

Semantic Theory

Lecture 4: Transitive Verbs & Cooper Storage

Manfred Pinkal & Stefan Thater
 FR 4.7 Allgemeine Linguistik (Computerlinguistik)
 Universität des Saarlandes

Summer 2011

Semantics Sonstruction (recap)

- **Semantic lexicon**
 - maps words to semantic representations (type theory)
 - Note: all semantic representations should have a uniform type (e.g., all determiners have type $\langle\langle e, t \rangle, \langle\langle e, t \rangle, t\rangle\rangle$)
- **Semantics construction rules**
 - tell for each syntactic rule $X \rightarrow Y_1 Y_2$ how to combine the semantic representations of $Y_1 Y_2$ to obtain a semantic representation for X
 - we assume here that there is only a single operation to combine meaning representation: functional application

2

Semantics Sonstruction (recap)

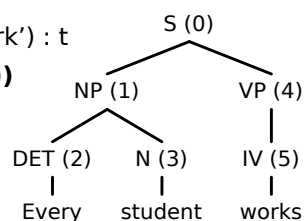
(2) $\mapsto \lambda P \lambda Q \forall x (P(x) \rightarrow Q(x)) : \langle\langle e, t \rangle, \langle\langle e, t \rangle, t\rangle\rangle$

(3) $\mapsto \text{student}' : \langle e, t \rangle$

(1) $\mapsto \lambda P \lambda Q \forall x (P(x) \rightarrow Q(x)) (\text{student}') : \langle\langle e, t \rangle, t\rangle$
 $\Rightarrow_{\beta} \lambda Q \forall x (\text{student}'(x) \rightarrow Q(x))$

(4) = (5) $\mapsto \text{work}' : \langle e, t \rangle$

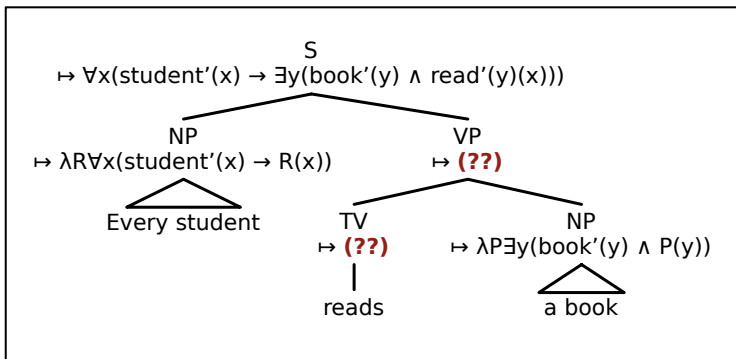
(0) $\mapsto \lambda Q \forall x (\text{student}'(x) \rightarrow Q(x)) (\text{work}') : t$
 $\Rightarrow_{\beta} \forall x (\text{student}'(x) \rightarrow \text{work}'(x))$



3

Transitive Verbs

- Every student reads a book
 - $\forall x(\text{student}'(x) \rightarrow \exists y(\text{book}'(y) \wedge \text{read}'(y)(x)))$



4

Transitive Verbs (1st attempt)

- $\text{read} \mapsto \text{read}' \in WE_{(((e,t), t), (e, t))}$
- $\text{read a book} \mapsto \text{read}'(\lambda P \exists y(\text{book}'(y) \wedge P(y))) \in WE_{(e, t)}$
- every student reads a book*
 - $\mapsto \lambda R \forall x(\text{student}'(x) \rightarrow R(x))(\text{read}'(\lambda P \exists y(\text{book}'(y) \wedge P(y))))$
 - $\Leftrightarrow \forall x(\text{student}'(x) \rightarrow \text{read}'(\lambda P \exists y(\text{book}'(y) \wedge P(y)))(x))$
- Problem:**
without an additional meaning postulate the formula does not capture the truth-conditions of the sentence.

