Semantic Theory
Lecture 8: Quantifier Storage

Manfred Pinkal
FR 4.7 Computational Linguistics and Phonetics

Summer 2014

UNIVERSITÄT
DES
SAARLANDES

Scope Ambiguities

- *Every student speaks two foreign languages*

  a. $\forall x(\text{student}'(x) \rightarrow \exists y(\text{foreign-language}'(y) \wedge \text{speak}^*(y)(x)))$
  b. $\exists y(\text{foreign-language}'(y) \wedge \forall x(\text{student}'(x) \rightarrow \text{speak}^*(y)(x)))$

- *Every student didn't pay attention*

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

- *An American flag stood in front of every building*

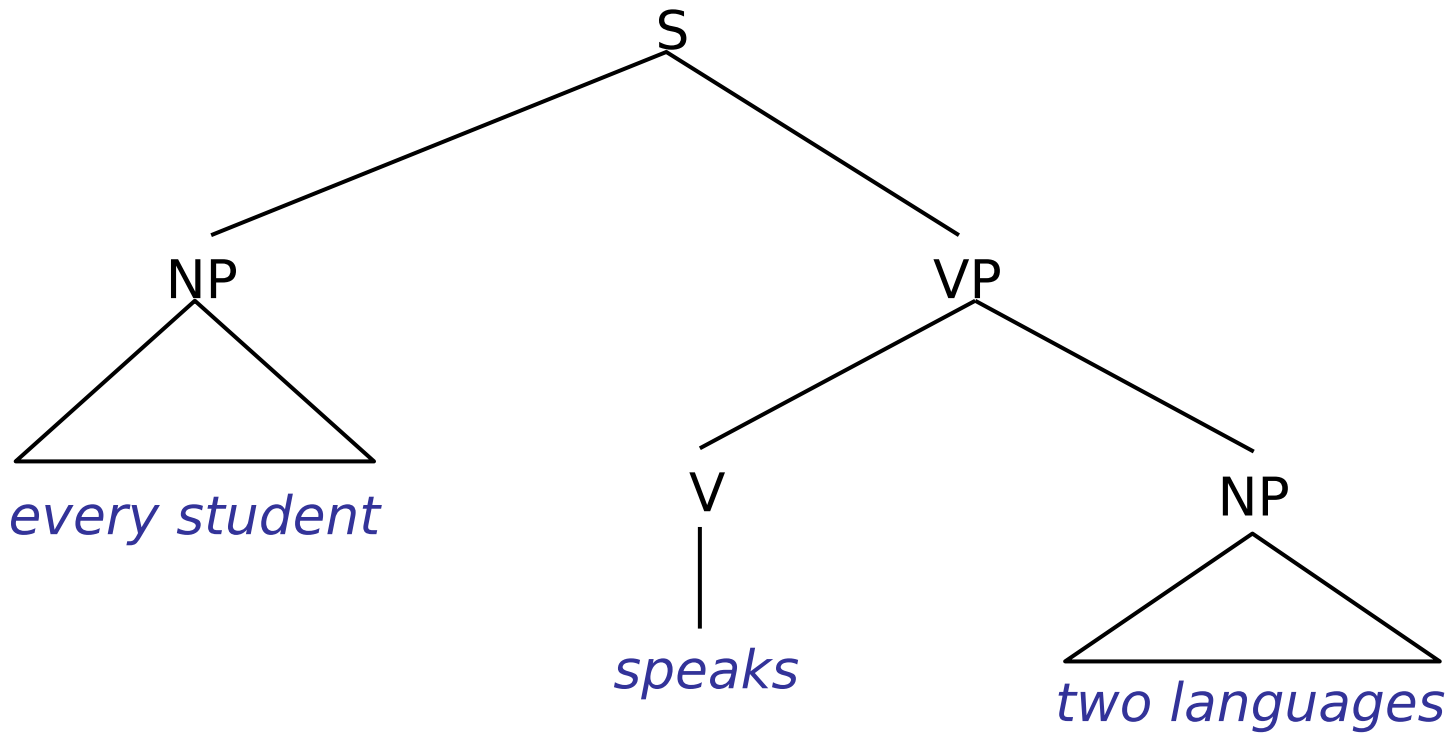
- *John is looking for a book about semantics*

