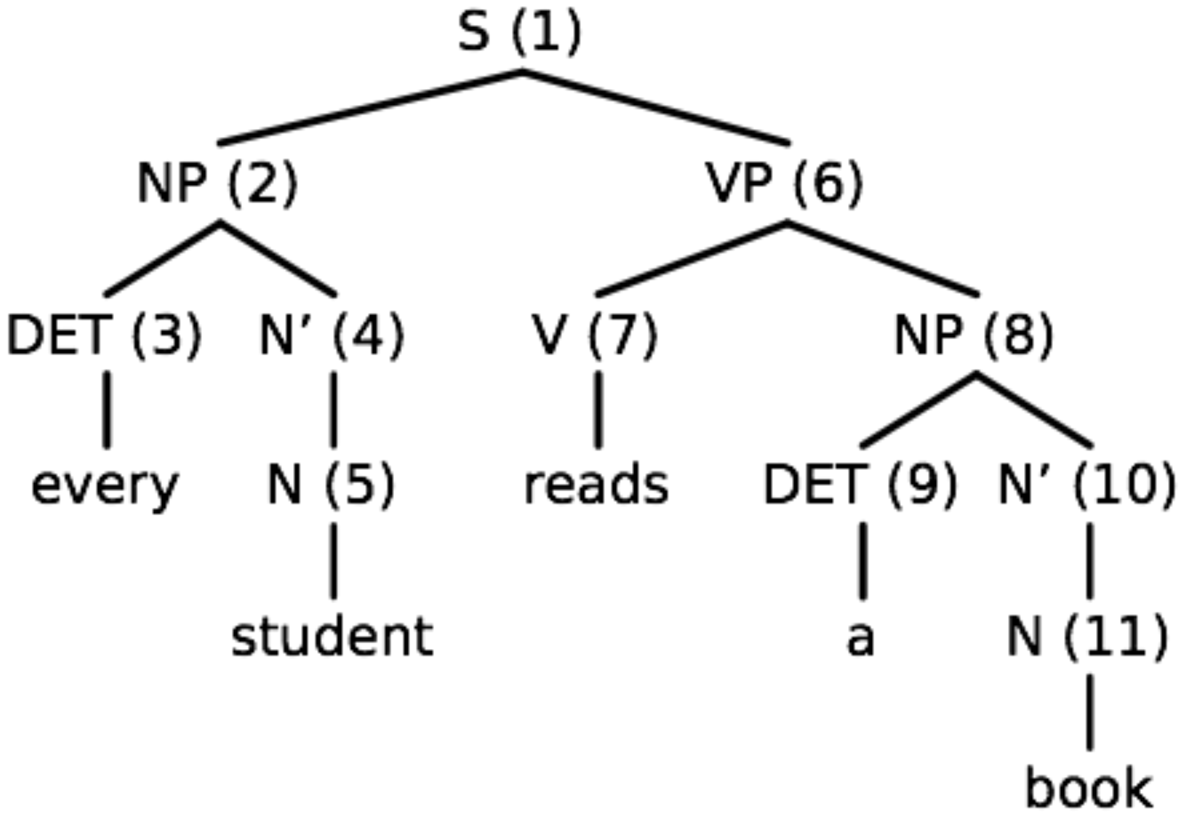
Semantic Theory
Lecture 9: Quantifier Storage/ Intensionality

