The story so far

- We want:
  - logic-based semantic representations that capture the truth conditions of a sentence
    - type theory, tense & modal logic, ...
  - compositional semantics construction
    - lambdas
- This works pretty well up to this point!
- And we could envisage that the system could be conservatively extended to deal with the rest of semantics too.
Some basic rules

- Rule of functional application:

\[
\begin{align*}
A \rightarrow \beta: \langle \sigma, \tau \rangle \\
B \Rightarrow \beta: \langle \sigma, \tau \rangle \\
C \Rightarrow \gamma: \sigma \\
A \Rightarrow \beta(\gamma): \tau
\end{align*}
\]

or

\[
\begin{align*}
A \Rightarrow \gamma(\beta): \tau
\end{align*}
\]

- Rule of non-branching nodes:

\[
\begin{align*}
A \\
B \Rightarrow \beta: \tau \\
A \Rightarrow \beta: \tau
\end{align*}
\]

Some basic rules

- Rule of lexical nodes:

\[
\begin{align*}
A \\
a \\
A \Rightarrow \beta: \tau
\end{align*}
\]

The semantic representation $\beta$ for the word "a" is supplied by the lexicon.
An example

\[
\lambda F, G \forall x (F(x) \rightarrow G(x)) \text{work'}
\]

\[
(\lambda F, G \forall x (F(x) \rightarrow G(x)))(\text{student'})
\]

\[
\equiv \lambda G \forall x (\text{student'}(x) \rightarrow G(x))
\]
An example

\[
\lambda G \forall x (\text{student}'(x) \rightarrow G(x)) \Rightarrow \forall x (\text{student}'(x) \rightarrow \text{work}(x))
\]
However ...

- ... perhaps we made an assumption that is not generally correct!

What does this mean?

- "Now we've got at least one city with all seven religions."
What does this mean?

• Headline: "A search engine for every subject"

(see: http://itre.cis.upenn.edu/~myl/languagelog/archives/002835.html)

What does this mean?

• "Every linguist speaks two languages."
  – the same set of languages for each linguist?
What does this mean?

• "During his visit to China, Helmut Kohl intends to visit a factory for CFC-free refrigerators."
  – are there concrete plans for a particular factory?
  – are there factories for CFC-free refrigerators in China?

What do all these mean?

• "Victoria refuses to trade all her techs."
• "The bishop sent a letter to all priests."
• "It just didn't occur to me that a Barracks might not be there!"
Scope ambiguities

- Some sentences have more than one possible semantic representation:

Every student presents a paper.

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

Every student didn't pay attention.

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

Scope ambiguities

- The number of readings of a sentence with scope ambiguities grows with the number of NPs:

Every researcher of a company saw some sample.

1. \( \forall x [\text{res}'(x) \wedge \exists y [\text{cp}'(y) \wedge \text{of}'(x,y)] \rightarrow \exists z [\text{spl}'(z) \wedge \text{see}'(x,z)]] \)
2. \( \exists z [\text{spl}'(z) \wedge \forall x [\text{res}'(x) \wedge \exists y [\text{cp}'(y) \wedge \text{of}'(x,y)] \rightarrow \text{see}'(x,z)]] \)
3. \( \exists y [\text{cp}'(y) \wedge \forall x [\text{res}'(x) \wedge \text{of}'(x,y)] \rightarrow \exists z [\text{spl}'(z) \wedge \text{see}'(x,z)]] \)
4. \( \exists y [\text{cp}'(y) \wedge \exists z [\text{spl}'(z) \wedge \forall x [\text{res}'(x) \wedge \text{of}'(x,y)] \rightarrow \text{see}'(x,z)]] \)
5. \( \exists z [\text{spl}'(z) \wedge \exists y [\text{cp}'(y) \wedge \forall x [\text{res}'(x) \wedge \text{of}'(x,y)] \rightarrow \text{see}'(x,z)]] \)

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

etc.
But: We get only one reading!

\[ \forall x [\text{student}(x) \rightarrow \exists y \left( \text{paper}(y) \land \text{present}^*(y)(x) \right)] : t \]

\[ \lambda H \forall x \text{paper}(y) \rightarrow H(y) : <<e,t>,t> \]

\[ \lambda x \exists y \left( \text{paper}(y) \land \text{present}^*(y)(x) \right) : <e,t> \]

\[ \lambda Q \lambda x \left[ Q(\lambda z [\text{present}^*(z)(x)]) \right] : <<e,t>,t>,<e,t>,<e,t>,<e,t>,<e,t>,<e,t>> \]

\[ \lambda Q \exists y \left( \text{paper}(y) \land Q(y) \right) : <<e,t>,t> \]

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?
Semantic ambiguity: A picture

Sentence → Syntactic analysis → Semantic representation

Solving the scope problem: Intuition

NP: every student

VP: presents a paper

∀x(student'(x) → ∃y(paper'(y) ∧ present'(y)(x)))

