Semantic Theory

Summer 2005

Basic Semantic Construction

M. Pinkal / A. Koller

Frege's Principle

... or the Principle of Compositionality:

- The meaning of a complex expression is uniquely determined by the meanings of its sub-expressions and its syntactic structure.

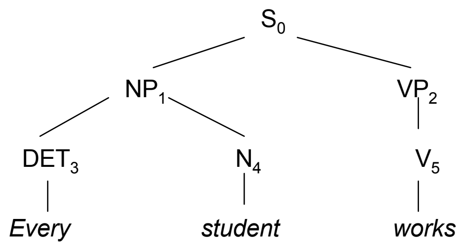
Two levels of interpretation

- Semantic analysis of a NL expression in a logical framework is a two-step process – construction of a semantic representation and its truth-conditional interpretation. Accordingly, the Compositionality Principle has two version:
- The semantic representation of a NL expression is uniquely determined by the semantic representation of its sub-expressions, and the way they are syntactically combined.
- The denotation of a semantic representation is uniquely determined by the denotation of its sub-expressions (and their syntactic combination).

Semantic construction: Three basic rules

- Rule of functional application:
  If A is a binary branching node with daughters B and C (in arbitrary order), $B \Rightarrow \beta: <\sigma, \tau>$, $C \Rightarrow \gamma:\sigma$, then
  $A \Rightarrow \beta(\gamma):\tau$
- Rule of non-branching nodes:
  If A is a non-branching node with daughter B and $B \Rightarrow \beta$, then $A \Rightarrow \beta$ as well.
- Rule of lexical nodes:
  If A is a pre-terminal node with lexical daughter *a*, then $A \Rightarrow a'$, where *a′* is the semantic representation of *a* provided by the lexicon.

An example



$DET_3 \Rightarrow \lambda F \lambda G \forall x(F(x) \rightarrow G(x))$: <<e,t>,<<e,t>,t>> $N_4 \Rightarrow$ student' : <e,t>

$NP_1 \Rightarrow \lambda F \lambda G \forall x(F(x) \rightarrow G(x))(\text{student})$: <<e,t>,t> $\Leftrightarrow_\beta \lambda G \forall x(\text{student}(x) \rightarrow G(x))$

$V_5 \Rightarrow$ work': <e,t>

$VP_2 \Rightarrow$ work': <e,t>

$S_0 \Rightarrow \lambda G \forall x(\text{student}(x) \rightarrow G(x))(\text{work})$: t $\Leftrightarrow_\beta \forall x(\text{student}(x) \rightarrow \text{work}(x))$

- *big elephant, talented logician, alleged murderer* demonstrate that A+N constructions cannot be analysed as conjunction of two standard one-place predicates. In the general case, the constants big', talentd', alleged' representing the adjective meaning have type <<e,t>,<e,t>>.
- There is a frequent special case of so-called "interesective" or "referential" adjectives, however. A *white elephant* is an animal that is white and an elephant, a *married logician* a person who is married and is a logician. The denotation of an A+N construction can be obtained by froming the intersection between the N denotation with an underlying set of "white objects" or "married persons" referred to by the adjective.

Reconsidering attributive adjectives [2]

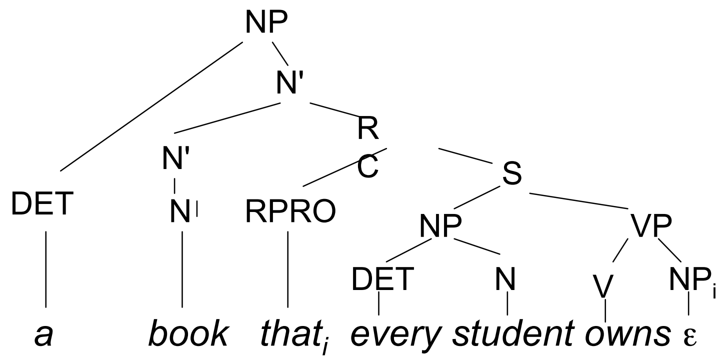
- We can represent the special semantics of intersective adjectives as λ-expressions of the higher type (<<e,t>,<e,t>>), which use a lower type predicate (married*, white*) representing their specific lexical meaning representation:
  - $\lambda F \lambda x$ [married*(x) $\wedge$ F(x)]: <<e,t>,<e,t>>
- For *married student*, we get:
  - $\lambda F \lambda x$ [married*(x) $\wedge$ F(x)](student') :<e,t>