Manfred Pinkal
FR 4.7 Computational Linguistics and Phonetics

Summer 2014

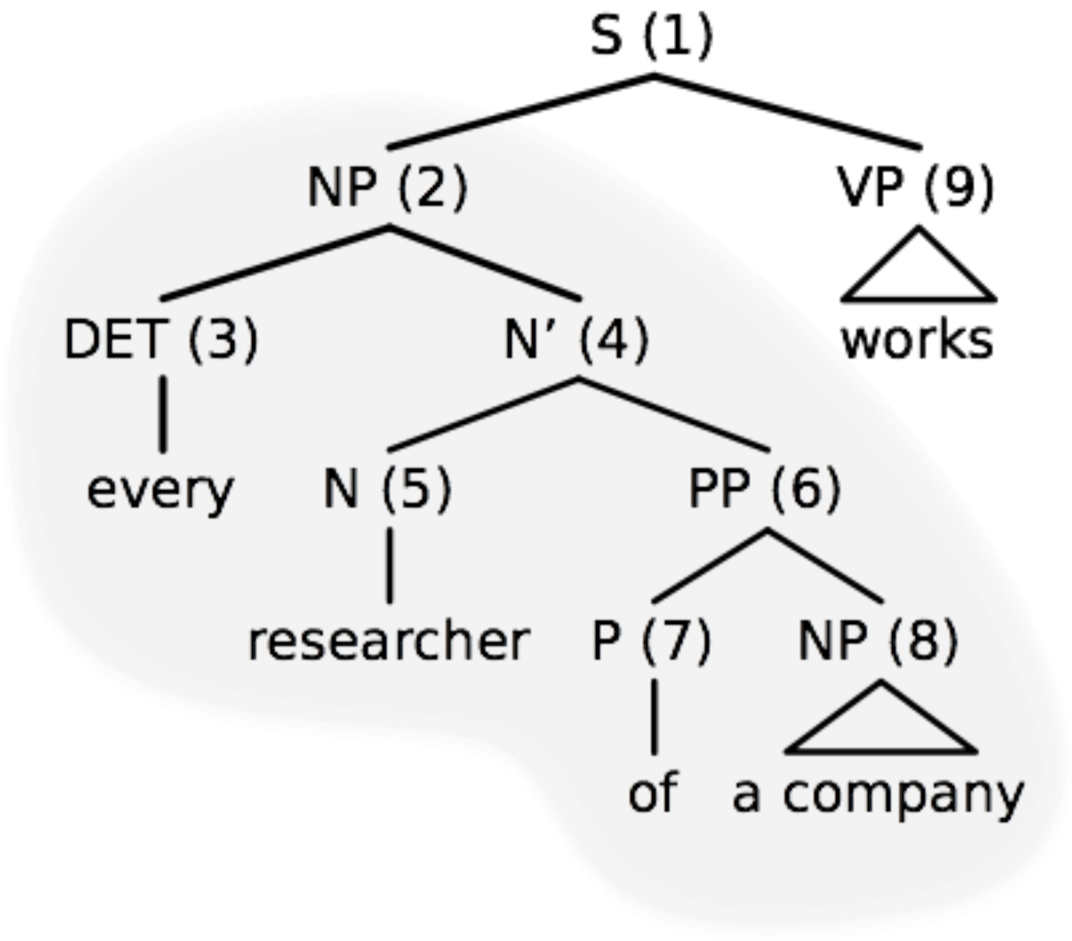
UNIVERSITÄT
DES
SAARLANDES

Every student reads a book



Problem: Nested noun phrases

- *Every researcher of a company works*

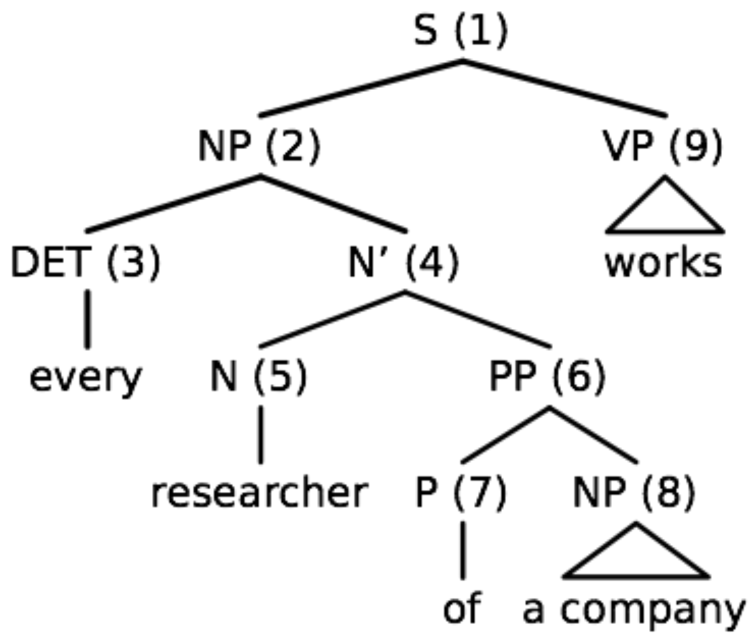


Storage, Preliminary Version

- **Storage:** $\langle Q, \Delta \rangle \Rightarrow_S \langle \lambda P.P(\mathbf{x_i}), \Delta \cup \{\mathbf{[}Q\mathbf{]_i}\}\rangle$
  - if A is an noun phrase with a semantic value $\langle Q, \Delta \rangle$, then select a new index $i \in N$ and add $\langle \lambda P.P(x_i), \Delta \cup \{[Q]_i\}\rangle$ as a semantic value for A.

