Semantic Theory

Type-Theory & Basic Semantics Construction

Manfred Pinkal
Stefan Thater

2009-06-09

Overview

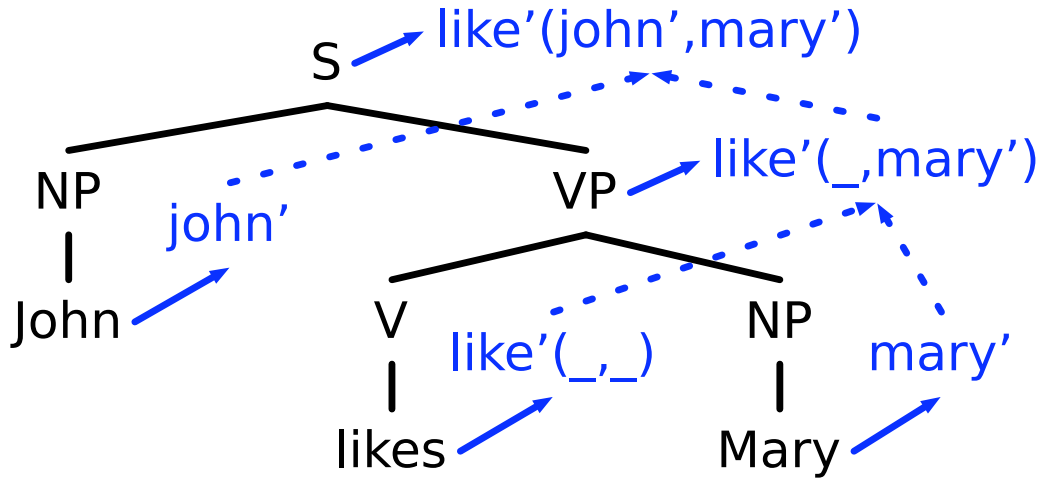
- Semantics construction & the principle of compositionality
- Limits of first-order predicate logic
- Higher-order logic: Type theory
- Basic semantics construction with type theory

Semantics Construction

- How can we derive appropriate semantic representations in a systematic way?
- The principle of compositionality: 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)

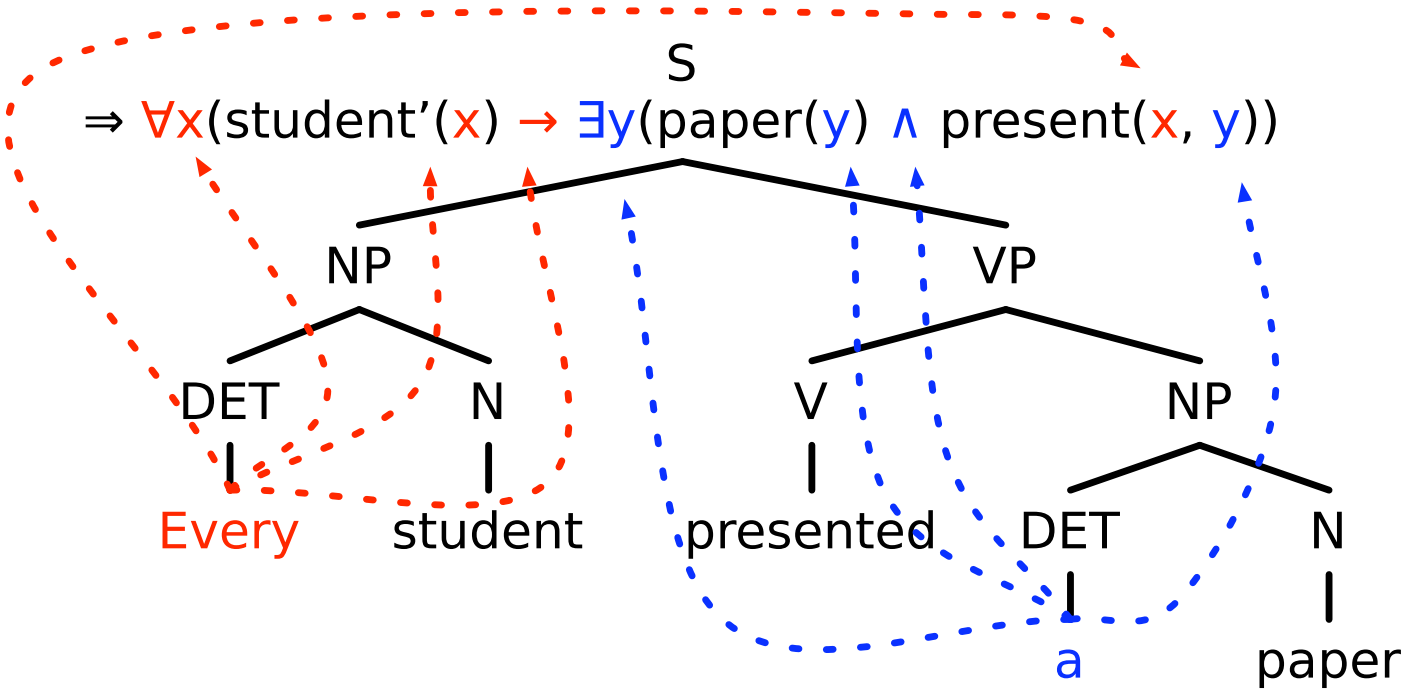
Basic Semantics Construction

- *John likes Mary*
  - like'(john', mary')