- *Marilyn wants to marry a millionaire*

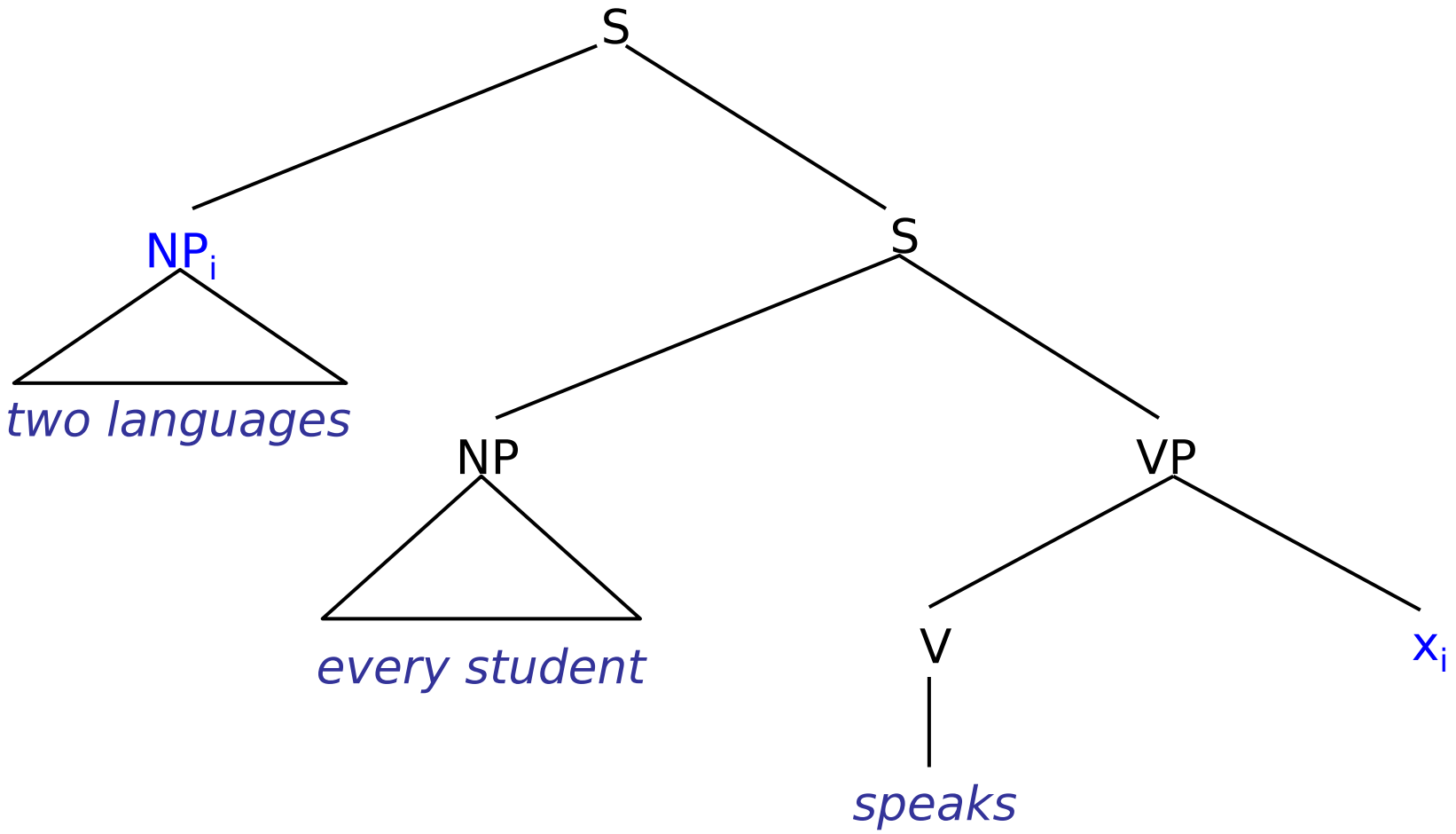
Scope Ambiguities

- Given the standard syntactic analyses and the basic semantic construction rules, we can derive only one scope reading for each sentence.
- To solve the problem, we can either:
  - assume a “deep syntactic structure”, as proposed by linguists in the past, and also used in standard Montague Grammar (Syntactic transformation rule of “Quantifier Raising”)
  - or modify the semantic construction principles, to make quantifier scope (partially) independent of syntactic structure.
- Quantifier storage (also called “Cooper Storage”) is the standard approach to realize the latter idea.