Nested noun phrases

- (8) $\langle \lambda F(F(\mathbf{x_1})), \{\mathbf{[}\lambda G \exists x(comp(x) \wedge G(x))\mathbf{]_1}\}\rangle$
- (4) $\langle \lambda x(res(x) \wedge of(\mathbf{x_1})(x)), \{\mathbf{[...]_1}\}\rangle$
- (2) $\langle \lambda G \forall y((res(y) \wedge of(\mathbf{x_1})(y)) \rightarrow G(y)), \{\mathbf{[...]_1}\}\rangle$
- $\Rightarrow_S \langle \lambda F(F(\mathbf{x_2})), \{\mathbf{[}\lambda G \forall y((res(y) \wedge of(\mathbf{x_1})(y)) \rightarrow G(y))\mathbf{]_2}, \mathbf{[...]_1}\}\rangle$
- (1) $\langle work(\mathbf{x_2}), \{\mathbf{[...]_2}, \mathbf{[...]_1}\}\rangle$



Retrieval, Preliminary Version

- **Retrieval:** $\langle \alpha, \ \Delta \ \cup \ \{[Q]_i\}\rangle \Rightarrow_R \langle Q(\lambda x_i \ \alpha), \Delta\rangle$
  - if A is any sentence with semantic value $\langle \alpha, \ \Delta\rangle$, and $[Q]_i \in \Delta$, then $\langle Q(\lambda x_i \ \alpha), \Delta - \{[Q]_i\}\rangle$ can be added as a semantic value for A.

Nested noun phrases

- $\langle \mathrm{work}(x_2), \{[Q_2 = \lambda G \forall y((\mathrm{res}(y) \wedge \mathrm{of}(\mathbf{x_1})(y)) \rightarrow G(y))]_2,$
  $[Q_1 = \lambda G \exists x(\mathrm{comp}(x) \wedge G(x))]_1\}\rangle$
- $\Rightarrow_R \langle Q_1(\lambda x_1.\mathrm{work}(x_2)), \{[Q_2]_2\}\rangle$
- $\Leftrightarrow_\beta \langle \exists x(\mathrm{comp}(x) \wedge \mathrm{work}(x_2)), \{[Q_2]_2\}\rangle$
- $\Rightarrow_R \langle Q_2(\lambda x_2.\exists x(\mathrm{comp}(x) \wedge \mathrm{work}(x_2))), \emptyset\rangle$
- $\Leftrightarrow_\beta \langle \forall y((\mathrm{res}(y) \wedge \mathrm{of}(\mathbf{x_1})(y)) \rightarrow \exists x(\mathrm{comp}(x) \wedge \mathrm{work}(y))), \emptyset\rangle$

