Semantic Theory: Scope

Summer 2007

M.Pinkal/ S. Thater

Sentence Semantics

- Step 1: FOL Representations
- Step 2: Types and Higher-order Logic
  - Higher-order expressions (higher-order predicates, adjectives, degree modifiers)
  - Function application as basic operation for semantics construction
  - Unified, compositional semantics for noun phrases
- Step 3: λ-expressions and β-reduction
  - Higher-order expressions for semantic composition
  - Obtaining FOL sentence representations through β-reduction
  - Semantics Construction with Transitive Verbs
- Step 4: Treatment of scope variation
  - Cooper Storage
  - Underspecification

Semantics Construction: Basic rules

- Rule of functional application:

A (B C)

$$\frac{B \Rightarrow \beta: \langle\sigma, \tau\rangle \qquad C \Rightarrow \gamma: \sigma}{A \Rightarrow \beta(\gamma): \tau} \quad \text{or} \quad \frac{B \Rightarrow \beta: \sigma \qquad C \Rightarrow \gamma: \langle\sigma, \tau\rangle}{A \Rightarrow \gamma(\beta): \tau}$$

- Rule of non-branching nodes:

A | B

$$\frac{B \Rightarrow \beta: \tau}{A \Rightarrow \beta: \tau}$$

Semantics Construction: Basic rules

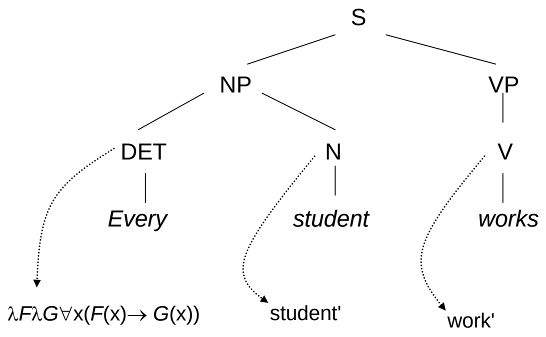
- Rule of lexical nodes:

A | a

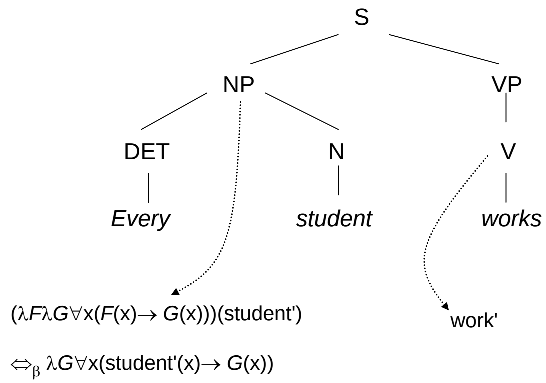
$$\frac{}{A \Rightarrow \beta: \tau}$$

The semantic representation β for the word "a" is supplied by the lexicon.

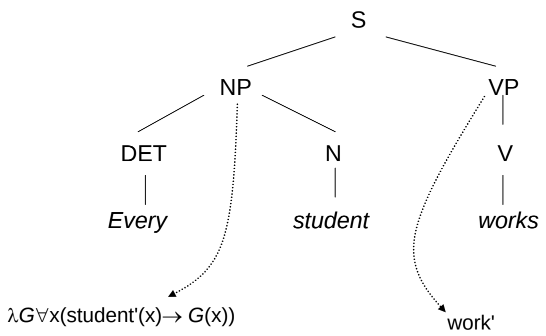
An example



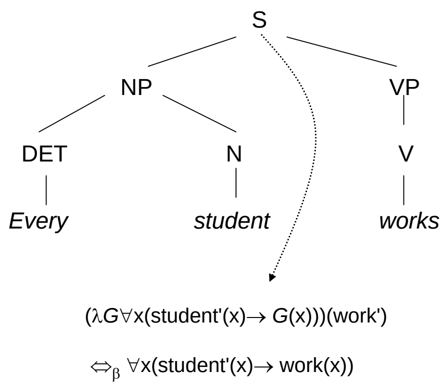
An example



An example



An example



Scope: Terminology

- Logic: Quantifier & Scope

  $\forall x(\text{student}'(x) \rightarrow \text{work}(x))$

- NL Semantics
  - Determiner+Restriction form NP-Denotation („Generalized Quantifier“)
  - NP Denotation is applied to its Nuclear Scope