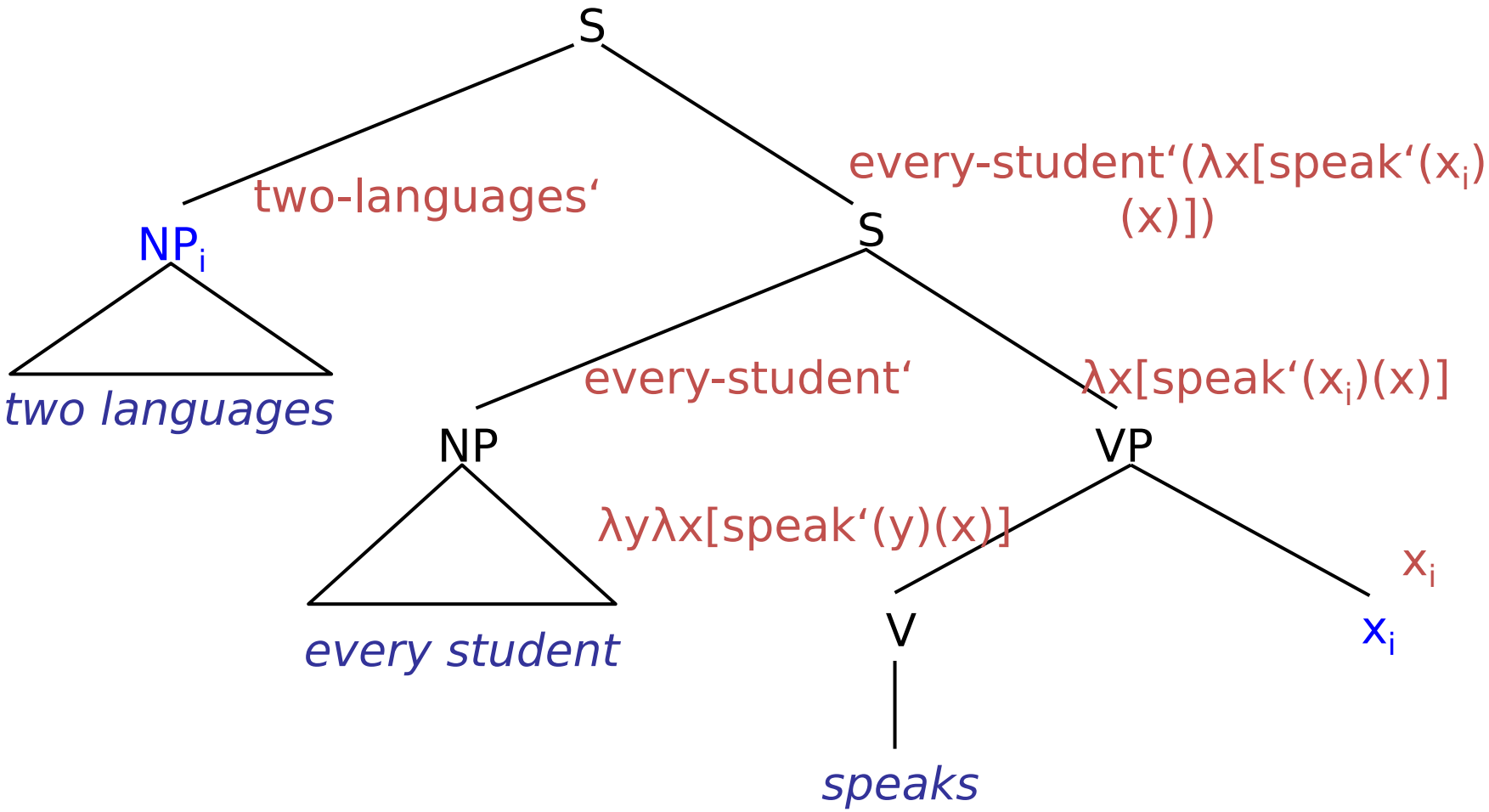
Topicalized Noun Phrases



Topicalized Noun Phrases



Topicalized NP: Semantic Construction



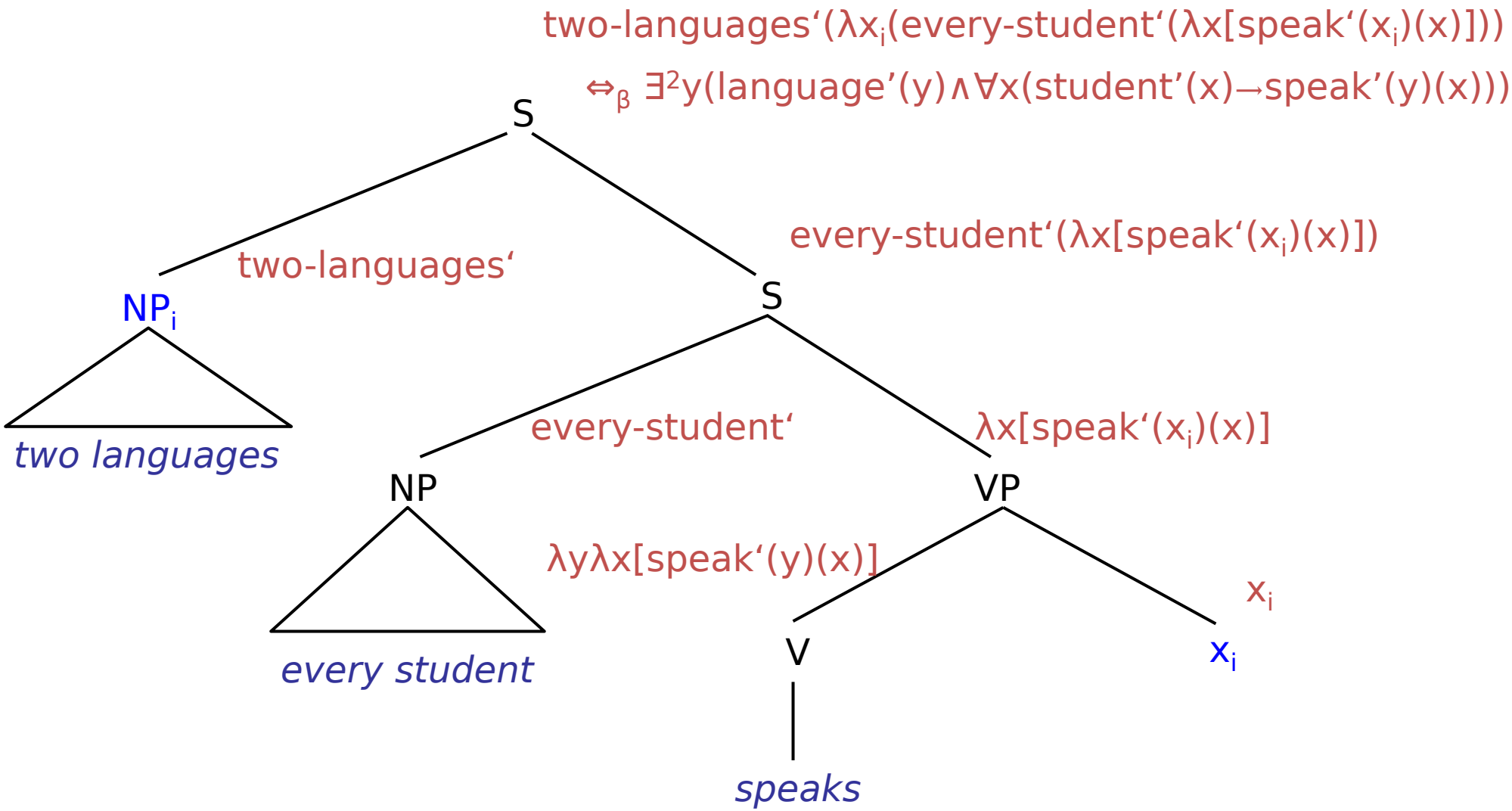
every-student‘ :⇔ λ*F*∀x(student’(x) → *F*(x))
two-languages’ :⇔ λP∃²y(language’(y) ∧ P(y))

„Indexed NP“ Rule

- If in a binary branching local syntactic structure
  - B and C are daughters of A,
  - B ⇒ β:⟨⟨e,t⟩,t⟩ is an NP with index i,
  - C ⇒γ:⟨e,t⟩,

  then A⇒ $\beta(\lambda x_i \gamma)$:t.