    $\Leftrightarrow_\beta$ λx [married*(x) $\wedge$ student'(x)]

A general strategy for semantic modelling

- Select the type for expressions of a lexical category as high as needed
  - to cover all lexical items of that category
  - to fit into the compositional process of semantic construction
- Encode the often much simpler meaning of specific (sub-types) of lexical items as lambda expressions with an appropriate underlying predicate.
- This way, the requirements of a uniform and straightforward semantic construction and a simple and direct resulting representation of sentence meanings can be fullfilled at the same time (in many cases).

Relative Clauses



Semantic Theory 2005 © M. Pinkal/A.Koller UdS Computerlinguistik 9

Semantics of relative clauses

- Syntactic structure provides non-local information about the relation holding between the relative pronoun and the empty (object) NP (here expressed via co-indexing)
- A pair of semantic composition rules transfers this information into the semantic interpretation: The empty NP translates to a variable, and later this variable is bound by abstraction, where the variable abstrcted over is read off the index of the relative pronoun.

Semantic Theory 2005 © M. Pinkal/A.Koller UdS Computerlinguistik 10

Two more construction rules

- Empty NP rule:
  If A is an empty NP node indexed with i, then A $\Rightarrow x_i$
- Relative clause rule:
  If A is a relative clause with with daughters B and C, B a relative pronoun indexed with i, C $\Rightarrow \gamma$:t, then

  A $\Rightarrow \lambda F\, \lambda x_i\, [Fx \wedge \gamma]$: <<e,t>,<e,t>>

Quantificational NPs and transitive verbs

Every student presented a paper

$\forall d\, (student(d) \rightarrow \exists p\, (paper(p) \wedge present(d,p)))$

Quantificational NPs and transitive verbs

A composition problem: There is a type mismatch between subject NP, object NP and relational verb representation.

- *every student* ⇒ $\lambda F \forall x(\text{student}'(x) \rightarrow F(x))$: <<e,t>,t>
- *a paper* ⇒ $\lambda G\, \exists y(\text{paper}'(y) \wedge G(y))$: <<e,t>,t>
- *presented* ⇒ present': <e,<e,t>>

... and an attempt for a solution: Raise the type of the first-order relation:

present': <<<e,t>,t>,<e,t>>

Non-referential arguments

- John finds a unicorn $\models \exists x\, \text{unicorn}'(x)$
- John seeks a unicorn $\not\models \exists x\, \text{unicorn}'(x)$

- Subject position of verbs is always referential.
- Direct object position of some verbs is referential, of some other verbs isn't.
- Semantic representation pattern for referential verbs:

  $\lambda Q\, \lambda x[Q(\lambda y[\text{find}^*(y)(x)])]$: <<<e,t>,t>,<e,t>>,

  where find*: <<e,t>,t>

Example

- *presented* $\Rightarrow \lambda Q\ \lambda x[Q(\lambda y[\text{present}^*(y)(x)])]$: <<<e,t>,t>,<e,t>>
- *a paper* $\Rightarrow \lambda G\ \exists z(\text{paper}'(z) \wedge G(z))$: <<e,t>,t>
- *presented a paper* $\Rightarrow \lambda Q\ \lambda x[Q(\lambda y[\text{present}^*(y)(x)])]$

  $(\lambda G\ \exists z(\text{paper}'(z) \wedge G(z)))$

  $\Leftrightarrow_\beta \lambda x[\lambda G\ \exists z(\text{paper}'(z) \wedge G(z))(\lambda y[\text{present}^*(y)(x)])]$

  $\Leftrightarrow_\beta \lambda x[\exists z(\text{paper}'(z) \wedge \lambda y[\text{present}^*(y)(x)](z))]$

  $\Leftrightarrow_\beta \lambda x[\exists z(\text{paper}'(z) \wedge \text{present}^*(z)(x))]$
- *every student* $\Rightarrow \lambda G \forall x(\text{student}'(x) \rightarrow G(x))$ <<e,t>,t>

  $\Leftrightarrow_\beta \forall x(\text{student}'(x) \rightarrow \exists z(\text{paper}'(z) \wedge \text{present}^*(z)(x)))$