5

Transitive Verbs (final version)

- Solution:**
 - use a more explicit λ -term for transitive verbs
- $\text{read} \mapsto \lambda Q \lambda z Q(\lambda x(\text{read}'(x)(z))) \in WE_{(((e,t), t), (e, t))}$
 - Note: $\text{read}' \in WE_{(e, (e, t))}$
- read a book*
 - $\mapsto \lambda Q \lambda z Q(\lambda x(\text{read}'(x)(z)))(\lambda P \exists y(\text{book}'(y) \wedge P(y)))$
 - $\Leftrightarrow_{\beta} \lambda z(\lambda P \exists y(\text{book}'(y) \wedge P(y))(\lambda x(\text{read}'(x)(z))))$
 - $\Leftrightarrow_{\beta} \lambda z(\exists y(\text{book}'(y) \wedge \lambda x(\text{read}'(x)(z))(y)))$
 - $\Leftrightarrow_{\beta} \lambda z(\exists y(\text{book}'(y) \wedge \text{read}'(y)(z)))$

6

Transitive Verbs (final version)

■ Solution:

- use a more explicit λ -term for transitive verbs
- *read a book*
 - $\mapsto \lambda z \exists y (\text{book}'(y) \wedge \text{read}'(y)(z))$
- *every student*
 - $\mapsto \lambda R \forall x (\text{student}'(x) \rightarrow R(x))$
- *every student reads a book*
 - $\mapsto \lambda R \forall x (\text{student}'(x) \rightarrow R(x)) (\lambda z \exists y (\text{book}'(y) \wedge \text{read}'(y)(z)))$
 - $\Leftrightarrow_{\beta} \forall x (\text{student}'(x) \rightarrow \lambda z \exists y (\text{book}'(y) \wedge \text{read}'(y)(z))(x))$
 - $\Leftrightarrow_{\beta} \forall x (\text{student}'(x) \rightarrow \exists y (\text{book}'(y) \wedge \text{read}'(y)(x)))$

7

Scope Ambiguities

- (1) *Every student reads a book*
 - a. $\forall x (\text{student}'(x) \rightarrow \exists y (\text{book}'(y) \wedge \text{read}^*(y)(x)))$
 - b. $\exists y (\text{book}'(y) \wedge \forall x (\text{student}'(x) \rightarrow \text{read}^*(y)(x)))$
- (2) *Every student didn't pay attention*
 - a. $\forall x (\text{student}'(x) \rightarrow \neg \text{pay-attention}'(x))$
 - b. $\neg \forall x (\text{student}'(x) \rightarrow \text{pay-attention}'(x))$
- (3) *Some inhabitant of every midwestern city participated*
- (4) *An American flag stood in front of every building*
- (5) *John searches a good book about semantics*
- (6) *Pola wants to marry a millionaire*

8

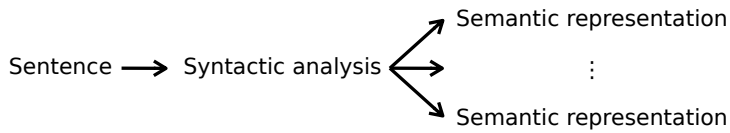
Scope Ambiguities

- Using the semantics construction rules from the previous lecture, we can derive only one reading for sentences exhibiting a scope ambiguity.
 - Assumption: the sentence has a unique syntactic structure.
- Quantifier scope is not determined by the syntactic position in which the corresponding NP occurs.
- Mismatch between syntactic and semantic structure is a challenge for compositional semantics construction.

9

Cooper Storage

- **Cooper-Storage** is a technique to derive different readings of sentences exhibiting a scope ambiguity
- The different readings are derived by using a **single, surface-based syntactic structure**



Cooper Storage

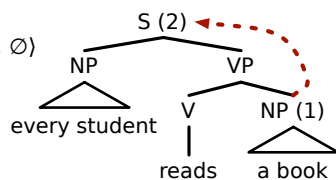
- Natural language expressions are assigned ordered pairs $\langle \alpha, \Delta \rangle$ as semantic values:
 - $\alpha \in WE_\tau$ is the content
 - $\Delta \subseteq WE_{\langle (e,t), t \rangle}$ is the quantifier store
- Quantifiers (NPs) can either apply *in situ*, or they can be moved to the store for later application (“storage”).
- At sentence nodes, quantifiers can be removed from the store and applied to the content (“retrieval”).
- A term α counts as a semantic representation for a sentence if we can derive $\langle \alpha, \emptyset \rangle$ as its semantic value.

