Semantic Theory
Scope

Manfred Pinkal
Stefan Thater

2008-05-13

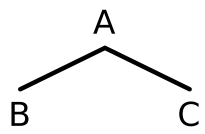
Sentence Semantics

- Step 1: First-order Logic
- Step 2: Type Theory
  - Higher-order predicates
  - Compositional semantics construction
- Step 3: λ-abstraction, β-reduction
  - Higher-order expressions for semantics construction
  - Obtaining first-order expressions by β-reduction
- Step 4: Treatment of Scope-Variations
  - Today: Nested Cooper Storage
  - Next Lecture: Underspecificaton

Basic Composition Rules

- Rule of functional application

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



- Rule for non-branching nodes

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

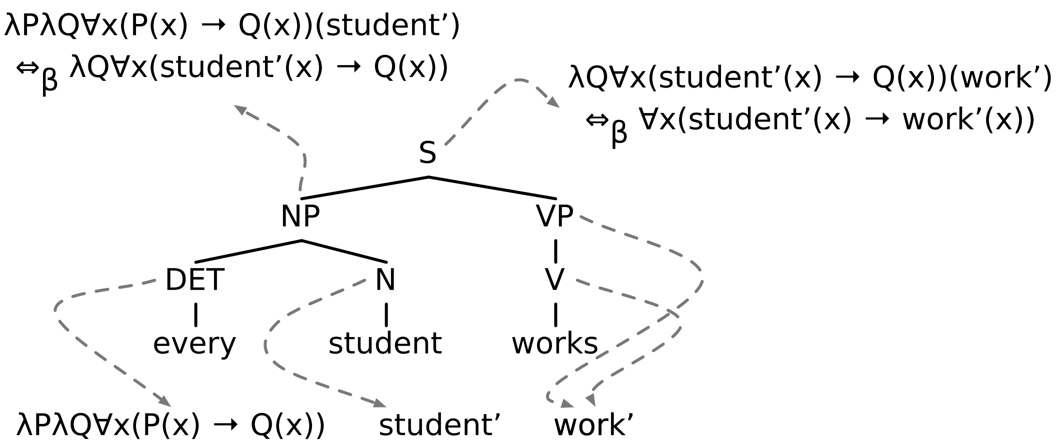
Basic Composition Rules

- Rule for lexical nodes:

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

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

“Every student works”



Scope – Terminology

- Logic: Quantifier and Scope
  - $\forall x(\text{student'}(x) \rightarrow \text{work'}(x))$
- Natural language semantics:
  - Determiner + Restriction form NP-Denotation (“generalized quantifiers”)
  - NP-denotation is applied to its nuclear scope
  - every’(student’)(work’)
  - $(\lambda P \lambda Q \forall x(P(x) \rightarrow Q(x)))(\text{student'})(\text{work'})$

Variable Quantifier-Scope

(1) *Every linguist speaks two languages*

(2) *Our company has an expert for every problem*

(3) Headline: *A search engine for every subject*

Quantifiers and Scope-Sensitive Operators

(1) *Every student didn't pay attention.*

(2) *Every citizen can become president.*

(3) *During his visit to China, Helmut Kohl intends to visit a factory for CFC-free refrigerators*

Scope Ambiguities

(1) *Every student presents a paper.*

(a) ∀x(student’(x) → ∃y(paper’(y) ∧ present’(x,y)))

(b) ∃y(paper’(y) ∧ ∀x(student’(x) → present(x,y)))

(2) *Every student didn’t pay attention.*

(a) ∀x(student’(x) → ¬pay-attention’(x))

(b) ¬∀x(student’(x) → pay-attention’(x))

Scope Ambiguities

(1) *Every researcher of a company saw some sample.*

(a) ∀x((res’(x) ∧ ∃y(cp’(y) ∧ of’(x,y))) → ∃z(spl’(z) ∧ see’(x,z))

(b) ∃z(spl’(z) ∧ ∀x((res’(x) ∧ ∃y(cp’(y) ∧ of’(x,y))) → see’(x,z))

(c) ∃y(cp’(y) ∧ ∀x((res’(x) ∧ of’(x,y)) → ∃z(spl’(z) ∧ see’(x,z)))