- **Variable x_1 occurs free.**

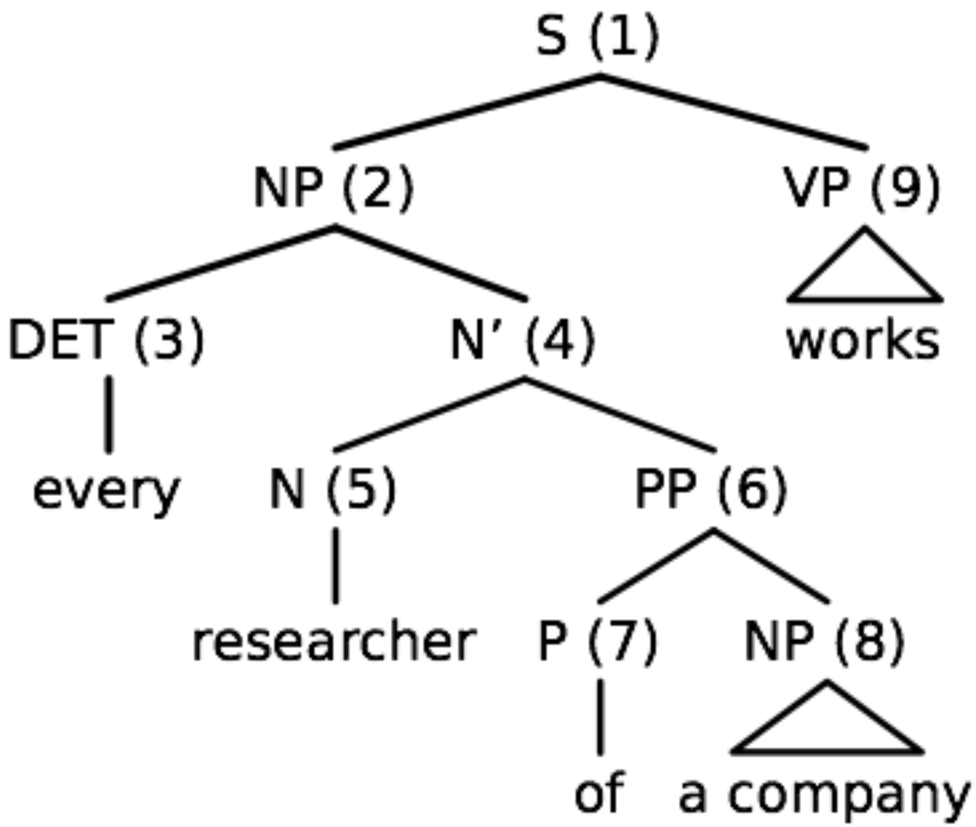
Cooper Storage, Revised

- **Storage:** $\langle Q, \Delta\rangle \Rightarrow_S \langle \lambda P.P(\mathbf{x_i}), \{\boldsymbol{\langle} Q, \Delta \boldsymbol{\rangle_i}\}\rangle$
  - If A is a noun phrase whose semantic value is $\langle Q, \Delta\rangle$, $i \in N$ a new index, then $\langle \lambda P.P(x_i), \{\langle Q, \Delta\rangle_i\}\rangle$ can be added as a semantic value for A.

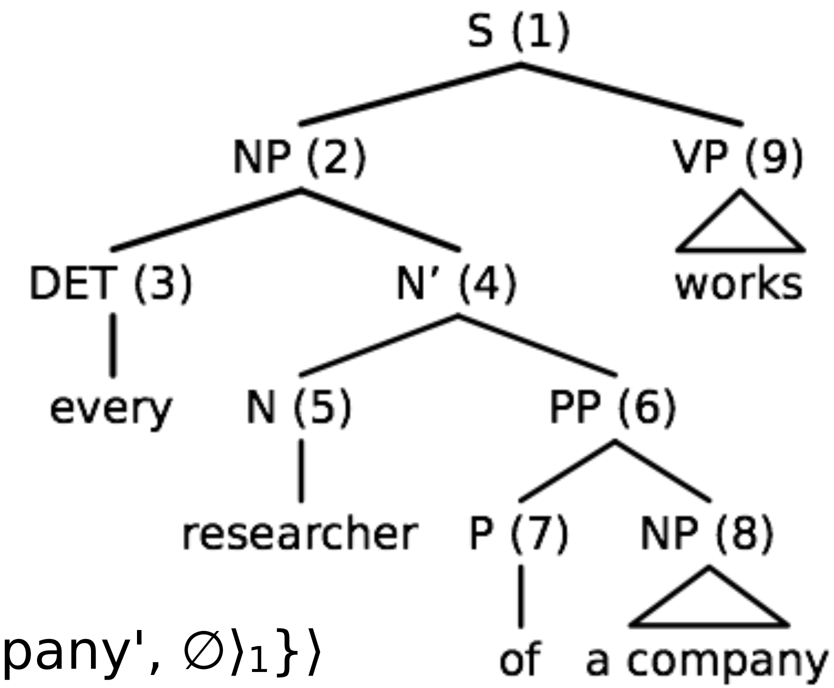
- **Retrieval:** $\langle \alpha, \Delta \cup \{\boldsymbol{\langle} Q, \Gamma \boldsymbol{\rangle_i}\}\rangle \Rightarrow \langle Q(\boldsymbol{\lambda x_i}\ \alpha), \Delta \cup \Gamma\rangle$
  - If A is a sentence with semantic value $\langle \alpha, \Delta\rangle$, $\{\langle Q, \Gamma\rangle_i\} \in \Delta$, then $\langle Q(\lambda x_i.\alpha), \Delta \backslash \{\langle Q, \Gamma\rangle_i\} \cup \Gamma\rangle$ can be added as a semantic value for A.

- **Note:** Deeper embedded quantifiers are **not accessible** for retrieval.

Every reasearcher of a ...



Every reasearcher of a ...