Semantics Construction: a Challenge

- *Every student presents a paper*
  - ∀x(student'(x) → ∃y(paper'(y) ∧ present'(x, y)))



Semantics Construction

- The structure of semantic representations of (first-order) logic does not necessarily correspond to the syntactic structure of natural language sentences.
- How can we model the semantic composition process? Can we maintain the principle of compositionality?
- We approach the problem by looking into higher-order phenomena in natural language semantics.

Limits of first-order logic

(1) Bill is a student
- student’(bill’)

(2) Bill is blond
- blond’(bill’)

(3) Bill is a blond student
- student’(bill’) ∧ blond’(bill’)

- (1) and (2) entail that Bill is a blond student ✓
- (3) entails that Bill is both blond and a student ✓

Limits of first-order logic

(1) Bill is a student
- student’(bill’)

(2) Bill is a tennis-player
- tennis-player’(bill’)

(3) Bill is a good student
- student’(bill’) ∧ good’(bill’)

(4) Bill is a good tennis-player
- tennis-player’(bill’) ∧ good’(bill’)

- (1), (2) and (3) might be true while (4) being false.
- But the representations of (2) and (3) entail (4) !!!

Limits of first-order logic

(1) Bill is a student

- student’(bill’)

(2) Bill is an alleged student

- student’(bill’) ∧ alleged’(bill’)

- (1) can be false while (2) being true
- But the representation of (2) entails that Bill is a student !!!

Limits of first-order logic

- Blond is a hair color
  - haircolor’(blond’)
- Bill and Mary have the same hair color
  - $\exists G$.haircolor’(G) $\wedge$ G(bill’) $\wedge$ G(mary’)
- Bill has all properties of a good student
  - $\forall G \exists$x(good-student’(x) $\wedge$ G(x)) $\rightarrow$ G(bill’)
  - $\forall G \forall$x((good-student’(x) $\wedge$ G(x)) $\rightarrow$ G(bill’))
- Bill is always late
- Possibly, Bill will pass the exam.

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.

Types

- Basic types:
  - e (“entities”)
  - t (“truth-values”)
- Complex types:
  - If σ, τ are types, then (σ, τ) is a type.
- Complex types are the type of functions mapping arguments of type σ to values of type τ.

Vocabulary

- For every type τ a possibly empty, pairwise disjoint sets of non-logical constants CON_τ
- For every type τ an infinite and pairwise disjoint sets of variables VAR_τ
- The usual logical operators: $\forall$, $\exists$, $\wedge$, $\vee$, ...

Type Theory – Syntax

- The sets of well-formed expressions WE_τ for every type τ are given by:
  - $CON_\tau \subseteq WE_\tau$, for every type τ
  - If α is in $WE_{(\sigma, \tau)}$, β in WE_σ, then $\alpha(\beta) \in WE_\tau$.
  - 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.
  - If A is in WE_t, then $\forall vA$ and $\exists vA$ are in WE_t, where v is a variable of arbitrary type.
  - If α, β are well-formed expressions of the same type, then $\alpha = \beta \in WE_t$.

Function Application

- The most important syntactic operation in type-theory is function application:
  - If α is in $WE_{(\sigma, \tau)}$, β in WE_{σ}, then $\alpha(\beta) \in WE_{\tau}$.
- A functor of complex type combines with an appropriate argument to a (syntactically) more complex expression of less complex.

“Bill is a blond student”	
blond’	: ((e, t), (e, t))
student’	: (e, t)
blond’(student’)	: (e, t)
bill’	: e
blond’(student’)(bill’)	: t

Well-formed or not?

