

# Semantic Theory

## Lecture 3 – Semantics Construction

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

Summer 2013

## First-order logic

- Formulas of first-order logic can talk about properties of and relations between individuals.
- Constants and variables denote individuals.
- Quantification is restricted to quantification over individuals.

2

## Limits of first-order logic

- First-order logic is not expressive enough to capture the full range of meaning of natural language:
  - Modification (“good student”, “former professor”)
  - Sentence embedding verbs (“knows that ...”)
  - Higher order quantification (“have the same hair color”)
  - ...
- First-order logic does not support compositional semantics construction.

3

## Limits of first-order logic

- **The principle of compositionality (recap):** *The meaning of a complex expression is a function of the meanings of its parts and of the syntactic rules by which they are combined* (cited from Partee & al., 1993)
- **Compositional semantics construction:**
  - compute meaning representations for sub-expressions.
  - combine them to obtain a meaning representation for a complex expression.
- $a\ man\ walks \mapsto \exists x(\text{man}'(x) \wedge \text{walk}'(x))$ 
  - $a\ man \mapsto (?)$
  - $walks \mapsto (?)$

4

## Type Theory

- The types of non-logical expressions provided by first-order logic are not sufficient to describe the semantic function of all natural language expressions.
- Type theory provides a much richer inventory of types: higher-order relations and functions of different kinds.

5

## Types

- **Basic types:**
  - **e** - the type of individual terms (“entities”)
  - **t** - the type of formulas (“truth-values”)
- **Complex types:**
  - If  $\sigma, \tau$  are types, then  $\langle \sigma, \tau \rangle$  is a type.
  - $\langle \sigma, \tau \rangle$  is the type of functions mapping arguments of type  $\sigma$  to values of type  $\tau$ .
- Types indicate, how many arguments a predicate has, and what types the arguments must have.

6

## Types of Predicate Logic

- Individual constants and variables:  $e$
- One-place predicates (sleep, walk, ...)
  - $(e, t)$
- Two-place predicates (read, admire, ...)
  - $(e, (e, t))$
- Three-place predicates (give, ...)
  - $(e, (e, (e, t)))$

7

## Type Theory - Vocabulary

- **Constants:** For every type  $\tau$  a possibly empty set of non-logical constants  $CON_\tau$  (pairwise disjoint)
- **Variables:** For every type  $\tau$  an infinite set of variables  $VAR_\tau$  (pairwise disjoint)
- **Logical symbols:**  $\forall, \exists, \wedge, \vee, \dots$
- **Brackets:**  $(, )$

8

## Type Theory - Syntax

- The sets of **well-formed expressions**  $WE_\tau$  for every type  $\tau$  are given by:
  - (i)  $CON_\tau \subseteq WE_\tau$  and  $VAR_\tau \subseteq WE_\tau$ , for every type  $\tau$
  - (ii) If  $\alpha$  is in  $WE_{(\sigma, \tau)}$ ,  $\beta$  in  $WE_\sigma$ , then  $\alpha(\beta) \in WE_\tau$ .
  - (iii) If  $A, B$  are in  $WE_t$ , then  $\neg A, (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B)$  are in  $WE_t$ .
  - (iv) If  $A$  is in  $WE_t$  and  $v$  is a variable of arbitrary type, then  $\forall v A$  and  $\exists v A$  are in  $WE_t$ .
  - (v) If  $\alpha, \beta$  are well-formed expressions of the same type, then  $\alpha = \beta \in WE_t$ .

9

## Type Theory – Semantics [1/3]

- Let  $U$  be a non-empty set of entities.
- The **domain of possible denotations**  $D_\tau$  for every type  $\tau$  is given by:
  - $D_e = U$
  - $D_t = \{0, 1\}$
  - $D_{\langle\sigma, \tau\rangle}$  is the set of all functions from  $D_\sigma$  to  $D_\tau$
- Expressions of type  $\tau$  denote elements of  $D_\tau$
- For instance
  - $\alpha \in WE_{\langle e, t \rangle}$  denotes a set of individual
  - $\alpha \in WE_{\langle\langle e, t \rangle, t\rangle}$  denotes a set of sets of individuals

10

## Characteristic Functions

- Many natural language expression have a type  $\langle\sigma, t\rangle$ .
- $\langle\sigma, t\rangle$  the type of functions mapping elements of type  $\sigma$  to true or false.
- Such function are also known as **characteristic functions**, and can be thought of as subsets of  $D_\sigma$ .
- Example: “student” is a constant of type  $\langle e, t \rangle$  and can be seen as characterising the set of students.