The basic idea

- **Storage at (1)**
 $\langle \lambda G \exists x (bk(x) \wedge G(x)), \emptyset \rangle \Rightarrow$
 $\langle \lambda F.F(x_1), \{ [\lambda G \exists x (bk(x) \wedge G(x))]_1 \} \rangle$
- **Retrieval at (2)**
 $\langle \forall y (st(y) \rightarrow rd(x_1)(y)), \{ [\lambda G \exists x (bk(x) \wedge G(x))]_1 \} \rangle \Rightarrow$
 $\langle \lambda G \exists x (bk(x) \wedge G(x)) (\lambda x_1 (\forall y (st(y) \rightarrow rd(x_1)(y))), \emptyset) \rangle$

■ **After β -reduction:**

$\langle \exists x (bk(x) \wedge \forall y (st(y) \rightarrow rd(x)(y))), \emptyset \rangle$



Sample Grammar

$S \rightarrow NP VP$ $PN \rightarrow Bill \mid John \mid \dots$
 $NP \rightarrow DET N'$ $DET \rightarrow every \mid the \mid \dots$
 $NP \rightarrow PN$ $N \rightarrow student \mid book \mid \dots$
 $N' \rightarrow N$ $P \rightarrow of$
 $N' \rightarrow N PP$ $TV \rightarrow likes \mid reads \mid \dots$
 $VP \rightarrow IV$ $IV \rightarrow works \mid sleeps \mid \dots$
 $VP \rightarrow TV NP$
 $PP \rightarrow P NP$

13

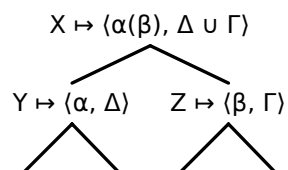
Semantic Lexicon

$Bill \mapsto \lambda F(F(b^*))$ $\in WE_{((e,t),t)}$
 $every \mapsto \lambda F \lambda G \forall x(F(x) \rightarrow G(x))$ $\in WE_{((e,t),((e,t),t))}$
 $a \mapsto \lambda F \lambda G \exists x(F(x) \wedge G(x))$ $\in WE_{((e,t),((e,t),t))}$
 $sleeps \mapsto sleep'$ $\in WE_{(e,t)}$
 $student \mapsto student'$ $\in WE_{(e,t)}$
 $reads \mapsto \lambda Q \lambda x(Q(\lambda y(read^*(y)(x))))$ $\in WE_{(((e,t),t), (e,t))}$
 $of \mapsto [\Rightarrow exercise]$ $\in WE_{(((e,t),t), ((e,t), (e,t)))}$

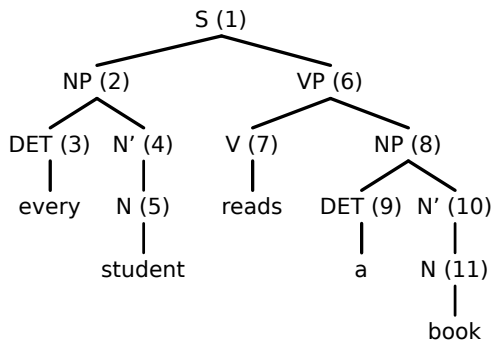
14

Semantic Construction [1/3]

- $X \rightarrow Y Z$ or $X \rightarrow Z Y$
 - if $Y \mapsto \langle \alpha, \Delta \rangle, \alpha \in WE_{(\sigma, \tau)}$
 - and $Z \mapsto \langle \beta, \Gamma \rangle, \beta \in WE_{\sigma}$
 - then $X \mapsto \langle \alpha(\beta), \Delta \cup \Gamma \rangle$
- $X \rightarrow Y$
 - if $Y \mapsto \langle \alpha, \Delta \rangle$
 - then $X \mapsto \langle \alpha, \Delta \rangle$
- $X \rightarrow w$
 - $X \mapsto \langle \alpha, \emptyset \rangle$, where $\alpha = \text{SemLex}(w)$



15

Every student reads a book

16

Every student reads a book

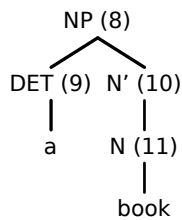
→ (9) $\langle \lambda F \lambda G \exists x (F(x) \wedge G(x)), \emptyset \rangle$

(11) $\langle \text{book}', \emptyset \rangle$

(10) $\langle \text{book}', \emptyset \rangle$

(8) $\langle \lambda F \lambda G \exists x (F(x) \wedge G(x))(\text{book}'), \emptyset \rangle$

$\Leftrightarrow_{\beta} \langle \lambda G \exists x (\text{book}'(x) \wedge G(x)), \emptyset \rangle$

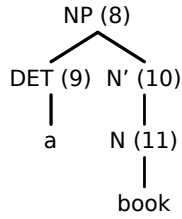
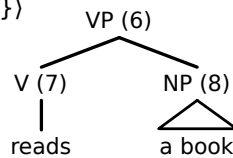
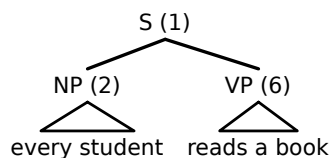