Semantic Construction



„Indexed NP“ Rule

- To provide a syntactic basis for a strictly compositional treatment of scope ambiguity, a syntactic rule of “Quantifier Raising” is assumed.
- Quantifier Raising looks precisely like topicalization. The „only“ difference is that the result cannot be seen at the linguistic surface, but is on the level of so-called “logical form”, which is input to semantic interpretation. (Logical forms are CFG trees, not logical expressions!)
- Interpretation is done in exactly the same way as in the topicalization case. Application of the “Indexed NP” rule is called “Quantifying In”.
- For several reasons, the logical form treatment of quantifier scope is not popular among computational linguists.

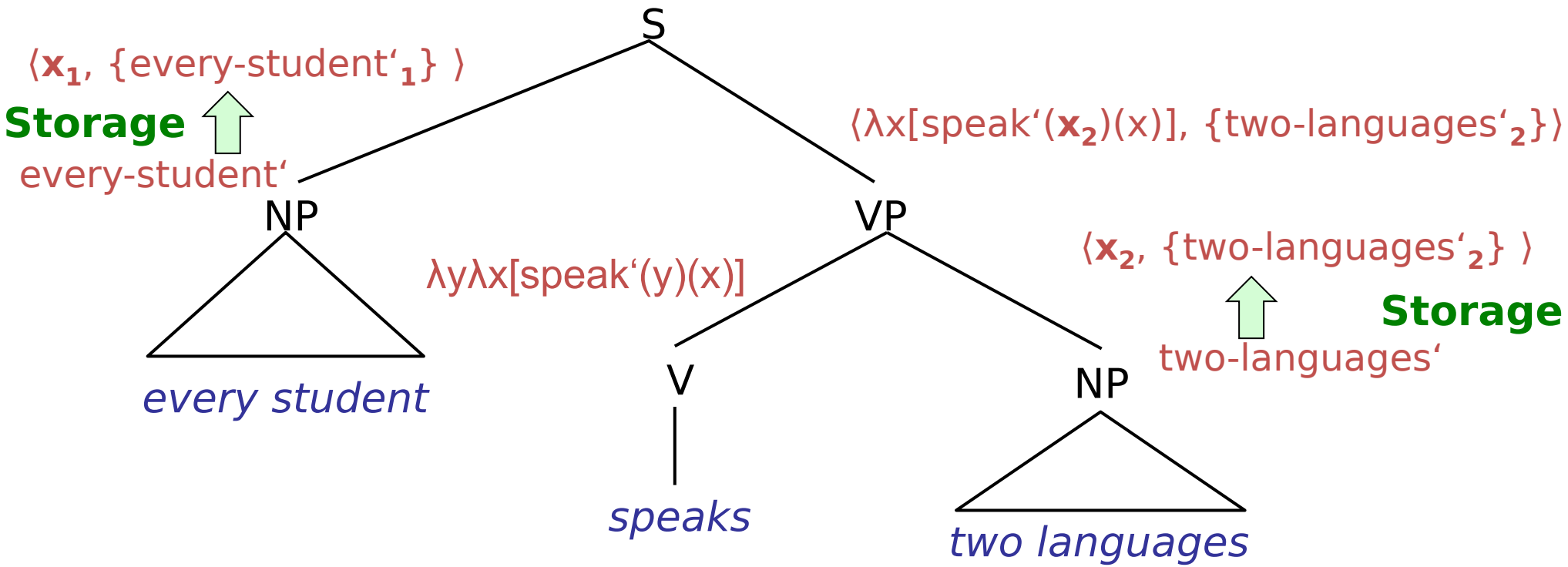
Quantifier Storage, Basic Idea

$\Leftrightarrow_\beta \exists^2 y(\text{language'}(y) \wedge \forall x\,(\text{student'}(x) \rightarrow \text{speak'}(y)(x)))$

Retrieval $\langle$two-languages' **(**$\boldsymbol{\lambda x_2}$[every-student'(λx_1[speak'(**$\mathbf{x_2}$**)(x_1)])]**)**, $\emptyset\rangle$

Retrieval $\langle$every-student'**(**$\boldsymbol{\lambda x_1}$[speak'(x_2)(**$\mathbf{x_1}$**)]**)**, {two-languages'$_2$}$\rangle$

