# Semantics 1

May 8, 2012

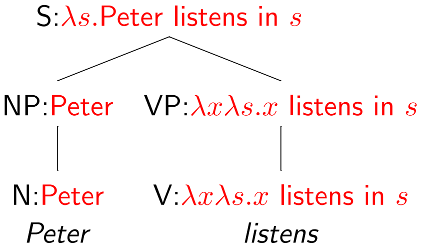
Gerhard Jäger

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

# Compositionality

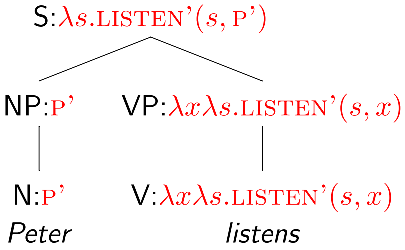
- sentence meaning = lexical meaning + syntax
- example:
  *Peter listens.*
  - sentence meaning: $\lambda s.$Peter listens in $s$
  - lexical meanings:
    - $\|$*Peter*$\|$ = Peter
    - $\|$*listens*$\|$ = $\lambda x \lambda s.x$ listens in $s$
  - syntax: $[_S\ [_{NP}\ [_N$ *Peter* $]\ ]\ [_{VP}\ [_V$ *listens* $]\ ]\ ]$

# Compositionality



# Compositionality

- So far, we used English + some lambda notation as meta language.
- Predicate logic is more precise than English; therefore it is to be preferred as meta language.
- note: all predicates have an additional argument for situations. (This is different from the translations you used in your logics class.)



# Compositionality

- meaning of the mother node can be computed from the meanings of the daughter nodes:
  - for non-branching nodes, mother node and daughter node have the same meaning
  - in an NP-VP structure, the meaning of the VP (which is a function) is applied to the meaning of the NP
- Assumption: this correspondence between syntax and semantics holds for all English sentences. (The correct syntax of English is of course much more complex, but I try to keep things simple for expository purposes.)

# Compositionality

- formally: for each syntactic rule, there is a corresponding semantic rule
- so far, we have
  - $S \rightarrow NP, VP$ :: $\|S\| = \|VP\|(\|NP\|)$
  - $NP \rightarrow N$ :: $\|NP\| = \|N\|$
  - $VP \rightarrow V$ :: $\|VP\| = \|V\|$

# Compositionality

**Schönfinkeling** (a.k.a. Currying)

- meaning of transitive verb: **two-place relation**
- e.g.: *loves* $\rightsquigarrow \{\langle x, y\rangle | \text{LOVE'}(x, y)\}$[1]
- expression as characteristic function:

$$\lambda\langle x, y\rangle \in E \times E.\text{LOVE'}(x, y)$$

- lambda conversion:

$$(\lambda\langle x, y\rangle \in E \times E.\text{LOVE'}(x, y))(\langle a, h\rangle) \quad = \quad \text{LOVE'}(a, h)$$

[1]We ignore situation dependence for a moment.

# Compositionality

**Schönfinkeling**

- What is the meaning of *loves John*? The set of individuals that love John.

$$\|\textit{loves John}\| = \{x|\text{LOVE'}(x, j)\} \approx \lambda x.\text{LOVE'}(x, j)$$

- *loves* can also be considered as a function that maps the meaning of $\alpha$ to the meaning of *loves* $\alpha$:

$$\|\textit{loves}\| = \lambda y \lambda x.\text{LOVE'}(x, y)$$

# Compositionality

## Schönfinkeling

- two-place relation $\{\langle x, y\rangle | \text{LOVE'}(x, y)\}$ is transformed into two-place characteristic function $\lambda\langle x, y\rangle.\text{LOVE'}(x, y)$, which, in turn, can be transformed into a one-place function with a one-place characteristic function as its value:

$$\lambda y \lambda x.\text{LOVE'}(x, y)$$

- general recipe:

$$\{\langle x, y\rangle | R(x, y)\} \rightsquigarrow \lambda\langle x, y\rangle.R(x, y) \rightsquigarrow \lambda y \lambda x.R(x, y)$$

- same principle also applies to $n$-ary relations:

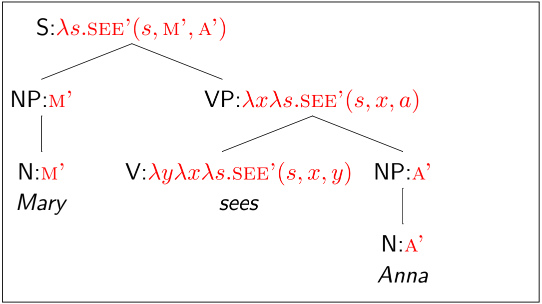
$$\{\langle x_1, \cdots, x_n\rangle | S(x_1, \cdots, x_n)\} \rightsquigarrow \lambda x_n. \cdots .\lambda x_1.S(x_1, \cdots, x_n)$$

Note: Order of the variables in the $\lambda$-prefix is mirror image of their order within the argument frame of the relation!