11

## Characteristic Functions

- $U = \{a, b, c, d\}$
- $X = \{a, b\}$
- Characteristic function  $f_X$  of  $X$  (over  $U$ ):
  - $f_X(a) = 1$
  - $f_X(b) = 1$
  - $f_X(c) = 0$
  - $f_X(d) = 0$
- More generally: For all  $a \in U$ ,  $f_X(a) = 1$  iff  $a \in X$

12

## Type Theory – Semantics [2/3]

- **A model structure** for a type theoretic language consists of a pair  $\mathbf{M} = \langle \mathbf{U}, \mathbf{V} \rangle$ , where
  - U is a non-empty domain of individuals
  - V is an interpretation function, which assigns to every member of  $\text{CON}_\tau$  an element of  $D_\tau$ .
- **Variable assignment** g assigns every variable of type  $\tau$  a member of  $D_\tau$

13

## Type Theory – Semantics [3/3]

- **Interpretation with respect to** a model structure  $\mathbf{M} = \langle \mathbf{U}, \mathbf{V} \rangle$  and a variable assignment  $\mathbf{g}$ :
  - $\llbracket \alpha \rrbracket^{\mathbf{M},\mathbf{g}} = V(\alpha)$ , if  $\alpha$  is a constant  
 $\llbracket \alpha \rrbracket^{\mathbf{M},\mathbf{g}} = g(\alpha)$ , if  $\alpha$  is a variable
  - $\llbracket \alpha(\beta) \rrbracket^{\mathbf{M},\mathbf{g}} = \llbracket \alpha \rrbracket^{\mathbf{M},\mathbf{g}}(\llbracket \beta \rrbracket^{\mathbf{M},\mathbf{g}})$
  - $\llbracket \neg\phi \rrbracket^{\mathbf{M},\mathbf{g}} = 1$  iff  $\llbracket \phi \rrbracket^{\mathbf{M},\mathbf{g}} = 0$   
 $\llbracket \phi \wedge \psi \rrbracket^{\mathbf{M},\mathbf{g}} = 1$  iff  $\llbracket \phi \rrbracket^{\mathbf{M},\mathbf{g}} = 1$  and  $\llbracket \psi \rrbracket^{\mathbf{M},\mathbf{g}} = 1$   
 $\llbracket \phi \vee \psi \rrbracket^{\mathbf{M},\mathbf{g}} = 1$  iff  $\llbracket \phi \rrbracket^{\mathbf{M},\mathbf{g}} = 1$  or  $\llbracket \psi \rrbracket^{\mathbf{M},\mathbf{g}} = 1$   
...
  - $\llbracket \alpha = \beta \rrbracket^{\mathbf{M},\mathbf{g}} = 1$  iff  $\llbracket \alpha \rrbracket^{\mathbf{M},\mathbf{g}} = \llbracket \beta \rrbracket^{\mathbf{M},\mathbf{g}}$

14

## Type Theory – Semantics [3/3]

- **Interpretation with respect to** a model structure  $\mathbf{M} = \langle \mathbf{U}, \mathbf{V} \rangle$  and a variable assignment  $\mathbf{g}$ :
  - $\llbracket \exists v\phi \rrbracket^{\mathbf{M},\mathbf{g}} = 1$  iff there is a  $d \in D_\tau$  such that  $\llbracket \phi \rrbracket^{\mathbf{M},\mathbf{g}[v/d]} = 1$   
 $\llbracket \forall v\phi \rrbracket^{\mathbf{M},\mathbf{g}} = 1$  iff for all  $d \in D_\tau$  :  $\llbracket \phi \rrbracket^{\mathbf{M},\mathbf{g}[v/d]} = 1$   
(where v is a variable of type  $\tau$ )

15

## Examples [ $\Rightarrow$ whiteboard]

- *Bill reads a book*
  - $\llbracket \exists x(\text{book}'(x) \wedge \text{read}'(x)(b^*)) \rrbracket^{M,g} = 1$  iff ...
- *Bill is a good student*
  - $\llbracket \text{good}'(\text{student}') (b^*) \rrbracket^{M,g} = 1$  iff ...

read' : (e, (e, t))	b* : e
book' : (e, t)	x : e
student' : (e, t)	
good' : ((e, t), (e, t))	