(1) $\exists x(\text{girl}'(x) \wedge \text{like}'(x)(\text{bill}'))$

(2) $\forall x((\text{blond}'(x) \wedge \text{girl}'(x)) \rightarrow \text{like}'(x)(\text{peter}'))$

(3) $\text{someone}'(\text{like}'(\text{peter}'))$

(4) $\text{blond}'(\text{girl}')$

(5) $\forall x(\text{blond}'(\text{girl}') \rightarrow \text{like}'(x)(\text{peter}'))$

(6) $\forall \mathcal{G} \forall x(\text{blond}'(\mathcal{G})(x) \rightarrow \mathcal{G}(x))$

girl’	: (e, t)
like’	: (e, (e, t))
blond’	: ((e, t),(e, t))
someone’	: ((e, t), t)
peter’	: e
mary’	: e
x	: e
$\mathcal{G}$	: (e, t)

Type Theory – Semantics [1/3]

- Let U be a non-empty set of entities.
- The domain of possible denotations D_τ for every type τ is given by:
  - $D_e = U$
  - $D_t = \{0,1\}$
  - $D_{(\sigma, \tau)}$ is the set of all functions from D_σ to D_τ

Characteristic Functions

- Many natural language expression have a type (σ, t).
- These types are the type of functions mapping elements of type σ to true or false.
- Such function are also known as characteristic functions, and can be thought of as subsets of D_σ.
- Example: “student” is a constant of type (e,t) and can be seen as characterising the set of students.

Currying

- All functional types are interpreted as one-place functions.
- How do we deal with functions with multiple arguments?
- Currying (“Schönfinkeln”): simulate an n-place function as as a a chain of one-place functions.
  - simulate (e • e, t) as (e, (e, t))
  - simulate (e • e • e, t) as (e, (e, (e, t)))
  - …

Type Theory – Semantics [2/3]

- A model structure for a type theoretic language consists of a pair $M = (U_M, V_M)$, where
  - U_M is a non-empty domain of individuals
  - V_M is an interpretation function, which assigns to every member of CON_τ an element of D_τ.
- Variable assignment g assigns every variable of type τ a member of D_τ

Type Theory – Semantics [3/3]

- Interpretation with respect to model structure M and variable assignment g:

$$
\begin{aligned}
[\![\alpha]\!]^{M,g} &= V_M(\alpha), \text{ if } \alpha \text{ is a constant} \\
[\![\alpha]\!]^{M,g} &= g(\alpha), \text{ if } \alpha \text{ is a variable} \\
[\![\alpha(\beta)]\!]^{M,g} &= [\![\alpha]\!]^{M,g}([\![\beta]\!]^{M,g}) \\
[\![\neg\phi]\!]^{M,g} = 1 \text{ iff } & [\![\phi]\!]^{M,g} = 0 \\
[\![\phi \wedge \psi]\!]^{M,g} = 1 \text{ iff } & [\![\phi]\!]^{M,g} = 1 \text{ and } [\![\psi]\!]^{M,g} = 1 \\
& \ldots \\
[\![\alpha = \beta]\!]^{M,g} = 1 \text{ iff } & [\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g} \\
[\![\exists v\phi]\!]^{M,g} = 1 \text{ iff } & \text{there is a } d \in D_\tau \text{ such that } [\![\phi]\!]^{M,g[v/d]} = 1 \\
[\![\forall v\phi]\!]^{M,g} = 1 \text{ iff } & \text{for all } d \in D_\tau : [\![\phi]\!]^{M,g[v/d]} = 1
\end{aligned}
$$

- where v is a variable of type τ

Type Theory

- The definition of the syntax and semantics of type theory is a straightforward extension of first-order logic.
- Notions like “satisfies,” “valid,” “satisfiable,” “entailment” carry over almost verbatim from first-order logic.
- Type theory is sometimes called “higher-order logic:”
  - first-order logic allows quantification over individual variables
  - second-order logic allows quantification over variables of type (σ, τ) where σ and τ are atomic
  - …

Example [⇒ whiteboard]

- *John reads a book*
  - $[\![\exists x(\text{book}'(x) \wedge \text{read}'(x)(\text{john}'))]\!]^{M,g} = 1$ iff …

Adjectives, revisited

- Uniform analysis of adjectives: type ((e, t), (e, t))

(1) Bill is a blond student

- blond’(student’)(bill’)

(2) Bill is a good student

- good’(student’)(bill’)

(3) Bill is a alleged student

- alleged’(student’)(bill’)

(3) ⊭ Bill is a student ✓
(2) ⊭ Bill is a student ↯
(1) ⊭ Bill is a student ↯
(1) ⊭ Bill is blond ↯