# Transitive Verbs

- examples: *love, know, see, help, ...*
- express **two-place relations** between individuals
- if situation dependence is added, we get three-place relations
- $\|Mary\ sees\ Anna\| = \lambda s.\text{SEE'}(s, \text{M'}, \text{A'})$
- $\|sees\| = \lambda y \lambda x \lambda s.\text{SEE'}(s, x, y)$

# Transitive Verbs




**Rules:**

- $S \rightarrow NP, VP$ ::
  $\|S\| = \|VP\|(\|NP\|)$
- $NP \rightarrow N$ ::
  $\|NP\| = \|N\|$
- $VP \rightarrow V$::
  $\|VP\| = \|V\|$
- $VP \rightarrow V, NP$ ::
  $\|VP\| = \|V\|(\|NP\|)$

# Boolean Operators

The compositional analysis of the Boolean operators can also be expressed in this format:

**Negation**

- Logical operator of negation can be expressed in two ways in English:
  - *It is not the case that Peter listens.*
  - *Peter doesn't listen.*
- in both cases, the semantic effect is set complementation:

$$\|\textit{Peter does not listen}\| = \lambda s.\neg\text{LISTEN'}(s, p)$$

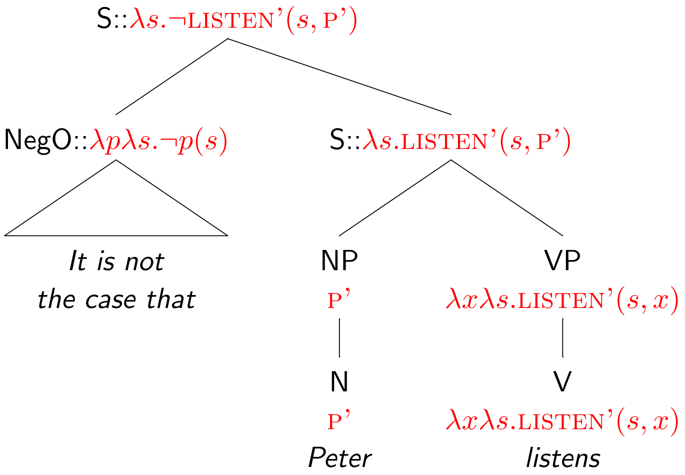
# Boolean Operators

**Negation**

- New rules:
  - $S_1 \to NegO, S_2$ :: $\|S_1\| = \|NegO\|(\|S_2\|)\|$
  - $VP_1 \to NegI, VP_2$ :: $\|VP_1\| = \|NegI\|(\|VP_2\|)\|$
  - $NegO \to$ *It is not the case that* :: $\|NegO\| = \lambda p \lambda s.\neg p(s)$
  - $NegI \to$ *doesn't* :: $\|NegI\| = \lambda P \lambda x \lambda s.\neg P(x, s)$

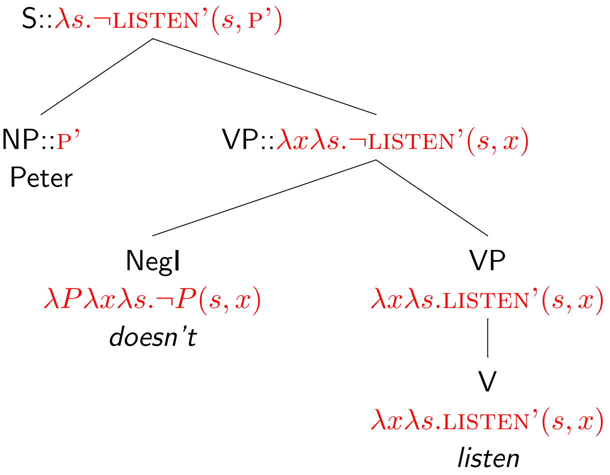
# Boolean Operatoren

**Negation**



# Boolean Operatoren

**Negation**



# Boolean Operatoren

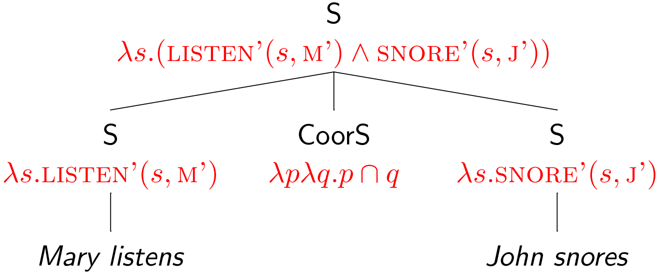
**Sentence Coordination**

- Rules:
  - $S_1 \to S_2, CoorS, S_3$ :: $\|S_1\| = \|CoorS\|(\|S_2\|)(\|S_3\|)$
  - $CoorS \to$ *and* :: $\lambda p \lambda q.p \cap q$
  - $CoorS \to$ *or* :: $\lambda p \lambda q.p \cup q$
- Note:

$$\begin{aligned} \lambda s.\phi \cap \lambda s.\psi &= \lambda s.(\phi \wedge \psi) \\ \lambda s.\phi \cup \lambda s.\psi &= \lambda s.(\phi \vee \psi) \end{aligned}$$

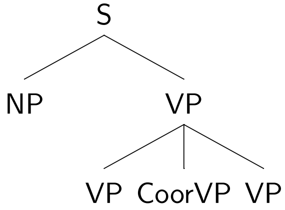
# Boolean Operatoren

**Sentence coordination**



## VP coordination

- Coordination may conjoin two VPs
  - *Peter sleeps and snores.*
  - *John walks and talks.*
- syntactic structure:




- semantics: similar to sentence operators
  *Peter sleeps and snores* ⇔ *Peter sleeps and Peter snores.*

# Boolean Operators

**VP coordination**

- Rules:
  - $VP_1 \to VP_2, CoorVP, VP_3$ :: $\|VP_1\| = \|CoorVP\|(\|VP_2\|)(\|VP_3\|)$
  - $CoorVP \to$ *and* :: $\lambda P \lambda Q \lambda x \lambda s.P(x)(s) \wedge Q(x)(s)$
  - $CoorVP \to$ *or* :: $\lambda P \lambda Q \lambda x \lambda s.P(x)(s) \vee Q(x)(s)$

# Boolsche Operatoren

## VP coordination