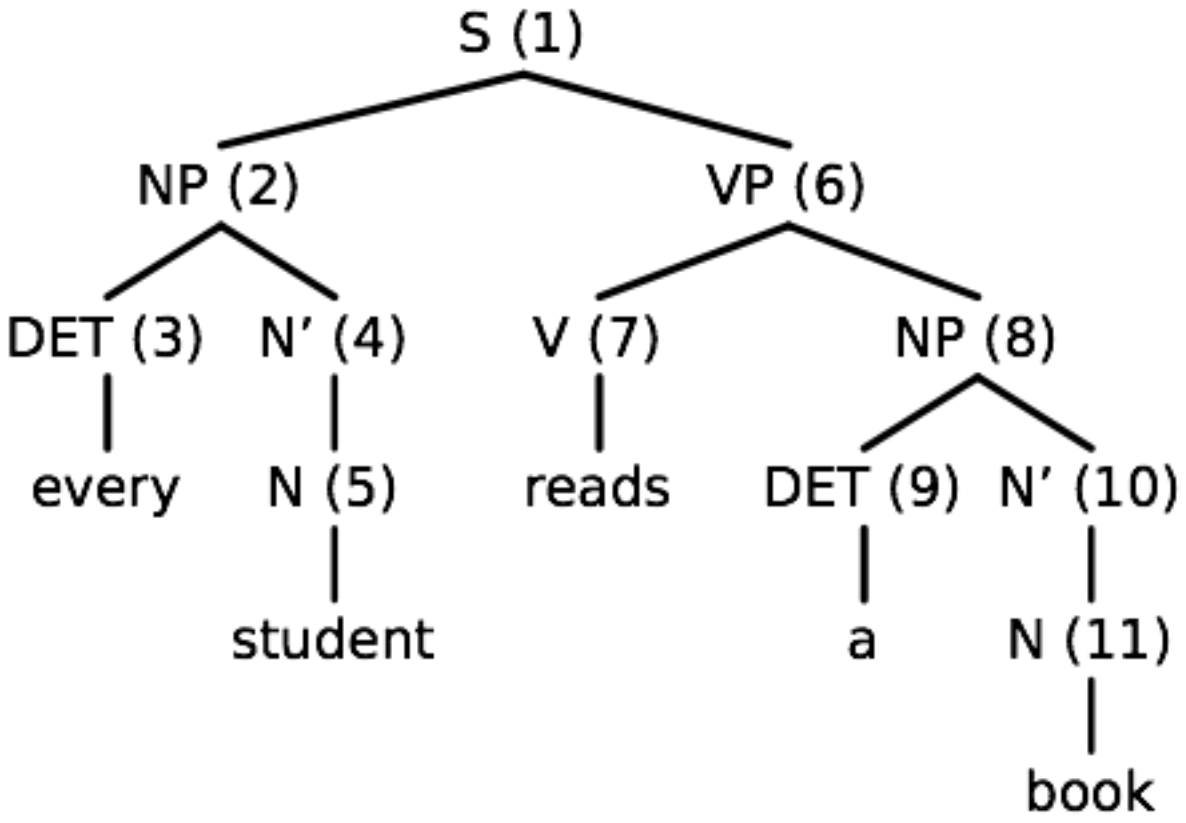
$\langle$speak'(x_2)(**$\mathbf{x_1}$**), {two-languages'$_2$, every-student'**$_1$**}$\rangle$



Quantifier Storage: Basic Idea

- NL quantifier phrases (in general, NPs) need not be directly applied, but can be moved to a quantifier store for later application (**storage**), passed up the syntax tree during the composition process, and removed from the store and applied at a sentence node (**retrieval**).
- Natural language expressions of type τ are in general assigned ordered pairs **$\langle \alpha, \Delta \rangle$** as a semantic values, consisting of
  - a **content $\alpha \in We_{\tau}$**, and
  - a **quantifier store $\Delta \subseteq WE_{\langle\langle e,t\rangle,t\rangle}$**
- A syntactic representation can have more than one semantic value. So, in general, the translation $A \Rightarrow \alpha$ reads: “α is among the semantic values of A”.
- A term α counts as a semantic representation for a sentence if we can derive $\langle \alpha, \emptyset \rangle$ as its semantic value.

