Semantic Theory
Lecture 3: Type Theory 1

Manfred Pinkal
FR 4.7 Computational Linguistics and Phonetics

Summer 2014

UNIVERSITÄT
DES
SAARLANDES

Limitations of first-order logic (1)

- First-order logic talks about
  - Individual objects
  - Properties of and relations between individual objects
  - Generalization across individual objects (quantification)
- FOL is not expressive enough to capture the full range of meaning information that can be expressed by **basic natural-language expressions**; examples:
  - *Bill is a poor piano player* (predicate modifiers)
  - *Blond is a hair color* (second-order predicates)
  - *Yesterday, it rained* (non-logical sentence operators)

Limitations of first-order logic (2)

- FOL cannot represent **higher-order quantification**, as in the following NL sentences, which express quantification over first-order predicates:
  - *Bill and John have the same hair color*
- To model these phenomena, we need higher order extensions of FOL. The maximally general higher order extension of logic is **Type Theory**.

Types

- **Basic types:**
  - **e** – the type of individual terms ("**e**ntities")
  - **t** – the type of formulas ("**t**ruth-values")
- **Complex types:**
  - If σ, τ are types, then **⟨σ, τ⟩** is a type.
  - An expression of type ⟨σ, τ⟩ is a functor expression that takes a σ type expression as argument and forms a type τ expression together with it.

Examples

- Types of first-order expressions:
  - Individual constants (*John, Saarbrücken*) : e
  - One-place predicates (*sleep, walk*): $\langle e, t \rangle$
  - Two-place predicates (*read, admire*): $\langle e, \langle e, t \rangle \rangle$
  - Three-place predicates (*give, introduce*): $\langle e, \langle e, \langle e, t \rangle \rangle \rangle$

- Higher-order types:
  - Predicate modifiers (*expensive, poor*): $\langle \langle e, t \rangle, \langle e, t \rangle \rangle$
  - Second-order predicates (*hair colour*): $\langle \langle e, t \rangle, t \rangle$
  - Sentence operators (*yesterday, possibly, unfortunately*): $\langle t, t \rangle$
  - Degree particles (*very, too*): $\langle \langle \langle e, t \rangle, \langle e, t \rangle \rangle, \langle \langle e, t \rangle, \langle e, t \rangle \rangle \rangle$

- Hint: If σ, τ are basic types, $\langle \sigma, \tau \rangle$ can be abbreviated as στ. Thus, the type of predicate modifiers and second-order predicates can be more conveniently written as $\langle et, et \rangle$ and $\langle et, t \rangle$ respectively.

Type Theory – Vocabulary

- **Non-logical constants:** For every type τ a (possibly empty) set of non-logical constants CON_τ (pairwise disjoint)
- **Variables:** For every type τ an infinite set of variables VAR_τ (pairwise disjoint)
- **Logical symbols:** $\forall$, $\exists$, $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $=$
- **Brackets:** (,)

Type Theory – Syntax

- The sets of **well-formed expressions WE_τ** for every type τ are given by:

(i) $CON_\tau \subseteq WE_\tau$ and $VAR_\tau \subseteq WE_\tau$, for every type τ

(ii) If α is in $WE_{\langle\sigma, \tau\rangle}$, β in WE_σ, then $\alpha(\beta) \in WE_\tau$.

(iii) If φ, ψ are in WE_t, then $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$ are in WE_t.

(iv) If φ is in WE_t and v is a variable of arbitrary type, then $\forall v\varphi$ and $\exists v\varphi$ are in WE_t.

(v) If α, β are well-formed expressions of the same type, then $\alpha = \beta \in WE_t$.