16

## Adjective Classes & Meaning Postulates

- **Natural language:**
  - Bill is a good student  $\models$  Bill is a student
- **Type theory:**
  - $\text{good}'(\text{student}') (b^*) \not\models \text{student}'(b^*)$
- We need additional “meaning postulates” to get the intended entailment relations
- Meaning postulates are restrictions on models and constrain the possible meaning of certain words

17

## Adjective Classes & Meaning Postulates

- **Intersective adjectives** (“blond”)
  - $\llbracket \text{blond } N \rrbracket = \llbracket \text{blond} \rrbracket \cap \llbracket N \rrbracket$
  - Meaning postulate:  $\forall G \forall x (\text{blond}(G)(x) \rightarrow (\text{blond}^*(x) \wedge G(x)))$
  - Note:  $\text{blond} \in \text{WE}_{((e, t), (e, t))}$ ,  $\text{blond}^* \in \text{WE}_{(e, t)}$
- **Subjective adjectives** (“good”)
  - $\llbracket \text{good } N \rrbracket \subseteq \llbracket N \rrbracket$
  - Meaning postulate:  $\forall G \forall x (\text{good}(G)(x) \rightarrow G(x))$
- **Privative adjectives** (“former”)
  - $\llbracket \text{former } N \rrbracket \cap \llbracket N \rrbracket = \emptyset$
  - Meaning postulate:  $\forall G \forall x (\text{former}(G)(x) \rightarrow \neg G(x))$

18

## Semantics Construction

- **The principle of compositionality (recap):** The meaning of a complex expression is a function of the meanings of its parts and of the syntactic rules by which they are combined (cited from Partee & al., 1993)
- **Compositional semantics construction:**
  - compute meaning representations for sub-expressions
  - combine them to obtain a meaning representation for a complex expression.

19

## A simple grammar

S → NP VP      PN → Bill  
NP → PN        PN → Mary  
VP → IV        IV → works  
VP → TV NP    TV → likes

- *Bill works*
- *Bill likes Mary*
- ...

20

## Semantic lexicon

- Bill ↦  $b^* : e$
- Mary ↦  $m^* : e$
- likes ↦  $like' : \langle e, \langle e, t \rangle \rangle$
- works ↦  $work' : \langle e, t \rangle$
  
- read “↦” as “translates into”

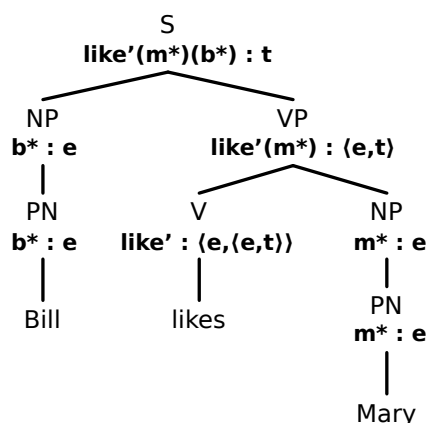
21

## Semantics Construction Rules (1<sup>st</sup> Version)

- **S → NP VP**  
if  $VP \mapsto \alpha'$  and  $NP \mapsto \beta'$ , then  $S \mapsto \alpha'(\beta')$
- **NP → PN**  
if  $PN \mapsto \alpha'$ , then  $NP \mapsto \alpha'$
- **VP → IV**  
if  $IV \mapsto \alpha'$ , then  $VP \mapsto \alpha'$
- **VP → TV NP**  
if  $TV \mapsto \alpha'$  and  $NP \mapsto \beta'$ , then  $VP \mapsto \alpha'(\beta')$

22

*Bill likes Mary*



23

## Noun Phrases

- John works**  $\mapsto$  work(j)
- Somebody works**  $\mapsto$   $\exists x(\text{work}(x))$
- Every student works**  $\mapsto$   $\forall x(\text{student}(x) \rightarrow \text{work}(x))$
- A student works**  $\mapsto$   $\exists x(\text{student}(x) \wedge \text{work}(x))$
- No student works**  $\mapsto$   $\neg \exists x(\text{student}(x) \wedge \text{work}(x))$
- John and Mary work**  $\mapsto$  work(j)  $\wedge$  work(m)

24

## $\lambda$ -Abstraction

- **$\lambda x(\text{drive}(x) \wedge \text{drink}(x))$** 
  - a term of type  $\langle e, t \rangle$
  - denotes the property (set of individuals) of being “an  $x$  such that  $x$  drives and drinks”
- $\lambda$ -abstraction is an operation that takes an expression and “opens” specific argument positions.
- The result of abstraction over individual variable  $x$  in the formula “ $\text{drive}(x) \wedge \text{drink}(x)$ ” results in the complex expression “ $\lambda x(\text{drive}(x) \wedge \text{drink}(x))$ .”

