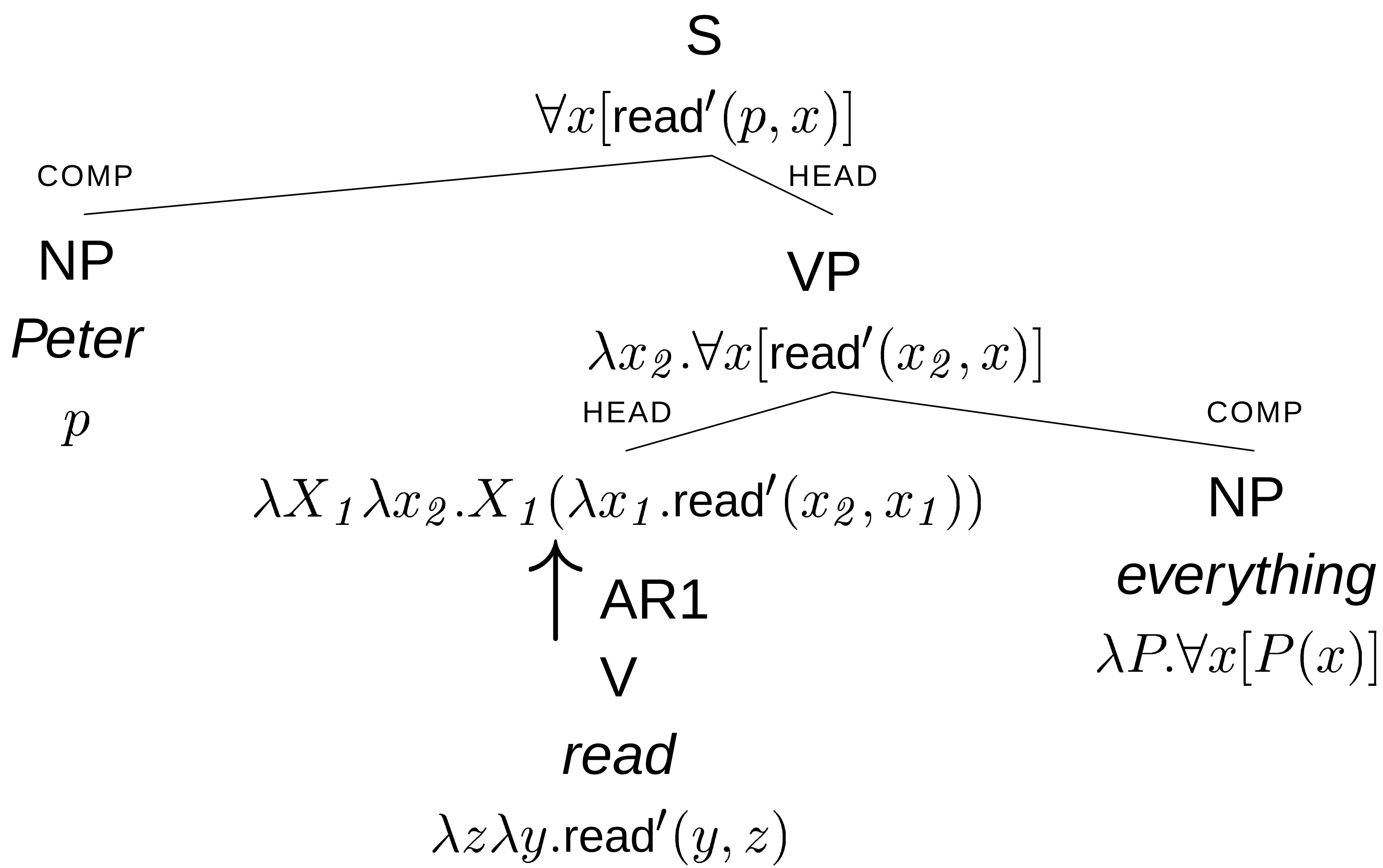
For each $i \in \mathbb{N}$, AR_i is a relation between two expressions α and β such that
if α is of type $(a_1(\ldots((a_i)(\ldots(a_n b)\ldots)))))$
then β is an expression of the form

$$\lambda x_{a_1,1} \ldots \lambda X_{((a_i)b)b,i} \ldots \lambda x_{a_n,n}.X(\lambda x_{a_i,i}.\alpha(x_1)\ldots(x_i)\ldots(x_n)).$$

In unserem Beispiel:

$$\begin{array}{lll} \textit{read} & \leadsto & \lambda y \lambda z.\text{read}'(y, z) \\ & \longrightarrow_{AR1} & \lambda X_1 \lambda x_2.X_1(\lambda x_1.[(\lambda y \lambda z.\text{read}'(y, z))(x_1)(x_2)] \\ & & = \lambda X_1 \lambda x_2.X_1(\lambda x_1.\text{read}'(x_2, x_1)) \end{array}$$

Peter read everything.



Every woman loves a man: $\forall\exists$

S

$\forall x[\mathsf{w}'(x) \rightarrow \exists y[\mathsf{m}'(y) \wedge \mathsf{l}'(x,y)]]$

COMP HEAD

NP

$\lambda P.\forall x[\mathsf{w}'(x) \rightarrow P(x)]$

every woman

VP

$\lambda x.\exists y[\mathsf{m}'(y) \wedge \mathsf{l}'(x,y)]$

HEAD COMP

V

loves

NP

$\lambda Q.\exists y[\mathsf{m}'(y) \wedge Q(y)]$

a man

love $\rightsquigarrow$ $\lambda y \lambda x.\mathsf{love}'(x,y)$

$\longrightarrow_{AR1}$ $\lambda Y \lambda x.Y(\lambda y.\mathsf{love}'(x,y))$

Every woman loves a man: $\exists\forall$

S

$$\exists y[\mathsf{m}'(y) \wedge \forall x[\mathsf{w}'(x) \rightarrow \mathsf{l}'(x,y)]]$$

COMP HEAD

NP

$$\lambda P.\forall x[\mathsf{w}'(x) \rightarrow P(x)]$$

every woman

VP

$$\lambda X.\exists y[\mathsf{m}'(y) \wedge X(\lambda x.\mathsf{l}'(x,y))]$$

HEAD COMP

V

loves

NP

$$\lambda Q.\exists y[\mathsf{m}'(y) \wedge Q(y)]$$

a man

$$\begin{array}{lll} \textit{love} & \rightsquigarrow & \lambda y \lambda x.\mathsf{love}'(x,y) \\ & \longrightarrow_{AR2} & \lambda y \lambda X.X(\lambda x.\mathsf{love}'(x,y)) \\ & \longrightarrow_{AR1} & \lambda Y \lambda X.Y(\lambda y.X(\lambda x.\mathsf{love}'(x,y))) \end{array}$$

LF-Ty2: Zusammenfassung

1. Es gibt
   - *basic translations* für Lexikoneinträge,
   - sowie eine Menge von type shifting Regeln
2. Die logische Form eines Wortes ist
   - seine basic translation oder
   - das Resultat einer endlichen Anzahl von Anwendungen der Type Shifting Regeln.
3. Die logische Form einer Phrase ist
   - das Resultat Funktionaler Applikation der LFs der Töchter
   - in vollständig β-reduzierter Gestalt.