17

Semantic Construction [2/3]

- **Storage:** $\langle Q, \Delta \rangle \Rightarrow_S \langle \lambda P.P(x_i), \Delta \cup \{[Q]_i\} \rangle$
 - if A is a noun phrase whose semantic value is $\langle Q, \Delta \rangle$, then $\langle \lambda P.P(x_i), \Delta \cup \{[Q]_i\} \rangle$ is also a semantic value for A, where $i \in N$ is a new index.
 - The original content is moved to the store.
 - The new content is a placeholder of type $\langle (e,t), t \rangle$
- **Note:** by using this rule, we can assign more than one semantic value to a noun phrase.

18

Every student reads ... (cont'd)(9) $\langle \lambda F \lambda G \exists x (F(x) \wedge G(x)), \emptyset \rangle$ (10) $\langle \text{book}', \emptyset \rangle$ (11) $\langle \text{book}', \emptyset \rangle$ (8) $\langle \lambda F \lambda G \exists x (F(x) \wedge G(x))(\text{book}'), \emptyset \rangle$ $\Leftrightarrow_{\beta} \langle \lambda G \exists x (\text{book}'(x) \wedge G(x)), \emptyset \rangle$ $\rightarrow \Leftrightarrow_s \langle \lambda P.P(\mathbf{x}_1), \{[\lambda G \exists x (\text{book}'(x) \wedge G(x))]\mathbf{1}\}_1 \rangle$ **Every student reads ... (cont'd)** \rightarrow (8) $\langle \lambda P.P(\mathbf{x}_1), \{[\lambda G \exists x (\text{book}'(x) \wedge G(x))]\mathbf{1}\}_1 \rangle$ (7) $\langle \lambda Q \lambda x (Q(\lambda y (\text{read}^*(y)(x)))) \rangle, \emptyset \rangle$ (6) $\langle \lambda Q \lambda x (Q(\lambda y (\text{read}^*(y)(x))))(\lambda P.P(\mathbf{x}_1)), \{[\lambda G \exists x (\dots)]\mathbf{1}\}_1 \rangle$ $\Leftrightarrow_{\beta} \langle \lambda x (\lambda P(P(\mathbf{x}_1))(\lambda y (\text{read}^*(y)(x)))) \rangle, \{[\lambda G \exists x (\dots)]\mathbf{1}\}_1 \rangle$ $\Leftrightarrow_{\beta} \langle \lambda x (\lambda y (\text{read}^*(y)(x))(\mathbf{x}_1)) \rangle, \{[\lambda G \exists x (\dots)]\mathbf{1}\}_1 \rangle$ $\Leftrightarrow_{\beta} \langle \lambda x (\text{read}^*(\mathbf{x}_1)(x)) \rangle, \{[\lambda G \exists x (\dots)]\mathbf{1}\}_1 \rangle$ **Every student reads ... (cont'd)** \rightarrow (6) $\langle \lambda x (\text{read}^*(\mathbf{x}_1)(x)) \rangle, \{[\lambda G \exists x (\text{book}'(x) \wedge G(x))]\mathbf{1}\}_1 \rangle$ (2) $\langle \lambda G \forall y (\text{student}'(y) \rightarrow G(y)), \emptyset \rangle$ (1) $\langle \lambda G \forall y (\text{student}'(y) \rightarrow G(y))(\lambda x (\text{read}^*(\mathbf{x}_1)(x))) \rangle, \{[\dots]\mathbf{1}\}_1 \rangle$ $\Leftrightarrow_{\beta} \langle \forall y (\text{student}'(y) \rightarrow \lambda x (\text{read}^*(\mathbf{x}_1)(x))(y)) \rangle, \{[\dots]\mathbf{1}\}_1 \rangle$ $\Leftrightarrow_{\beta} \langle \forall y (\text{student}'(y) \rightarrow \text{read}^*(\mathbf{x}_1)(y)) \rangle, \{[\dots]\mathbf{1}\}_1 \rangle$ 

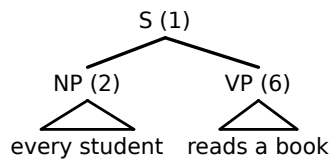
Semantic Construction [3/3]

- **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 $\langle \alpha, \Delta \cup \{[Q]_i\} \rangle$, then $\langle Q(\lambda x_i \alpha), \Delta \rangle$ is also a semantic value for A.
 - Notation: read “ \cup ” as “disjoint union”