Every student reads a book

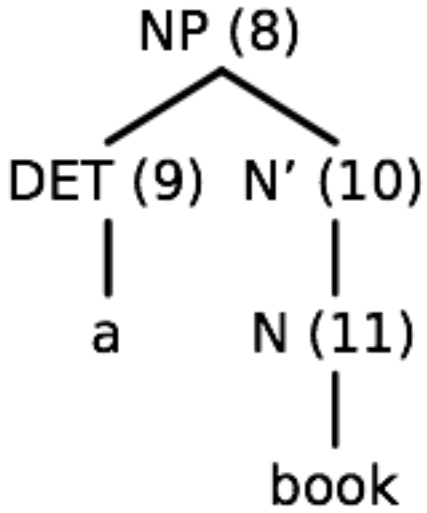


Basic Semantic Construction Rules, Revised

- Let in a binary branching local tree, B and C be (non-NP) daughters of A
  - if B ⇒ ⟨α, Δ⟩, α ∈ $WE_{\langle\sigma,\tau\rangle}$
  - and C ⇒ ⟨β, Γ⟩, β ∈ WE_{σ}
  - then A ⇒ ⟨α(β), Δ ∪ Γ⟩
- Let in a unary branching tree B be daughter of A
  - if B ⇒ ⟨α, Δ⟩
  - then A ⇒ ⟨α, Δ⟩
- Let w be lexical expression with mother A
  - A ⇒ ⟨α, ∅⟩, where α the lexical-semantic entry of w

Every student reads a book

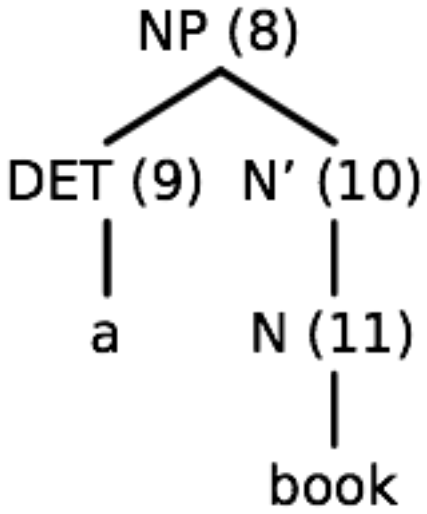
- (9) ⟨λFλG∃x(F(x) ∧ G(x)), ∅⟩
- (11) ⟨book’, ∅⟩
- (10) ⟨book’, ∅⟩
- (8) ⟨λFλG∃x(F(x) ∧ G(x))(book’), ∅⟩
- $\Leftrightarrow_{\beta}$ ⟨λG∃x(book’(x) ∧ G(x)), ∅⟩



Every student reads … (cont'd)

- (8) ⟨λGЭx(book'(x) ∧ G(x)), ∅⟩

  ⇒$_S$ ⟨λP.P(**x$_1$**), {**[**λGЭx(book'(x) ∧ G(x))**]$_1$**}⟩

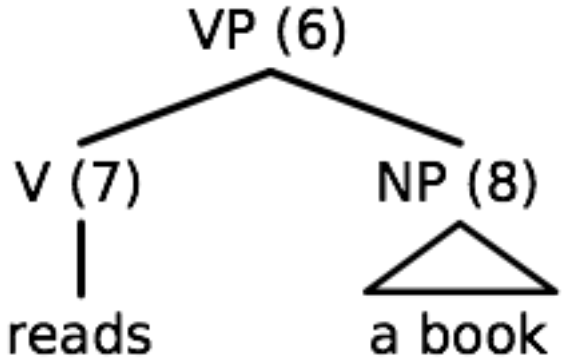


Semantic Construction [2/3]

- **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.
- Effect of the storage operation: The node is assigned another semantic value, where the content of the original value is moved to the store, and the new content is a placeholder (type-raised individual variable) of type $\langle\langle e,t\rangle,t\rangle$.
- Note: The NP node (8) of the example now has two semantic values connected with it (represented by the two lines above). Both values can be used in the further composition process: The quantifier can be applied either directly, **“in situ”** (first line), or **“quantified in”** via storage and retrieval (second line).

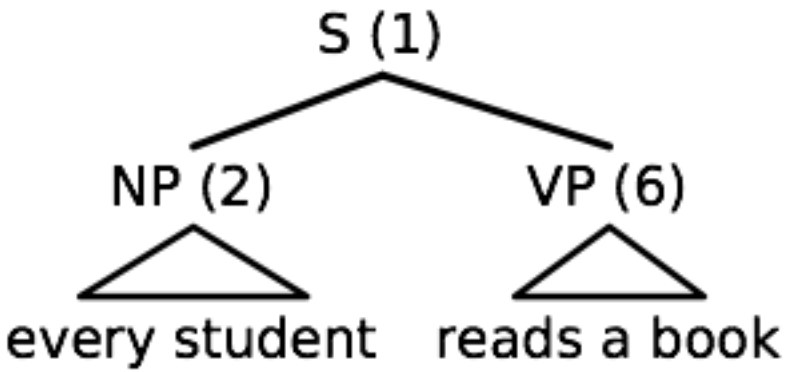
Every student reads … (cont'd)