  Every'(student')(work')

A Note on Notation

- Either: Use expanded notation from the beginning (e.g., $\lambda G \lambda G \forall x(F(x) \rightarrow G(x))$), and simplify (i.e., beta-reduce) as early as possible
- Or: Use abbreviations (every'), and expand them later:
  - Every'(student')(work')
  - $\lambda G \lambda G \forall x(F(x) \rightarrow G(x))$(student')(work')
- Or: Combine both in a sensible way
- But: Don‘t rewrite expanded forms, whenever you can avoid it

Variable NP Scope

- *Every linguist speaks two languages*
- *Our company has an expert for every problem*
- *A search engine for every subject*

NPs and scope-sensitive operators

- *Every student didn‘t pay attention*
- *Every citizen can become president*
- *During his visit to China, Helmut Kohl intends to visit a factory for CFC-free refrigerators*

The problem of scope variation

- The scope of noun phrases is not determined by the syntactic position in which they occur.
- Divergence between syntactic and semantic structure is a challenge for compositionality and semantics constructions.
- Scope variation may lead to a proliferation of readings

Scope ambiguity

Every student presents a paper.

(a) $\forall x[\mathit{student}'(x) \rightarrow \exists y[\mathit{paper}'(y) \wedge \mathit{present}'(x,y)]]$

(b) $\exists y[\mathit{paper}'(y) \wedge \forall x[\mathit{student}'(x) \rightarrow \mathit{present}'(x,y)]]$

Every student didn't pay attention.

(a) $\forall x[\mathit{student}'(x) \rightarrow \neg \mathit{pay\text{-}attention}'(x)]$

(b) $\neg \forall x[\mathit{student}'(x) \rightarrow \mathit{pay\text{-}attention}'(x)]$

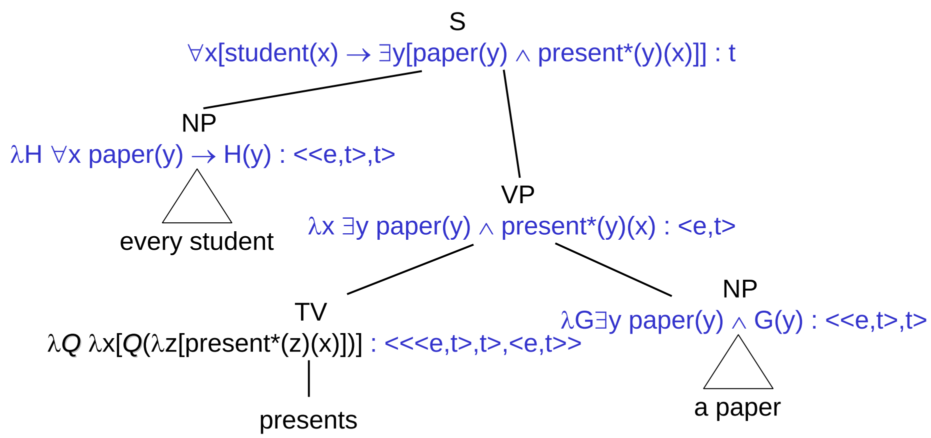
Scope ambiguity

- Every researcher of a company saw some sample.

1. $\forall x(res'(x) \wedge \exists y(cp'(y) \wedge of'(x,y)) \rightarrow \exists z(spl'(z) \wedge see'(x,z))$
2. $\exists z(spl'(z) \wedge \forall x(res'(x) \wedge \exists y(cp'(y) \wedge of'(x,y)) \rightarrow see'(x,z))$
3. $\exists y(cp'(y) \wedge \forall x(res'(x) \wedge of'(x,y)) \rightarrow \exists z(spl'(z) \wedge see'(x,z))$
4. $\exists y(cp'(y) \wedge \exists z(spl'(z) \wedge \forall x(res'(x) \wedge of'(x,y)) \rightarrow see'(x,z))$
5. $\exists z(spl'(z) \wedge \exists y(cp'(y) \wedge \forall x(res'(x) \wedge of'(x,y)) \rightarrow see'(x,z))$

Every researcher of a company saw some samples of most products.