Type Theory – Function Application

- The sets of **well-formed expressions WE_τ** for every type τ are given by:
  - (i) $CON_\tau \subseteq WE_\tau$ and $VAR_\tau \subseteq WE_\tau$, for every type τ
  - (ii) If α is in $WE_{\langle\sigma, \tau\rangle}$, β in WE_σ, then $\alpha(\beta) \in WE_\tau$.
  - (iii) If φ, ψ are in WE_t, then $\neg\varphi$, $(\varphi\wedge\psi)$, $(\varphi\vee\psi)$, $(\varphi\rightarrow\psi)$, $(\varphi\leftrightarrow\psi)$ are in WE_t.
  - (iv) If φ is in WE_t and v is a variable of arbitrary type, then $\forall v\varphi$ and $\exists v\varphi$ are in WE_t.
  - (v) 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 $\alpha \in \mathrm{WE}_{<\sigma, \tau>}$, $\beta \in \mathrm{WE}_{\sigma}$, then $\alpha(\beta) \in \mathrm{WE}_{\tau}$.

- Note: A functor of complex type combines with an appropriate argument to a **more complex expression** of **less complex type**.

- The syntactic composition of complex type-theoretic expressions is unsually represented as a "type inference schema":

Bill drives fast

drive: et   *fast:* ⟨et, et⟩

bill: e   *fast(drive):* et

fast(drive)(bill): t

“Inverse” Type Inference

- *John believes that Bill likes Mary*

like: ⟨e, ⟨e,t⟩⟩ mary: e

bill: e like (mary): ⟨e,t⟩

believe: ? like (mary)(bill): t

john: e believe(like (mary)(bill)): ?

believe(like (mary)(bill))(john): t

- *believe* must be a functor **⟨σ, τ⟩**, because its sister term is basic; more specifically, the type of “like(mary)(bill)“ is t, so *believe* must be **⟨t, τ⟩**.
- τ is the result type, i.e., the type of “believe(like (mary)(bill))“; the expression takes an e to form a t, so **τ is ⟨e,t⟩**.
- *believe* must have type **⟨t, ⟨e,t⟩**⟩

Type Theory – Higher-Order Quantification

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

Higher-Order Quantification, Examples

- *Bill has the same hair colour as John.*

  $$\exists \mathcal{G}\,(\text{hair\_colour}(\mathcal{G}) \wedge \mathcal{G}\,(\text{bill}) \wedge \mathcal{G}\,(\text{john}))$$

- Construction using the type inference schema:

$$\dfrac{\text{hair colour: } \langle\langle e,t\rangle,t\rangle \quad \mathcal{G}\text{: } \langle e,t\rangle}{\text{hair\_colour}(\mathcal{G})\text{: t}} \qquad \dfrac{\dfrac{\text{bill: e} \quad \mathcal{G}\text{: } \langle e,t\rangle}{\mathcal{G}(\text{bill})\text{: t}} \qquad \dfrac{\text{john: e} \quad \mathcal{G}\text{: } \langle e,t\rangle}{\mathcal{G}(\text{john})\text{: t}}}{\mathcal{G}(\text{bill}) \wedge \mathcal{G}(\text{john})\text{: t}}$$

$$\dfrac{\text{hair\_colour}(\mathcal{G})\text{: t} \qquad \mathcal{G}(\text{bill}) \wedge \mathcal{G}(\text{john})\text{: t}}{\dfrac{\text{hair\_colour}(\mathcal{G}) \wedge \mathcal{G}\,(\text{bill}) \wedge \mathcal{G}\,(\text{john})\text{: t}}{\exists \mathcal{G}\,(\text{hair\_colour}(\mathcal{G}) \wedge \mathcal{G}\,(\text{bill}) \wedge \mathcal{G}\,(\text{john}))\text{: t}}}$$

Type Theory – Higher-Order Equality

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

Higher-Order Equality

- Type-theoretic equality is an operator with very strong expressive power. It can express equivalence between representations of any type. Examples:

- For p, q $\in$ CON_t, "p=q" expresses material equivalence:"$p \leftrightarrow q$".

- For F, G $\in$ $CON_{\langle e, t \rangle}$, "F=G" expresses co-extensionality: "$\forall x(Fx \leftrightarrow Gx)$"

- For any formula φ, φ=(x=x) is a representation of "φ is true".