25

## Type Theory with $\lambda$ -Operator

- Syntax like basic type theory, plus:
  - **If  $\alpha$  is in  $WE_\tau$  and  $v$  is a variable of type  $\sigma$ , then  $\lambda v \alpha$  is a well-formed expression of type  $\langle \sigma, \tau \rangle$ .**
- The scope of the  $\lambda$ -operator is the smallest WE to its right. Wider scope must be indicated by brackets.
- We often use the “dot notation”  **$\lambda x. \dots$**  indicating that the  $\lambda$ -operator takes widest possible scope.

26

## $\lambda$ -Abstraction – Semantics

- If  $\alpha \in WE_\tau$ ,  $v \in VAR_\sigma$ , then  $\llbracket \lambda v \alpha \rrbracket^{M,g}$  is that function  $f : D_\sigma \rightarrow D_\tau$  such that for all  $a \in D_\sigma$ ,  $f(a) = \llbracket \alpha \rrbracket^{M,g[v/a]}$
- $\llbracket \lambda x(\text{drink}(x) \wedge \text{drive}(x)) \rrbracket^{M,g} = \dots$ 
  - [ $\Rightarrow$  whiteboard]

27

## λ-Abstraction – Semantics

- If  $\alpha \in WE_\tau$ ,  $v \in VAR_\sigma$ , then  $\llbracket \lambda v \alpha \rrbracket^{M,g}$  is that function  $f : D_\sigma \rightarrow D_\tau$  such that for all  $a \in D_\sigma$ ,  $f(a) = \llbracket \alpha \rrbracket^{M,g[v/a]}$
- If the  $\lambda$ -expression is applied to some argument, we can simplify the interpretation:
  - $\llbracket \lambda v \alpha \rrbracket^{M,g}(A) = \llbracket \alpha \rrbracket^{M,g[v/A]}$
- $\llbracket \lambda x(\text{drink}(x) \wedge \text{drive}(x))(b^*) \rrbracket^{M,g} = 1$ 
  - iff  $\llbracket \lambda x(\text{drink}(x) \wedge \text{drive}(x)) \rrbracket^{M,g}(\llbracket b^* \rrbracket^{M,g}) = 1$
  - iff  $\llbracket \lambda x(\text{drink}(x) \wedge \text{drive}(x)) \rrbracket^{M,g}(V_M(b^*)) = 1$
  - iff  $\llbracket \text{drink}(x) \wedge \text{drive}(x) \rrbracket^{M,g[x/V(b^*)]} = 1$
  - iff  $\llbracket \text{drink}(x) \rrbracket^{M,g[x/V(b^*)]}$  and  $\llbracket \text{drive}(x) \rrbracket^{M,g[x/V(b^*)]} = 1$
  - iff  $V_M(\text{drink})(V_M(b^*)) = 1$  and  $V_M(\text{drive})(V_M(b^*)) = 1$

28

## β-Reduction

- $\llbracket \lambda v \alpha(\beta) \rrbracket^{M,g} = \llbracket \alpha \rrbracket^{M,g[v/\llbracket \beta \rrbracket^{M,g}]}$ 
  - $\Rightarrow$  all (free) occurrences of the  $\lambda$ -variable in  $\alpha$  get the interpretation of  $\beta$  as value.
- **Syntactic shortcut: β-reduction**
  - $\lambda v \alpha(\beta) \Leftrightarrow [\beta/v]\alpha$
  - $[\beta/v]\alpha$  is the result of replacing all *free occurrences* of  $v$  in  $\alpha$  with  $\beta$ .
- **Achtung:** The equivalence is not unconditionally valid

