Semantic Theory

Type Theory

Manfred Pinkal
Stefan Thater

Summer 2007

Logic as a framework for NL semantics

- Approximate NL meaning as truth conditions.
- Logic supports precise, consistent and controlled meaning representation via truth-conditional interpretation.
- Logic provides deduction systems to model inference processes, controlled through a formal entailment concept.
- Logic supports uniform modelling of the semantic composition process.

Outline

- A reminder: First-order predicate logic (FOL).
- The limits of FOL as a formalism for semantic representations.
- Type theory

Syntax of FOL [1]

- Non-logical expressions:
  - Individual constants: CON = { j*, b*, … }
  - n-place relation constants: REL^n, for all $n \geq 0$
- Individual variables: VAR = { x, y, z, … }
- Terms: TERM = VAR ∪ CON
- Atomic formulas:
  - $R(t_1, \ldots, t_n)$ for $R \in REL^n$ and $t_1, \ldots, t_n \in TERM$
  - $s = t$ for $s, t \in TERM$

Syntax of FOL [2]

- FOL formulas: The smallest set FORM such that
  - all atomic formulas are in FORM
  - if φ, ψ are in FORM, then $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$ are in FORM
  - if x is individual variable, and φ is in FORM, then $\forall x\varphi$ and $\exists x\varphi$ are in FORM

Scope

- If $\forall x\varphi$ ($\exists x\varphi$) is a subformula of ψ, then we call φ the scope of this occurrence of $\forall x$ ($\exists x$) in ψ.
  - We distinguish distinct occurrences of quantifiers as there are formulae like $\forall xA(x) \wedge \forall xB(x)$.
- An example:
  - $\exists x(\underline{\forall y\underline{(T(y) \leftrightarrow x=y)}} \wedge F(x))$

Free and Bound Variables

- An occurrence of a variable x in a formula φ is said to be free in φ if this occurrence of x does not fall within the scope of a quantifier $\forall x$ or $\exists x$ in φ.
- If $\forall x\psi$ (or $\exists x\psi$) is a subformula of φ and x is free in ψ, then this occurrence of x is said to be bound by this occurrence of the quantifier $\forall x$ (or $\exists x$).
  - $\forall x\ (A(x) \wedge B(x))$
  - $\forall x\ A(x) \wedge B(x)$
- A sentence is a formula without free variables.

Notational Variants

- We usually omit outermost brackets
  - $A \wedge B$ instead of $(A \wedge B)$
- We usually omit brackets if no ambiguities can arise
  - $A \wedge B \wedge C$ instead of $(A \wedge (B \wedge C))$
- We sometimes omit brackets for atomic formulas:
  - Rxy instead of R(x,y)
- Alternative notation for quantifiers
  - $\exists x\ .\ A(x) \wedge B(x)$ instead of $\exists x(A(x) \wedge B(x))$

Semantics of FOL [1]

- Model structure for FOL: $M = \langle U_M, V_M \rangle$
  - U_M is non-empty universe (individual domain)
  - V_M is an interpretation function, which assigns
    - individuals ($\in U_M$) to individual constants and
    - n-ary relations between individuals ($\in U_M^n$) to n-place predicate symbols.
- Assignment function for variables $g: VAR \rightarrow U_M$

Semantics of FOL [2]

- Interpretation of terms with respect to model structure M and variable assignment g:
  - $[\![\alpha]\!]^{M,g} = V_M(\alpha)$, if α is an individual constant
  - $[\![\alpha]\!]^{M,g} = g(\alpha)$, if α is a variable

Semantics of FOL [3]

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

$$[\![R(t_1, \ldots, t_n)]\!]^{M,g} = 1 \text{ iff } \langle [\![t_1]\!]^{M,g}, \ldots, [\![t_n]\!]^{M,g} \rangle \in V_M(R)$$

$$[\![s = t]\!]^{M,g} = 1 \text{ iff } [\![s]\!]^{M,g} = [\![t]\!]^{M,g}$$

$$[\![\neg\varphi]\!]^{M,g} = 1 \text{ iff } [\![\varphi]\!]^{M,g} = 0$$

$$[\![\varphi \wedge \psi]\!]^{M,g} = 1 \text{ iff } [\![\varphi]\!]^{M,g} = 1 \text{ and } [\![\psi]\!]^{M,g} = 1$$

$$[\![\varphi \vee \psi]\!]^{M,g} = 1 \text{ iff } [\![\varphi]\!]^{M,g} = 1 \text{ or } [\![\psi]\!]^{M,g} = 1$$

$$[\![\varphi \rightarrow \psi]\!]^{M,g} = 1 \text{ iff } [\![\varphi]\!]^{M,g} = 0 \text{ or } [\![\psi]\!]^{M,g} = 1$$

$$[\![\varphi \leftrightarrow \psi]\!]^{M,g} = 1 \text{ iff } [\![\varphi]\!]^{M,g} = [\![\psi]\!]^{M,g}$$