Adjectives, revisited

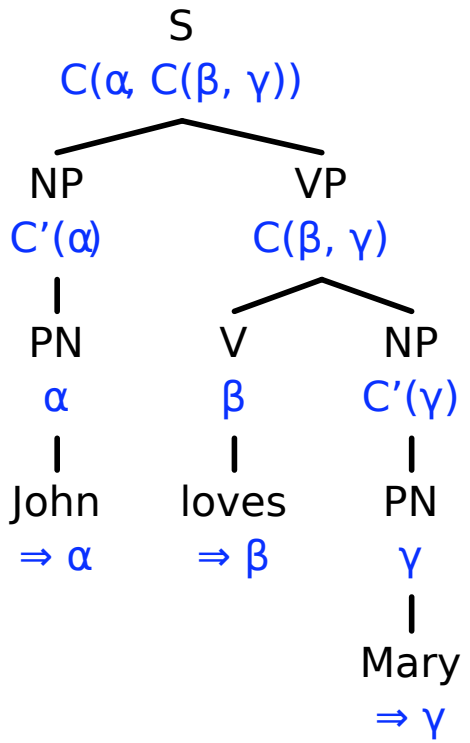
- Uniform analysis of adjectives: type ((e, t), (e, t))
- Meaning postulate intersective adjectives (“blond”)
  - ⟦ blond N ⟧ = ⟦ blond ⟧ ∩ ⟦ N ⟧
  - $\forall G \forall x(\text{blond}'(G)(x) \rightarrow (\text{blond}^*(x) \wedge G(x))$
  - Note: blond’ and blond* are different predicates
- Meaning postulate for subsective adjectives (“good”)
  - ⟦ good N ⟧ ⊆ ⟦ N ⟧
  - *[⇒ exercise]*
- Meaning postulate for privative adjectives (“former”)
  - ⟦ former N ⟧ ∩ ⟦ N ⟧ = ∅
  - *[⇒ exercise]*

Semantic interpretation of NL

- Semantic interpretation is a two-step process
  - Natural language (NL) expressions are assigned a semantic representation (logical formulas).
  - The semantic representation is truth-conditionally interpreted.
- Truth-conditional interpretation of logical representations is strictly compositional.
- We also want this for the process of computing logical representations from natural language expressions.

Semantics construction – Idea

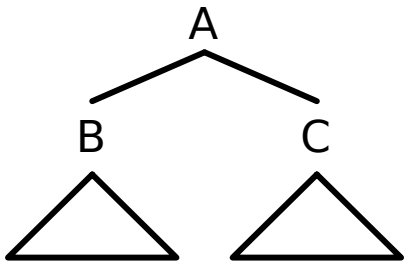
- Start with a syntactic analysis of a natural language expression
- Assign each syntactic node in the syntax tree a semantic representation by combining the representations of its daughter nodes.



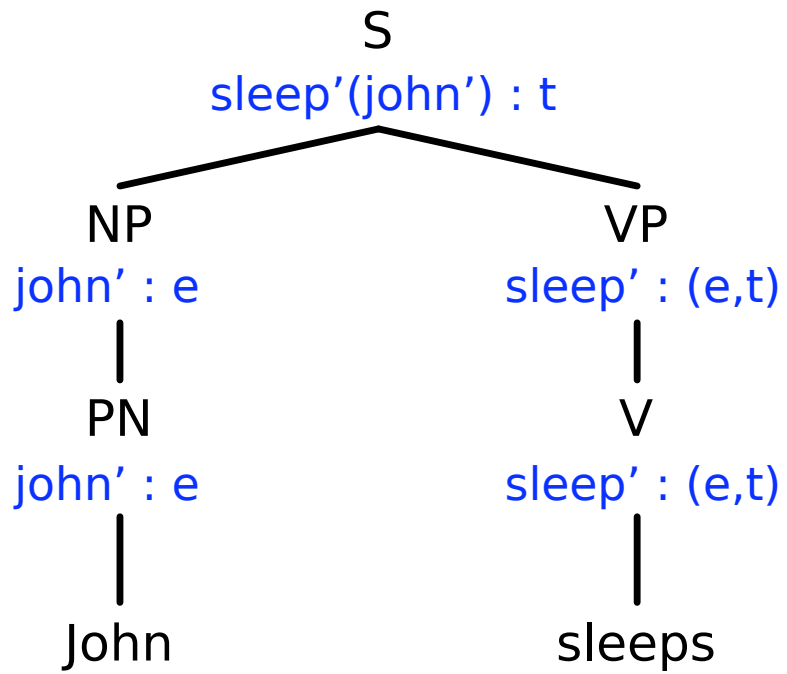
Basic composition rules

- Lexical nodes: semantic representation β for a word w is supplied by the lexicon.
- Non-branching nodes: if A is a node with child B, then the semantic representation of A is equal to the one of B.
- Binary branching nodes: if A is a node with children B, C, and $\beta \in WE_{(\sigma, \tau)}$, $\gamma \in WE_{\sigma}$ are the semantic representations of B and C, respectively, then the semantic representation of A is obtained by applying β to γ:

$$\frac{B \Rightarrow \beta : (\sigma, \tau) \qquad C \Rightarrow \gamma : \sigma}{A \Rightarrow \beta(\gamma) : \tau}$$



John sleeps


S
sleep’(john’) : t
NP
john’ : e
PN
john’ : e
John
VP
sleep’ : (e,t)
V
sleep’ : (e,t)
sleeps

John loves Mary