∃y(paper'(y) ∧ present'(y)(x_i))

present'(x_j)(x_i)
The missing reading

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

\[ \forall x (\text{student'}(x) \rightarrow \exists y (\text{paper'}(y) \land \text{present'}(y)(x))) \]
\[ \exists y (\text{paper'}(y) \land \text{present'}(y)(x_1)) \]
\[ \text{present'}(x_2)(x_1) \]

- We could construct the second reading as follows:

\[ \exists y (\text{paper'}(y) \land \forall x (\text{student'}(x) \rightarrow \text{present'}(y)(x))) \]
\[ \forall x (\text{student'}(x) \rightarrow \text{present'}(x_2)(x)) \]
\[ \text{present'}(x_2)(x_1) \]

Solving the scope problem: Principles

- **Structural ambiguity**: We can obtain the two readings by embedding an intermediate term into the NP representations in different orders.
- **Invariant variable binding**: At the same time, we must make sure that the variables will be bound in the same way in both readings.
- To a certain degree, we can solve both problems using lambda abstraction in a clever way.
Using lambda abstraction ("Montague’s Trick")

- Intermediate results are all of type t. Abstract over the correct variable and then apply the NP representation to the abstracted term.

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

- 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 Storages 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
  – $\Delta$ is the quantifier store: a set of NP representations that must still be applied.
• At NP nodes, we may store the content in $\Delta$.
• 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: 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 $\langle M, \emptyset \rangle$ 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: Old Rules

- Rule of functional application:

```
A               B ⇒ ⟨β, Δ⟩
\[\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad"
Nested Cooper Storage: Retrieval

A ⇒ ⟨α, Δ ∪ {⟨γ, Γ1⟩}⟩  A is any sentence node

A ⇒ ⟨γ(λxα), Δ ∪ Γ⟩

• 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.
• This implements Montague's Trick.

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:

\[
\langle \text{pres}^*(x_2)(x_1), \{\langle \lambda x_2 \text{[student]}(x_2) \rightarrow P(x_1)\}, \emptyset\rangle, \langle \lambda Q \exists y [\text{paper}^*(y) \wedge Q(y)], \emptyset\rangle\rangle
\]

\[ \Rightarrow_R \langle \lambda Q \forall x [\text{student}^*(x) \rightarrow P(x)], \emptyset \rangle, \langle \lambda Q \exists y [\text{paper}^*(y) \wedge Q(y)], \emptyset \rangle \rangle \]

\[ \Rightarrow_\beta \langle \exists y [\text{paper}^*(y) \wedge \text{pres}^*(y)(x_1)], \{\langle \lambda x_2 \exists y [\text{paper}^*(y) \wedge Q(y)], \emptyset \rangle \rangle \rangle \]

Retrieval: Reading 2

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

\[
\langle \text{pres}^*(x_2)(x_1), \{\langle \lambda P \forall x [\text{student}^*(x) \rightarrow P(x)], \emptyset\rangle, \langle \lambda Q \exists y [\text{paper}^*(y) \wedge Q(y)], \emptyset\rangle\rangle
\]

\[ \Rightarrow_R \langle \lambda P \forall x [\text{student}^*(x) \rightarrow P(x)], \emptyset \rangle, \langle \lambda Q \exists y [\text{paper}^*(y) \wedge Q(y)], \emptyset \rangle \rangle \]

\[ \Rightarrow_\beta \langle \forall x [\text{student}^*(x) \rightarrow \exists y [\text{paper}^*(y) \wedge \text{pres}^*(y)(x_1)]], \emptyset \rangle \]}
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 – as long as we accept that there are multiple meanings – the principle is also still true if we see NCS as a nondeterministic process.

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.
Other types of scope ambiguities

• Nested Cooper Storage makes the simplifying assumption that only NPs can participate in scope ambiguities.
• This is not true in general:
  – Every student didn't pay attention.
  – Sometimes every student is sleepy.
• NCS can be extended to deal with these, and you'll do it in the exercises, but we'll do something even better next week.

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 a girl. (2 readings)
  – John said that he saw every girl. (1 reading)
• Universal quantifiers may not cross scope island boundaries; the second sentence doesn't mean "for every girl x, John said that he saw x".
De dicto/de re ambiguities

- De dicto/de re ambiguities are a special kind of scope ambiguity in which one scope bearer is a verb:
  \[
  \exists x. \text{factory}(x) \land \text{intend}(hk, \ ^\exists x. \text{visit}(gs, x)) \quad \text{(de re)}
  \]
  \[
  \text{intend}(hk, \ ^\exists x. \text{factory}(x) \land \text{visit}(gs, x)) \quad \text{(de dicto)}
  \]

- We need a more expressive (intensional) logic to represent the different readings, but the ambiguity is just a scope ambiguity and can be resolved by NCS.

- Compare the status of "a factory" to the unicorn in "John seeks a unicorn."

Scope ambiguities in the real world

- Scope ambiguities are not a very intuitive type of ambiguity, and are sometimes not seen as a serious problem for computational linguistics.

- In practice, they are often resolved by context, world knowledge, preferences, etc.

- We consider them here because they pose a fundamental challenge for semantics construction.

- If we want "deep" semantic representations that say something about scope, we must take scope ambiguities into account.
Scope ambiguities in the real world

- Also, 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.