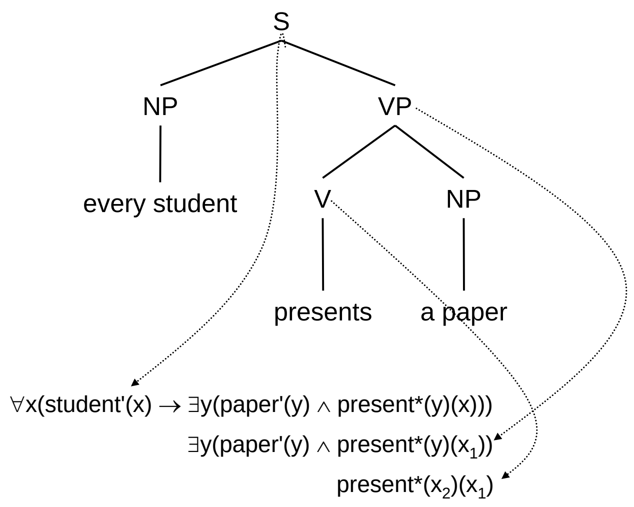
So far, we get only one reading



The problem with scope

- Sentences with scope ambiguities can have multiple semantic representations for a syntactic constituent.
- The order of the scope-bearing elements (quantifiers, negation, adverbs, ...) don't necessarily follow the order of the syntactic combination.
- But: With the approach we have so far, we can only derive a single semantic representation for each constituent.
- How can we solve this problem?

Example



The missing reading

- We get one reading of the sentence by deriving the following terms:

$$\forall x(student'(x) \rightarrow \exists y(paper'(y) \wedge present^*(y)(x)))$$
$$\exists y(paper'(y) \wedge present^*(y)(x_1))$$
$$present^*(x_2)(x_1)$$

- We should be able to construct the second reading correspondingly:

$$\exists y(paper'(y) \wedge \forall x(student'(x) \rightarrow present^*(y)(x)))$$
$$\forall x(student'(x) \rightarrow present^*(x_2)(x))$$
$$present^*(x_2)(x_1)$$

Solving the scope problem: Principles

- We can obtain the second reading by delaying the application of the inner noun phrase.
- To this purpose, we have to:
  - temporarily store the noun phrase denotation away
  - formally bind the object argument position by a variable
  - make sure that the correct argment position will be bound, when the „real“ noun prase denotation is eventually applied

Using lambda abstraction („Quantifying-in“)

- Abstract over the correct variable and then apply the NP representation to the abstracted term.

$\lambda F\forall x(\text{student}'(x)\rightarrow F(x))(\lambda x_1.\ \lambda G\exists y(\text{paper}'(y)\wedge G(y))(\lambda x_2.\text{present}^*(x_2)(x_1)))$
$\lambda G\exists y(\text{paper}'(y)\wedge G(y))(\lambda x_2.\text{present}^*(x_2)(x_1))$
$\text{present}^*(x_2)(x_1)$

$\lambda G\exists y(\text{paper}'(y)\wedge G(y))(\lambda x_2.\ \lambda F\forall x(\text{student}'(x)\rightarrow F(x))(\lambda x_1.\text{present}^*(x_2)(x_1)))$
$\lambda F\forall x(\text{student}'(x)\rightarrow F(x))(\lambda x_1.\text{present}^*(x_2)(x_1))$
$\text{present}^*(x_2)(x_1)$

- Problem: How can we do this compositionally?

Nested Cooper Storage

- One algorithm for deriving such representations compositionally is Nested Cooper Storage (Keller 1988). It repairs some problems of the original Cooper Storage (Cooper 1975).
- Cooper Storage technique is used to compute the set of all semantic readings nondeterministically from a single syntactic analysis.

Nested Cooper Storage: Principles

- The semantic values of syntactic constituents are ordered pairs $\langle\alpha, \Delta\rangle$:
  - $\alpha \in WE_\tau$ is the content
  - Δ is the quantifier store: a set of NP representations that must still be applied.
- At NP nodes, we may store the content in Δ.
- At sentence nodes, we can retrieve NP representations from the store in arbitrary order and apply them to the appropriate argument positions.