(d) ∃y(cp’(y) ∧ ∃z(spl’(z) ∧ ∀x((res’(x) ∧ of’(x,y)) → see’(x,z)))

(e) ∃z(spl’(z) ∧ ∃y(cp’(y) ∧ ∀x((res’(x) ∧ of’(x,y)) → see’(x,z)))

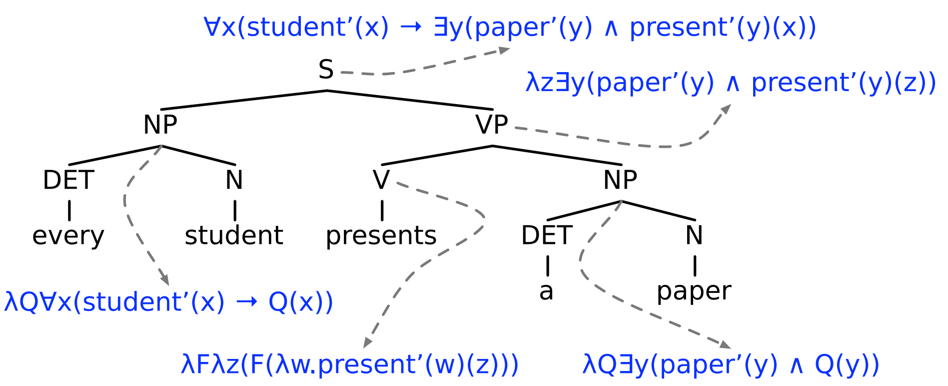
(2) *Every researcher of a company saw some sample of most products.*

- The number of readings can grow exponentially with the number of noun-phrases!

Scope Ambiguities – Problems

- 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 (compositional) semantics constructions.
- Scope ambiguities may lead to a combinatorial explosion of readings.

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, etc. – doesn'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?

Solving the Problem: Principles

(1) *Every student presents a paper.*

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

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

- We can obtain the second reading if we delay the application of the inner noun phrase (“a paper“).
- To this end, we have to:
  - temporarily store the noun phrase representation away
  - 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.

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

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

$$\text{present}^*(x_2)(x_1)$$

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

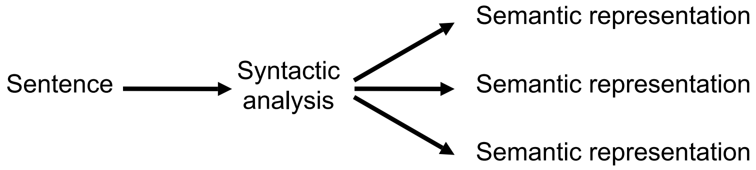
$$\lambda F \forall x(\text{student}'(x) \rightarrow F(x))(\lambda x_1 . \text{present}^*(x_2)(x_1))$$

$$\text{present}^*(x_2)(x_1)$$

- Problem: How can we do this compositionally?

Nested Cooper Storage

- One algorithm for deriving such representations compositionally is Nested Cooper Storage (Keller 1988).
- Nested Cooper Storage is an extension of the original Cooper Storage technique (Cooper 1975).
- (Nested) Cooper Storage computes 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

- Storage: If B is an NP node whose semantic value is $\langle \gamma, \Delta \rangle$, then $\langle \lambda P.P(x_i), \{\langle \gamma, \Delta \rangle_i\} \rangle$ is also a semantic value for B, where $i \in N$ is a new index.

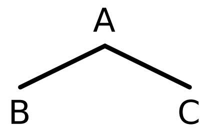
$$\frac{B \Rightarrow \langle \gamma, \Delta \rangle}{B \Rightarrow \langle \lambda P.P(x_i), \{\langle \gamma, \Delta \rangle_i\} \rangle}$$

- Using this rule, we can assign more than one semantic value to NP nodes.
- The content of the new semantic value is a placeholder of type $\langle\langle e,t \rangle,t \rangle$, and the original value (including its store) is moved to the store.

Nested Cooper Storage: Basic Composition Rules (adapted)

- Rule of functional application

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



- Rule for non-branching nodes

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

- Rule for lexical nodes:

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

Nested Cooper Storage

- 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.