- (8) ⟨λP.P(**x_1**), {**[**λGΞx(book'(x) ∧ G(x))**]$_1$**}⟩
- (7) ⟨λQλx(Q(λy(read*(y)(x)))), ∅⟩
- (6) ⟨**λQ**λx(**Q**(λy(read*(y)(x))))**(λP.P(**x_1**))**, {**[**λGΞx(…)**]$_1$**}⟩
- ⇔$_\beta$ ⟨λx(**λP**(**P**(**x_1**))**(λy(read*(y)(x)))**), {**[**λGΞx(…)**]$_1$**}⟩
- ⇔$_\beta$ ⟨λx(**λy**(read*(**y**)(x))**(x_1)**), {**[**λGΞx(…)**]$_1$**}⟩
- ⇔$_\beta$ ⟨λx(read*(**x_1**)(x)), {**[**λGΞx(…)**]$_1$**}⟩



Every student reads … (cont'd)

- (6) ⟨λx(read*(**x_1**)(x)), {**[**λGΞx(book'(x) ∧ G(x))**]$_1$**}⟩
- (2) ⟨λGΑy(student'(y) → G(y)), ∅⟩
- (1)⟨**λG**Αy(student'(y) → **G**(y))**(λx(read*(x_1)(x)))**, {**[…]$_1$**}⟩
- $\Leftrightarrow_\beta$ ⟨Αy(student'(y) → **λx**(read*(x_1)(**x**))**(y)**), {**[…]$_1$**}⟩
- $\Leftrightarrow_\beta$ ⟨Αy(student'(y) → read*(**x_1**)(y)), {**[…]$_1$**}⟩

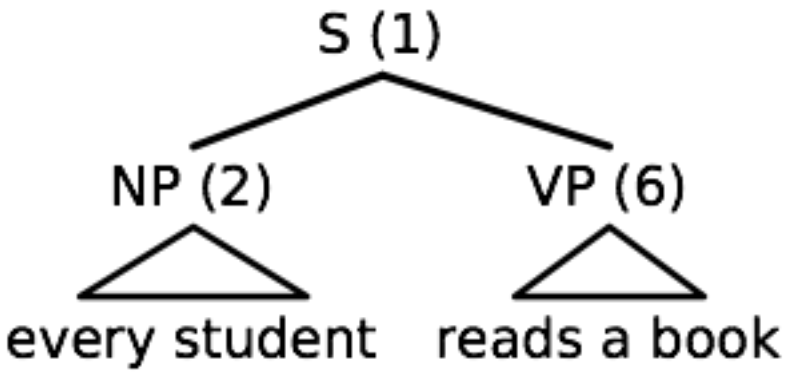


Semantic Construction [3/3]

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

Every student reads … (cont'd)

- (1) $\langle \forall y(\text{student}'(y) \rightarrow \text{read}^*(\mathbf{x_1})(y)), \{[\lambda G \exists x(\ldots)]_1\} \rangle$
- $\Rightarrow_R \langle \lambda G \exists x(\text{book}'(x) \wedge G(x))(\boldsymbol{\lambda x_1}(\forall y(\ldots \mathbf{x_1} \ldots))), \varnothing \rangle$
- $\Leftrightarrow_\beta \langle \exists x(\text{book}'(x) \wedge \boldsymbol{\lambda x_1}(\forall y(\ldots \mathbf{x_1} \ldots))(x)), \varnothing \rangle$
- $\Leftrightarrow_\beta \langle \exists x(\text{book}'(x) \wedge \forall y(\text{student}'(y) \rightarrow \text{read}^*(x)(y))), \varnothing \rangle$



Quantifier Storage

- Semantic construction using quantifier storage is a non-deterministic process: Choice points are storage (yes/no?) and retrieval (which NP?).
- To obtain the set of all possible sentence representations, we need an exhaustive search through all options, which is highly inefficient.
- Underspecification formalisms enable the compact representation and efficient computation of alternative scope readings. (➔ Computational Linguistics Course)