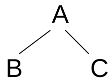
Nested Cooper Storage: Storage

$B \Rightarrow \langle\gamma, \Gamma\rangle$	B is an NP node
$B \Rightarrow \langle\lambda P.P(x_i), \{\langle\gamma, \Gamma\rangle_i\}\rangle$	where $i \in \mathbf{N}$ is a new index

- Using this rule, we can assign more than one semantic value to an NP node.
- The content of the new semantic value is just a placeholder of type <<e,t>,t>, and the old value (including its store) is moved to the store.

Nested Cooper Storage: Old Rules Adjusted

- Rule of functional application:



$$\frac{B \Rightarrow \langle \beta, \Delta \rangle \quad C \Rightarrow \langle \gamma, \Gamma \rangle}{A \Rightarrow \langle \beta(\gamma), \Delta \cup \Gamma \rangle} \quad \text{or} \quad \frac{B \Rightarrow \langle \beta, \Delta \rangle \quad C \Rightarrow \langle \gamma, \Gamma \rangle}{A \Rightarrow \langle \gamma(\beta), \Delta \cup \Gamma \rangle}$$

- Rule of non-branching nodes:

$$\frac{B \Rightarrow \langle \beta, \Delta \rangle}{A \Rightarrow \langle \beta, \Delta \rangle}$$

- Rule of lexical nodes:

$$\frac{}{A \Rightarrow \langle \beta, \varnothing \rangle}$$

Nested Cooper Storage: Principles

- A syntactic constituent may be associated with multiple semantic values of this form.
- A lambda term M counts as a semantic representation for the entire sentence iff we can derive ⟨M, ∅⟩ as a value for the root of the syntax tree.
- Hence, there may be more than one valid semantic representation for the complete sentence.

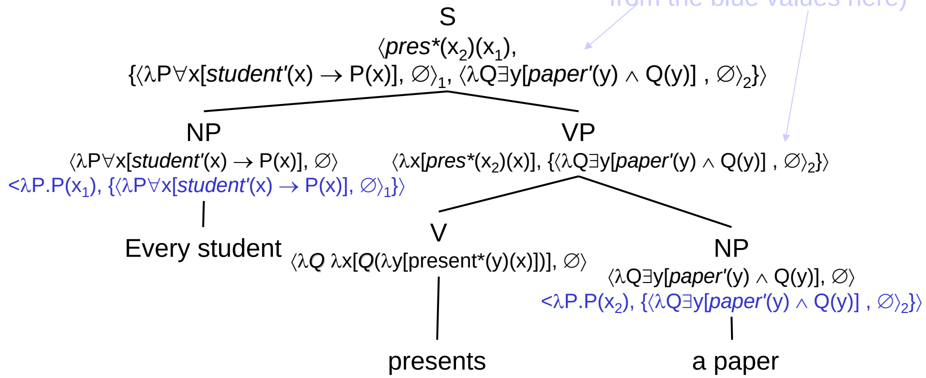
Nested Cooper Storage: Retrieval

$$\frac{A \Rightarrow \langle \alpha, \Delta \cup \{\langle \gamma, \Gamma \rangle_i\} \rangle \qquad \text{A is any sentence node}}{A \Rightarrow \langle \gamma(\lambda x_i \alpha), \Delta \cup \Gamma \rangle}$$

- Using this rule, we can apply a stored NP.
- At this point, the correct λ-abstraction for the variable associated with the stored element is introduced.
- The old store Γ is released into the store for A.