Semantics of FOL [4]

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

  $[\![\exists x\varphi]\!]^{M,g} = 1$
  iff there is an $a \in U_M$ such that $[\![\varphi]\!]^{M,g[x/a]} = 1$

  $[\![\forall x\varphi]\!]^{M,g} = 1$
  iff for all $a \in U_M$, $[\![\varphi]\!]^{M,g[x/a]} = 1$
- $g[x/a]$ is the variable assignment which is identical to g except that it assigns a to the variable x:
  - $g[x/a](y) = a$, if $x = y$
  - $g[x/a](y) = g(y)$, if $x \neq y$

Semantics of FOL [5]

- Formula φ is true in the model structure M iff $[[\varphi]]^{M,g} = 1$ for every variable assignment g.
- A model structure M satisfies a set of formulas Γ iff every formula $\varphi \in \Gamma$ is true in M.
  - We say that M is a model of Γ in this case.
- φ is valid iff φ is true in all model structures.
- φ is satisfiable iff there is a model structure that makes φ true; else it is unsatisfiable.
- φ is contingent iff φ is satisfiable but not valid.

Entailment and Deduction [1]

- A set of formulas Γ entails formula φ ($\Gamma \vDash \varphi$) iff φ is true in every model of Γ.
- A (sound and complete) calculus for FOL allows us to prove φ from Γ iff $\Gamma \vDash \varphi$ by manipulating the formulas syntactically.
  - There are many calculi for FOL: resolution, tableaux, natural deduction, ...

Entailment and Deduction [2]

- Calculi can be implemented to obtain:
  - theorem provers: check entailment, validity, and unsatisfiability
  - model builders: check satisfiability, compute models
  - model checkers: determine whether model satisfies a formula

Dolphins in FOL

- "Dolphins are mammals, not fish."
  - $\forall x\,(\text{dolphin}'(x) \rightarrow (\text{mammal}'(x) \wedge \neg\text{fish}'(x)))$
- "Dolphins live in pods."
  - $\forall x\,(\text{dolphin}'(x) \rightarrow \exists y(\text{pod}'(y) \wedge \text{live-in}'(x, y))$
- "Dolphins give birth to one baby at a time."
  - $\forall x\,(\text{dolphin}'(x) \rightarrow$
    $\forall y\,\forall z\,\forall t\,((\text{give-birth-to}'(x, y, t) \wedge$
    $\text{give-birth-to}'(x, z, t)) \rightarrow x = y)$

Students

- "Mary is a student."
  - student'(m*)
- "Mary reads a book."
  - ∃x(book'(x) ∧ read'(m*, x))
- "Every student presents a paper"
  - ∀x(student'(x) → ∃y(paper'(y) ∧ present'(x,y)))

Expressiveness of FOL [1]

- "John is a blond criminal"
  - criminal'(j*) ∧ blond'(j*)
- "John is a famous criminal"
  - criminal'(j*) ∧ famous'(j*) ?
- "John is an alleged criminal"
  - criminal'(j*) ∧ alleged'(j*) ???

Expressiveness of FOL [2]

- "John is walking quickly."
  - walk'(j*) ∧ quick'(j*) ?
- "John is walking very quickly."
  - ???

Expressiveness of FOL [3]

- "Bill is blond."
- "Blond is a hair-color."

Expressiveness of FOL [4]

- "It rains."
- "It rained yesterday."
- "It rains occasionally."

Expressiveness of FOL [5]

- "Mary has all properties of a successful student."

Type Theory

- The types of non-logical expressions provided by FOL 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

- For NL meaning representation the (minimal) set of basic types is {e, t}
  - e ("entity") is the type of individual terms
  - t ("truth value") is the type of formulas
- Complex types
  - If σ,τ are types, then $\langle\sigma, \tau\rangle$ is a type
  - $\langle\sigma, \tau\rangle$ is the type of functions which map arguments of type σ to values of type τ.

Type Theory – Syntax [1]

- Vocabulary:
  - Possibly empty, pairwise disjoint sets of non-logical constants:
    - CON_τ for every type τ
  - Infinite and pairwise disjoint sets of variables:
    - VAR_τ for every type τ
  - The logical operators known from FOL.