22

Every student reads ... (cont'd)

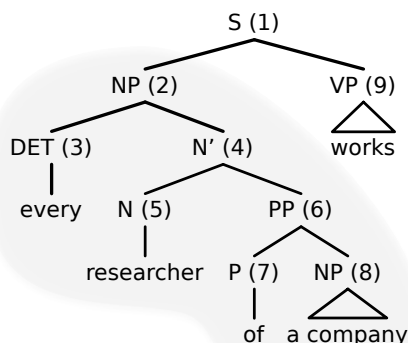
- (1) $\langle \forall y(\text{student}'(y) \rightarrow \text{read}^*(x_1)(y)), \{[\lambda G \exists x(\dots)]_1\} \rangle$
 $\Rightarrow_R \langle \lambda G \exists x(\text{book}'(x) \wedge G(x))(\lambda x_1(\forall y(\dots x_1 \dots))), \emptyset \rangle$
 $\Leftrightarrow_\beta \langle \exists x(\text{book}'(x)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



23

Problem: Nested noun phrases

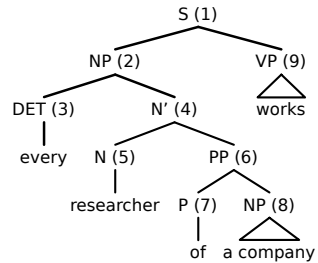
- (1) *Every researcher of a company works*



24

Problem: Nested noun phrases

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



25

Problem: Nested noun phrases

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

Not a reading! Variable x_1 occurs free!

26

Problem: Nested noun phrases

- The unstructured store does not reflect the dependencies between quantifiers in complex noun phrases like „every [researcher of a company]“
- \Rightarrow quantifiers can be retrieved in any order!

- $\langle \text{work}(x_2), \{[\lambda G\forall y(\dots x_1 \dots)]\}_2, [\lambda G\exists x(\dots)]\}_1 \}$
 - **We want:** Q_1 cannot be retrieved if Q_2 is still on the store

27

Nested Cooper Storage

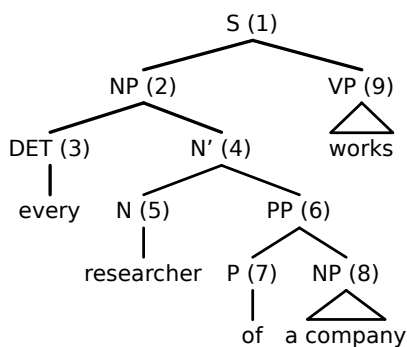
- **Storage:** $\langle Q, \Delta \rangle \Rightarrow_S \langle \lambda P.P(x_i), \{\langle Q, \Delta \rangle_i\} \rangle$
 - If A is a noun phrase whose semantic value is $\langle Q, \Delta \rangle$, then $\langle \lambda P.P(x_i), \{\langle Q, \Delta \rangle_i\} \rangle$ is also a semantic value for A, where $i \in \mathbb{N}$ is a new index.
- The original semantic value **including its store** is moved to the store.

Nested Cooper Storage

- **Retrieval:** $\langle \alpha, \Delta \cup \{\langle Q, \Gamma \rangle_i\} \rangle \Rightarrow \langle Q(\lambda x_i. \alpha), \Delta \cup \Gamma \rangle$
 - If A is a sentence with semantic value $\langle \alpha, \Delta \cup \{\langle Q, \Gamma \rangle_i\} \rangle$, then $\langle Q(\lambda x_i. \alpha), \Delta \cup \Gamma \rangle$ is also a semantic value of the sentence.
- \Rightarrow nested stores are **not accessible** for retrieval

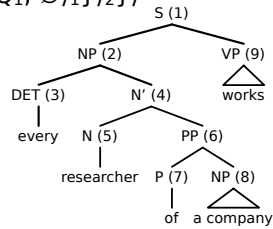
Nested Cooper-Storage

Every reasearcher of a ...



Every reasearcher of a ...