- (8) ⟨a-company', ∅⟩

  $\Rightarrow_S$ ⟨λF.F(x_1), {⟨a-company', ∅⟩$_1$}⟩
- (4) ⟨λy(res'(y) ∧ of'(x_1)(y)), {⟨a-company', ∅⟩$_1$}⟩
- (2) ⟨λG∀z((res'(z) ∧ of'(x_1)(z)) → G(z)), {⟨a-company', ∅⟩$_1$}⟩

  $\Rightarrow_S$ ⟨λF.F(x_2), {⟨every-researcher-of-x_1', {⟨a-company', ∅⟩$_1$}⟩$_2$}⟩
- (9) ⟨work', ∅⟩
- (1) ⟨work(x_2), {⟨every-researcher-of-x_1', {⟨a-company', ∅⟩$_1$}⟩$_2$}⟩

Every reasearcher of a …

- $\langle$work(x_2), {$\langle$every-researcher-of-x_1', {$\langle$a-company', $\emptyset\rangle_1$}$\rangle_2$}$\rangle$

  $\Rightarrow_R$ $\langle$every-researcher-of-x_1'(λx_2.work(x_2)), {$\langle$a-company', $\emptyset\rangle_1$}$\rangle$

  $\Leftrightarrow_\beta$ $\langle\forall z((res'(z) \wedge of'(x_1)(z)) \rightarrow work'(z))$, {$\langle$a-company', $\emptyset\rangle_1$}$\rangle$

  $\Rightarrow_R$ $\langle$a-company' $(\lambda x_1.\forall z((res'(z) \wedge of'(x_1)(z)) \rightarrow work'(z)))$, $\emptyset\rangle$

- $\Leftrightarrow_\beta$ $\langle\exists x(comp'(x) \wedge \forall z((res'(z) \wedge of'(x)(z)) \rightarrow work'(z)))$, $\emptyset\rangle$

Every researcher of a ...

- $\langle \text{work}(x_2), \{\langle \lambda G \forall z(\ldots), \{\langle \lambda G \exists x(\ldots), \emptyset\rangle_1\}\rangle_2\}\rangle$
- $\Rightarrow^*_R \exists x(\text{comp}(x) \land \forall z((\text{res}(z) \land \text{of}(x)(z)) \rightarrow \text{work}(z)))$

- No other reading can be derived via retrieval!
  - But how do we derive the “direct scope” reading?
  - Answer: don’t store, apply quantifiers “in situ”.

Some restrictions on scope

- *Some inhabitant of every midwestern city participated*
  - two readings: (a) direct scope and (b) every ⊲ some
- *Someone who inhabits every midwestern city participated*
  - only the direct scope reading available
- *You will inherit a fortune if every man dies*
  - “every man” cannot take scope over complete sentence
- **Finite clauses can create “scope islands”**
  - Quantifiers must take scope within such clauses

Intensionality

- *Bill expects to pass*
- *It is possible that Bill will pass*
- *Yesterday, it rained*
- In general, sentence operators (type <t,t>) need meaning representations as arguments that are richer than first-order denotations. We distinguish for a sentence
- its **extension**: the truth value
- its **intension**: called the "**proposition**"
- Functor expression requiring intensions as semantic arguments are called "intensional".

Intensionality

- *John is a poor bagpiper*
- *John is a poor speaker of Gaelic*

In the case of predicates,

- extensions are sets of entities
- intensions are called “**properties**”

- *John seeks a unicorn*
- *John seeks an even prime number greater 2.*

Possible-World Semantics

- **A model structure** for a type theoretic language consists of a pair $M = \langle U, \mathbf{W}, V \rangle$, where
  - U is a non-empty domain of individuals
  - W is a non-empty set of possible worlds, disjoint from U
  - V is an interpretation function, which assigns to every $\alpha \in CON_\tau$ an element of D_τ.

- The **domain of possible denotations** for every type τ: $\mathbf{D_\tau}$ is given by:
  - $D_e = U$
  - $\mathbf{D_t = \{0, 1\}^W}$
  - $D_{\langle\sigma, \tau\rangle}$ is the set of all functions from D_σ to D_τ

Adding Time

- **A model structure** for a type theoretic language consists of a pair $M = \langle U, \mathbf{W}, \mathbf{T}, \mathbf{<}, V\rangle$, where
  - U, W and T are non-empty, pairwise disjoint sets of individuals, possible worlds, and time points, respectively
  - $< \subseteq T \times T$ is a strict ordering relation
  - V is an interpretation function, which assigns to every $\alpha \in CON_\tau$ an element of D_τ.

- The **domain of possible denotations** for every type τ: $\mathbf{D_\tau}$ is given by:
  - $D_e = U$
  - $\mathbf{D_t = \{0, 1\}^{W \times T}}$
  - $D_{\langle\sigma, \tau\rangle}$ is the set of all functions from D_σ to D_τ