Type Theory – Syntax [2]

- 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 $\alpha \in WE_{\langle\sigma,\, \tau\rangle}$, $\beta \in WE_\sigma$, then $\alpha(\beta) \in WE_\tau$.
  - If φ, ψ are in WE_t (*i.e.,* formulas), then so are $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$
  - If φ is in WE_t , then so are $\forall v\varphi$ and $\exists v\varphi$, where v is a variable of arbitrary type.
  - If α, β are well-formed expressions of the same type, then $\alpha = \beta \in WE_t$.

Examples

- "Bill is walking."

$$\frac{bill^* : e \qquad walk' : \langle e,t\rangle}{walk'(bill^*)}$$

- "Bill is walking quickly."

$$\frac{bill^* : e \qquad \dfrac{walk' : \langle e,t\rangle \quad quick' : \langle\langle e,t\rangle,\langle e,t\rangle\rangle}{quick'(drive') : \langle e,t\rangle}}{quick'(drive')(bill^*) : t}$$

Examples

- "Mary works in Saarbrücken"

$$\frac{\dfrac{mary^* : e \qquad work' : \langle e,t\rangle}{work'(mary^*) : t} \qquad \dfrac{in' : \langle e,\langle t,t\rangle\rangle \quad sb^* : e}{in'(sb^*) : \langle t,t\rangle}}{in'(sb^*)(work'(mary^*)) : t}$$

Second-order predicates

- Bill is blond. Blond is a hair colour:
  - "Bill" is represented as a term of type e.
  - "blond" is represented as a term of type ⟨e,t⟩.
  - "hair colour" is represented as a term of type ⟨⟨e,t⟩,t⟩.

Type Theory – Semantics [1]

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

Type Theory – Semantics [2]

- A model structure for a type theoretic language is a pair $M = \langle U_M, V_M\rangle$, where
  - U_M is non-empty domain of individuals
  - V_M is function, which assigns every non-logical constant ($\in CON_\tau$) of type τ a member of D_τ.
- Variable assignment g assigns every variable of type τ a member of D_τ.

Type Theory – Semantics [3]

- Interpretation with respect to model structure M and variable assignment g:
  - $[\![\alpha]\!]^{M,g} = V_M(\alpha)$, if α constant
  - $[\![\alpha]\!]^{M,g} = g(\alpha)$, if α variable
  - $[\![\alpha(\beta)]\!]^{M,g} = [\![\alpha]\!]^{M,g}([\![\beta]\!]^{M,g})$
  - $[\![\neg\phi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = 0$
  - $[\![\phi \wedge \psi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = 1$ and $[\![\psi]\!]^{M,g} = 1$,
  - ...
  - $[\![\alpha{=}\beta]\!]^{M,g} = 1$ iff $[\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g}$

Type Theory – Semantics [3]

- Interpretation with respect to model structure M and variable assignment g:
  - $[\![\exists v \phi]\!]^{M,g} = 1$
    iff there is an $a \in D_\tau$ such that $[\![\phi]\!]^{M,g[v/a]} = 1$
  - $[\![\forall v \phi]\!]^{M,g} = 1$
    iff for all $a \in D_\tau$, $[\![\phi]\!]^{M,g[v/a]} = 1$
  - where $v \in VAR_\tau$

Characteristic Functions

- A function of type $\langle \sigma, t \rangle$ maps each member of D_σ to true or false.
- See this as representing a subset of D_σ
  - namely, the set of members of D_σ that are mapped to true.
- Example: "blond" is a constant of type $\langle e, t \rangle$. It can be seen as characterising the set of blond individuals (of type e).

Currying

- All functional types are interpreted as one-place functions.
- How do we deal with functions/relations with multiple arguments?
- Currying (a.k.a. "Schönfinkeln"):
  - simulate term P(a,b) as the term P(a)(b)
  - simulate type $\langle e \times e, t \rangle$ as the type $\langle e, \langle e,t \rangle \rangle$

Mary reads a book

- Predicate logic
  - $\exists x(\text{book}'(x) \wedge \text{read}'(m^*, x))$
- Type theory
  - $\exists x(\text{book}'(x) \wedge \text{read}'(m^*)(x))$

Type Theory

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

Meaning Postulates

- "John is walking quickly"
  - quick'(walk')(john*)
- "Mary works in Saarbrücken."
  - in'(sb*)(work'(mary*))

Summary

- First-order logic is nice, but its expressiveness is limited, and some NL phenomena cannot be modelled adequately.
  - modification
  - modification of modifiers
  - higher-order properties
  - …
- Type theory is a generalisation of first-order logic that allows us to represent the semantics of all these expressions.