Nested Cooper Storage: Example

Every student presents a paper. (only showing the results from the blue values here)



Retrieval: Reading 1

- By applying the Retrieval rule, we can derive the following representation for the S node:

$\langle \mathit{pres^*}(x_2)(x_1), \{\langle \lambda P \forall x[\mathit{student}(x) \rightarrow P(x)], \varnothing\rangle_1,$

$\langle \lambda Q \exists y[\mathit{paper}(y) \wedge Q(y)]\ , \varnothing\rangle_2\}\rangle \Rightarrow_R \langle \lambda Q \exists y[\mathit{paper}(y) \wedge Q(y)](\lambda x_2.\mathit{pres^*}(x_2)(x_1)),$
$\{\langle \lambda P \forall x[\mathit{student}(x) \rightarrow P(x)], \varnothing\rangle_1 \}\rangle$

$\Rightarrow_\beta \langle \exists y[\mathit{paper}(y) \wedge \mathit{pres^*}(y)(x_1)], \{\langle \lambda P \forall x[\mathit{student}(x) \rightarrow P(x)], \varnothing\rangle_1 \}\rangle$

$\Rightarrow_R \langle \lambda P \forall x[\mathit{student}(x) \rightarrow P(x)](\lambda x_1.\exists y[\mathit{paper}(y) \wedge \mathit{pres^*}(y)(x_1)]), \varnothing\rangle$

$\Rightarrow_\beta \langle \forall x[\mathit{student}(x) \rightarrow \exists y[\mathit{paper}(y) \wedge \mathit{pres^*}(y)(x)]], \varnothing\rangle$

Retrieval: Reading 2

$\langle \mathit{pres^*}(x_2)(x_1), \{\langle \lambda P \forall x[\mathit{student}(x) \rightarrow P(x)], \varnothing\rangle_1,$
$\langle \lambda Q \exists y[\mathit{paper}(y) \wedge Q(y)]\ , \varnothing\rangle_2\}\rangle \Rightarrow_R \langle \lambda P \forall x[\mathit{student}(x) \rightarrow P(x)]\ (\lambda x_1.\mathit{pres^*}(x_2)(x_1)),$
$\{\langle \lambda Q \exists y[\mathit{paper}(y) \wedge Q(y)]\ , \varnothing\rangle_2\}\rangle$

$\Rightarrow_\beta \langle \forall x[\mathit{student}(x) \rightarrow \mathit{pres^*}(x_2)(x)], \{\langle \lambda Q \exists y[\mathit{paper}(y) \wedge Q(y)]\ , \varnothing\rangle_2\}\rangle$

$\Rightarrow_R \langle \lambda Q \exists y[\mathit{paper}(y) \wedge Q(y)](\lambda x_2.\forall x[\mathit{student}(x) \rightarrow \mathit{pres^*}(x_2)(x)]), \varnothing\rangle$

$\Rightarrow_\beta \langle \exists y[\mathit{paper}(y) \wedge \forall x[\mathit{student}(x) \rightarrow \mathit{pres^*}(y)(x)]], \varnothing\rangle$

Compositionality

- The Compositionality Principle as stated earlier: The meaning of a complex expression is uniquely determined by the meaning of its subexpressions and its syntactic structure.
- Nested Cooper Storage shows: We can maintain this principle even in the face of semantic (scope) ambiguity, if we use a relaxed concept of „meaning".

Compositionality and NCS

- Two versions of the Compositionality Principle:
  - on the level of denotations
  - on the level of semantic representations
- Nested Cooper Storage is clearly compositional on the level of semantic representations - but in a less straightforward way than last week's construction algorithm.
- Compositional on the level of denotations: only in a very indirect sense.

Scope islands

- Nested Cooper Storage makes the simplifying assumption that NPs can be retrieved at all sentence nodes.
- This is not true in general because sentence-embedding verbs create scope islands:
  - John said that he saw every girl. (1 reading)
- Quantifiers may not be lifted across the S node of the embedded clause; the sentence cannot mean "for every girl x, John said that he saw x".

Scope ambiguities in real-world texts

- Some large-scale grammars (e.g. the English Resource Grammar) compute semantic representations with scope.
- The ERG analyses all NPs as scope bearers to keep the grammar simple. (This is not necessarily correct: proper names, definites, etc.)
- Median number of scope readings in the Rondane corpus: 55.
  (But: The median number of semantic equivalence classes is only 3!)

Conclusion

- Last week's type-driven semantics construction is a nice first step.
- But it is fundamentally unable to deal with semantically ambiguous sentences.
- Scope ambiguity: Application order of NP representations can be different from syntactic structure.
- Nested Cooper Storage: Equip semantic representations with a quantifier store to allow flexible application of quantifiers; multiple semantic representations per syntactic constituents allowed.