29

## Variable capturing

- Are  $\lambda v \alpha(\beta)$  and  $[\beta/v]\alpha$  always equivalent?
  - $\lambda x[\text{drive}'(x) \wedge \text{drink}'(x)](j^*) \Leftrightarrow \text{drive}'(j^*) \wedge \text{drink}'(j^*)$
  - $\lambda x[\text{drive}'(x) \wedge \text{drink}'(x)](y) \Leftrightarrow \text{drive}'(y) \wedge \text{drink}'(y)$
  - $\lambda x[\forall y \text{ know}'(x)(y)](j^*) \Leftrightarrow \forall y \text{ know}(j^*)(y)$
  - **NOT:**  $\lambda x[\forall y \text{ know}'(x)(y)](y) \Leftrightarrow \forall y \text{ know}(y)(y)$
- Let  $v, v'$  be variables of the same type,  $\alpha$  any well-formed expression.
- **v is free for v' in α** iff no free occurrence of  $v'$  in  $\alpha$  is in the scope of a quantifier or a  $\lambda$ -operator that binds  $v$ .

30

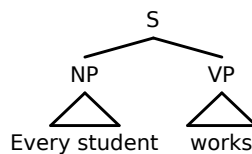
## Conversion rules

- **$\beta$ -conversion:**  $\lambda v \alpha(\beta) \Leftrightarrow [\beta/v]\alpha$ 
  - if all free variables in  $\beta$  are free for  $v$  in  $\alpha$ .
- **$\alpha$ -conversion:**  $\lambda v \alpha \Leftrightarrow \lambda w [w/v]\alpha$ 
  - if  $w$  is free for  $v$  in  $\alpha$ .
- **$\eta$ -conversion:**  $\lambda v (\alpha(v)) \Leftrightarrow \alpha$

31

## Back to noun phrases

- *Every student*
  - $\mapsto \lambda P \forall x (\text{student}'(x) \rightarrow P(x))$
  - Type:  $\langle (e, t), t \rangle$
- Interpretation:
  - *Every student* denotes the set of properties that apply to every student ("property" = sets of individuals).
  - $\llbracket \text{Every student} \rrbracket = \{ P \mid \text{every student has property } P \}$
- Semantic construction rule for  $S \rightarrow NP VP$ :
  - if  $VP \mapsto \alpha'$  and  $NP \mapsto \beta'$ , then  $S \mapsto \beta'(\alpha')$



32

## Back to noun phrases

- Interpretation of "every student:" the set of properties  $P$  that apply to every student
  - *every student*  $\mapsto \lambda P \forall x (\text{student}'(x) \rightarrow P(x))$
- Interpretation of "a student:" the set of properties  $P$  that apply to some student.
  - *a student*  $\mapsto \lambda P \exists x (\text{student}'(x) \wedge P(x))$
- Interpretation of "Bill:" the set of properties  $P$  that apply to Bill.
  - *Bill*  $\mapsto \lambda P.P(b^*)$

33

## Determiners

- *a, some*  $\mapsto \lambda F \lambda G \exists x (F(x) \wedge G(x))$
- *every*  $\mapsto \lambda F \lambda G \forall x (F(x) \rightarrow G(x))$
- *no*  $\mapsto \lambda F \lambda G \neg \exists x (F(x) \wedge G(x))$
- ...

34

## Semantics Construction Rules (2<sup>nd</sup> Version)

- **S  $\rightarrow$  NP VP**  
if VP  $\mapsto \alpha'$  and NP  $\mapsto \beta'$ , then S  $\mapsto \beta'(\alpha')$
- **NP  $\rightarrow$  DET N**  
if DET  $\mapsto \alpha'$  and N  $\mapsto \beta'$ , then NP  $\mapsto \alpha'(\beta')$
- **NP  $\rightarrow$  PN**  
if PN  $\mapsto \alpha'$ , then NP  $\mapsto \alpha'$
- **VP  $\rightarrow$  IV**  
if IV  $\mapsto \alpha'$ , then VP  $\mapsto \alpha'$

35

## *Every student works*

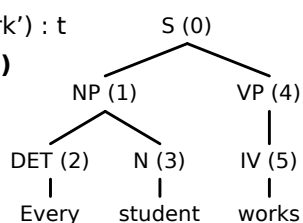
(2)  $\mapsto \lambda P \lambda Q \forall x (P(x) \rightarrow Q(x)) : \langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle$

(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  
 $\Rightarrow_{\beta} \forall x (\text{student}'(x) \rightarrow \text{work}'(x))$



36

## Literature

- L.T.F. Gamut (1991): *Logic, Language and Meaning, Vol II*. University of Chicago Press. Chapter 4
- David Dowty, Robert Wall and Stanley Peters (1981): *Introduction to Montague Semantics*. Dordrecht, Reidel. Chapter 4.