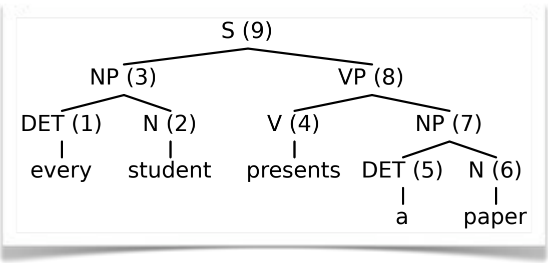
Nested Cooper Storage: Retrieval

- If B is a sentence node, we can retrieve quantifiers from the store:

$$\frac{B \Rightarrow \langle \alpha, \Delta \cup \{\langle \gamma, \Gamma \rangle_i\} \rangle}{B \Rightarrow \langle \gamma(\lambda x_i.\alpha), \Delta \cup \Gamma \rangle}$$

- Using this rule, we can apply a previously 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.

An Example



(1) $\langle \lambda F \lambda P \forall x(F(x) \rightarrow P(x)), \varnothing \rangle$

(2) $\langle student', \varnothing \rangle$

(3) $\langle \lambda F \lambda P \forall x(F(x) \rightarrow P(x))(student'), \varnothing \rangle$

$\langle \lambda P \forall x(student'(x) \rightarrow P(x)), \varnothing \rangle$ (β-reduction)

$\langle \lambda P.P(x_1), \{\langle \lambda P \forall x(student'(x) \rightarrow P(x)), \varnothing \rangle_1\} \rangle$ (storage)

(4) $\langle \lambda G \lambda x(G(\lambda y(pres^*(y)(x)))), \varnothing \rangle$

(7) $\langle \lambda Q \exists y(paper'(y) \wedge Q(x)), \varnothing \rangle$

$\langle \lambda P.P(x_2), \{\langle \lambda Q \exists y(paper'(y) \wedge Q(x)), \varnothing \rangle_2\} \rangle$ (storage)

(8) $\langle \lambda G \lambda x(G(\lambda y(pres^*(y)(x))))(\lambda P.P(x_2)), \{\langle \lambda Q \exists y(paper'(y) \wedge Q(x)), \varnothing \rangle_2\} \rangle$

$\langle \lambda x.pres^*(x_2)(x), \{\langle \lambda Q \exists y(paper'(y) \wedge Q(x)), \varnothing \rangle_2\} \rangle$ (β-redcution)

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

Retrieval: Reading #1

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

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

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

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

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

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

Retrieval: Reading #2

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

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

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

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

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

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

Nested Stores

(1) *[Every researcher of a company] saw some sample.*

- Nested stores are needed to model nested NPs as in (1)
- If both NPs are stored, we must make sure that "every researcher (of)" is retrived before "a company."
  - Otherwise, we would obtain a wrong semantic representation containing a free variable for the complete sentence.
- The nesting of quantifier stores forces the quantifier for the nested NP to take scope over the quantifier for the nesting NP (if both NP-representations are stored).

Compositionality

- The Compositionality Principle as stated earlier:
  The meaning of a complex expression is uniquely determined by the meanings of its sub-expressions 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

- 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 ina 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:”

  (1) *John said that he saw a girl.* (2 readings)

  (2) *John said that he saw every girl.* (1 reading)
- Non-existential quantifiers may not cross scope island boundaries: The second sentence doesn’t mean “for every girl x, John said that he saw x.”

Scope Ambiguities in Real-World Texts

- Some broad-coverage grammars such as the English Resource Grammar (ERG) compute semantic representations with scope.
- The ERG analyses all NPs as scope bearers. This keeps the syntax-semantics interface simple, but is not necessarily correct (proper names, definites, etc).
- The median number of scope readings for typical sentences (in the Rondane corpus) is 55.
- But: The median number of semantic equivalence classes is only 3!

Summary

- The syntax-semantics-interface presented last week is a nice first step, but it is unable to deal with semantically ambiguous sentences.
- Scope ambiguity: Application order of NP representations is not determined by the 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.