- (8) $\langle \lambda G \exists x(\text{comp}(x) \wedge G(x)), \emptyset \rangle$
 $\Rightarrow_S \langle \lambda F.F(x_1), \{\{Q_1 = \lambda G(\exists x(\text{comp}(x) \wedge G(x)), \emptyset)\}_1\} \rangle$
- (4) $\langle \lambda y(\text{res}(y) \wedge \text{of}(x_1)(y)), \{\{Q_1, \emptyset\}_1\} \rangle$
- (2) $\langle \lambda G \forall z((\text{res}(z) \wedge \text{of}(x_1)(z)) \rightarrow G(z)), \{\{Q_1, \emptyset\}_1\}_2 \rangle$
 $\Rightarrow_S \langle \lambda F.F(x_2), \{\{Q_2 = \lambda G \forall z(\dots), \{\{Q_1, \emptyset\}_1\}_2\} \rangle$
- (9) $\langle \text{work}, \emptyset \rangle$
- (1) $\langle \text{work}(x_2), \{\{Q_2, \{\{Q_1, \emptyset\}_1\}_2\} \rangle$



31

Every reasearcher of a ...

- $\langle \text{work}(x_2), \{\{Q_2, \{\{Q_1, \emptyset\}_1\}_2\} \rangle$
 $\Rightarrow_R \langle Q_2 (\lambda x_2.\text{work}(x_2)), \{\{Q_1, \emptyset\}_1\} \rangle$
- $\Leftrightarrow_\beta \langle \forall z((\text{res}(z) \wedge \text{of}(x_1)(z)) \rightarrow \text{work}(z)), \{\{Q_1, \emptyset\}_1\} \rangle$
- $\Rightarrow_R \langle Q_1(\lambda x_1.\forall z((\text{res}(z) \wedge \text{of}(x_1)(z)) \rightarrow \text{work}(z))), \emptyset \rangle$
- $\Leftrightarrow_\beta \langle \exists x(\text{comp}(x) \wedge \forall z((\text{res}(z) \wedge \text{of}(x)(z)) \rightarrow \text{work}(z))), \emptyset \rangle$

32

Every reasearcher of a ...

- $\langle \text{work}(x_2), \{\{\lambda G \forall z(\dots), \{\lambda G \exists x(\dots), \emptyset\}_1\}_2\} \rangle$
 $\Rightarrow_R^* \langle \exists x(\text{comp}(x) \wedge \forall z((\text{res}(z) \wedge \text{of}(x)(z)) \rightarrow \text{work}(z))), \emptyset \rangle$

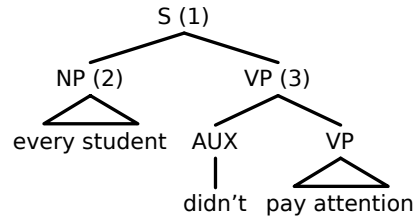
No other reading can be derived!

- But how do we derive the “direct scope” reading?
- Simple answer: don’t store, apply quantifiers “in situ”

33

Can we derive all readings?

- Storing a quantifier means to “move it upwards” in the syntax tree (roughly speaking).
- *Every student did not pay attention*
 - “Every student” is higher in the tree than the negation
 - ⇒ the negation cannot take scope over “every student”



34

(see Ruys & Winter, 2008)

Some restrictions on scope

- (1) *Some inhabitant of every midwestern city participated*
 - two readings: (a) direct scope and (b) every \lt^* some
- (2) *Someone who inhabits every midwestern city participated*
 - only the direct scope reading available

Finite clauses can create “scope islands”

- Quantifiers must take scope within such clauses

35

(see Ruys & Winter, 2008)

Some restrictions on scope

- (1) *You will inherit a fortune if every man dies*
 - “every man” cannot take scope over complete sentence
- (2) *If a friend of mine from Texas had died in a fire, I would have inherited a fortune* (Fodor & Sag 1982)
 - “a friend of mine from Texas” can take wide scope

Finite clauses can create “scope islands”

- Quantifiers must take scope within such clauses
- Indefinites can “escape” scope islands

36

Compositionality

- **Denotations** (“D-compositionality”)
The denotation of a complex expression is a function of the denotations its parts.
- **Semantic representations** (“S-compositionality”)
The semantic representation of a complex expression is a function of the semantic representations of its parts.

37

Compositionality

- Storage techniques are (up to non-determinism) compositional on the level of semantic representations.
- But are not compositional on the level of denotations: Semantic values (α , Δ) don't receive an interpretation.

38

Literature

- Patrick Blackburn, Johan Bos (2005): Representation and Inference for Natural Language. A First Course in Computational Semantics. CSLI Press.
- W. R. Keller (1988). Nested Cooper storage: The proper treatment of quantification in ordinary noun phrases. In Reyle, Rohrer (Ed.). Natural Language Parsing and Linguistic Theories
- E. G. Ruys, Yoad Winter (2008). Quantifier scope in formal linguistics. To appear in: Handbook of Philosophical Logic, 2nd Edition.

39