Semantic Theory
Lecture 2: Type theory

M. Pinkal / A. Koller
Summer 2006

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.

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.
- Modal operators in logic.

Dolphins

Dolphins are mammals, not fish.

$\forall$d (dolphin(d)$\rightarrow$mammal(d) $\wedge\neg$fish(d))

Dolphins live in pods.

$\forall$d (dolphin(d)$\rightarrow$ $\exists$x (pod(p) $\wedge$ live-in (d,p))

Dolphins give birth to one baby at a time.

$\forall$d (dolphin(d)$\rightarrow$ $\forall$x $\forall$y $\forall$t (give-birth-to (d,x,t) $\wedge$ give-birth-to (d,y,t) $\rightarrow$ x=y)

Syntax of FOL [1]

- Non-logical expressions:
  - Individual constants: IC
  - n-place predicate symbols: RC^n (n ≥ 0)
- Individual variables: IV
- Terms: T = IV$\cup$IC
- Atomic formulas:
  - $R(t_1,...,t_n)$ for R$\in$ RC^n, if t_1, ..., $t_n \in$T
  - s=t for s, t $\in$T

Syntax of FOL [2]

- FOL formulas: The smallest set *For* such that:
  - All atomic formulas are in *For*
  - If A, B are in *For*, then so are $\neg$ A, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$,$(A \leftrightarrow B)$
  - If x is an individual variable and A is in *For*, then $\forall x A$ and $\exists x A$ are in *For*.

Dolphins in FOL

Dolphins are mammals, not fish.

$\forall d\ (dolphin(d) \rightarrow mammal(d) \wedge \neg fish(d))$

Dolphins live in pods.

$\forall d\ (dolphin(d) \rightarrow \exists x\ (pod(p) \wedge live\text{-}in\ (d,p))$

Dolphins give birth to one baby at a time.

$\forall d\ (dolphin(d) \rightarrow \forall x\ \forall y\ \forall t\ (give\text{-}birth\text{-}to\ (d,x,t) \wedge give\text{-}birth\text{-}to\ (d,y,t) \rightarrow x=y)$

Semantics of FOL [1]

- Model structures for FOL: M = <U, V>
  - U (or U_M) is a non-empty universe (domain of individuals)
  - V (or 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: IV → U_M

Semantics of FOL [2]

- Interpretation of terms (with respect to a model structure M and a 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):

$[[R(t_1, ..., t_n)]]^{M,g} = 1$	iff	$\langle [[t_1]]^{M,g}, ..., [[t_n]]^{M,g} \rangle \in V_M(R)$
$[[s=t]]^{M,g} = 1$	iff	$[[s]]^{M,g} = [[t]]^{M,g}$
$[[\neg\varphi]]^{M,g} = 1$	iff	$[[\varphi]]^{M,g} = 0$
$[[\varphi \wedge \psi]]^{M,g} = 1$	iff	$[[\varphi]]^{M,g} = 1$ and $[[\psi]]^{M,g} = 1$
$[[\varphi \vee \psi]]^{M,g} = 1$	iff	$[[\varphi]]^{M,g} = 1$ or $[[\psi]]^{M,g} = 1$
$[[\varphi \rightarrow \psi]]^{M,g} = 1$	iff	$[[\varphi]]^{M,g} = 0$ or $[[\psi]]^{M,g} = 1$
$[[\varphi \leftrightarrow \psi]]^{M,g} = 1$	iff	$[[\varphi]]^{M,g} = [[\psi]]^{M,g}$
$[[\exists x\varphi]]^{M,g} = 1$	iff	there is $a \in U_M$ 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 with g except that it assigns the individual *a* to the variable x.

Semantics of FOL [4]

- Formula A is true in the model structure M iff $[[A]]^{M,g} = 1$ for every variable assignment g. This works best if A has no free variables.
- A model structure M satisfies a set of formulas Γ (or: M is a model of Γ) iff every formula $A \in \Gamma$ is true in M.
- A is valid iff A is true in all model structures.
- A is satisfiable iff there is a model structure that makes it true.
- A is unsatisfiable iff there is no model structure that makes it true.
- A is contingent iff it it is satisfiable but not valid.

Entailment and Deduction

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

Two levels of interpretation

- Semantic interpretation of a NL expression in a logical framework is a two-step process:
  - The NL expression is assigned a semantic representation
  - The semantic representation is truth-conditionally interpreted.

The expressive power of FOL [1]

John is a blond criminal

The expressive power of FOL [1]

John is a blond criminal

criminal(j) $\wedge$ blond(j)

The expressive power of FOL [1]

John is a blond criminal

criminal(j) ∧ blond(j)

John is an honest criminal

The expressive power of FOL [1]

John is a blond criminal

criminal(j) ∧ blond(j)

John is an honest criminal

criminal(j) ∧ honest(j) ?

The expressive power of FOL [1]

John is a blond criminal

criminal(j) ∧ blond(j)

John is an honest criminal

criminal(j) ∧ honest(j) ?

John is an alleged criminal

The expressive power of FOL [1]

John is a blond criminal

criminal(j) ∧ blond(j)

John is an honest criminal

criminal(j) ∧ honest(j) ?

John is an alleged criminal

criminal(j) ∧ alleged(j) ??

The expressive power of FOL [2]

John is driving fast

The expressive power of FOL [2]

John is driving fast

drive(j) ∧ fast(j)

The expressive power of FOL [2]

John is driving fast

drive(j) ∧ fast(j)

John is eating fast

The expressive power of FOL [2]

John is driving fast

drive(j) ∧ fast(j)

John is eating fast

eat(j) ∧ fast(j) ??

The expressive power of FOL [2]

John is driving fast

drive(j) ∧ fast(j)

John is eating fast

eat(j) ∧ fast(j) ??

John is driving very fast.

The expressive power of FOL [2]

John is driving fast

drive(j) ∧ fast(j)

John is eating fast

eat(j) ∧ fast(j) ??

John is driving very fast.

???

The expressive power of FOL [3]

It rains.

It rained yesterday.

It rains occasionally.

Bill is blond. Blond is a hair colour. (|≠ *Bill is a hair colour.*)

Type theory

- The types of non-logical expressions provided by FOL – terms and n-ary first-order relations – 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 (for entity) is the type of individual terms
  - t (for truth value) is the type of formulas
- All pairs $<\sigma, \tau>$ made up of (basic or complex) types σ, τ are types. $<\sigma, \tau>$ is the type of functions which map arguments of type σ to values of type τ.
- In short: The set of types is the smallest set **T** such that $e,t \in \mathbf{T}$, and if $\sigma,\tau \in \mathbf{T}$, then also $<\sigma,\tau> \in \mathbf{T}$.

Some useful complex types for NL semantics

- Individual: e
- Sentence: t
- One-place predicate constant: <e,t>
- Two-place relation: <e,<e,t>>
- Sentence adverbial: <t,t>
- Attributive adjective: <<e,t>,<e,t>>
- Degree modifier: <<<e,t>,<e,t>>,<<e,t>,<e,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>.
  - "*Bill is a hair colour"* is not even a well-formed statement.

Some useful complex types for NL semantics

- Individual: e
- Sentence: t
- One-place predicate constant: <e,t>
- Two-place relation: <e,<e,t>>
- Sentence adverbial: <t,t>
- Attributive adjective: <<e,t>,<e,t>>
- Degree modifier: <<<e,t>,<e,t>>,<<e,t>,<e,t>>>
- Second-oder predicate: <<e,t>,t>

Type-theoretic syntax [1]

- Vocabulary:
  - Possibly empty, pairwise disjoint sets of non-logical constants: Con_τ for every type τ

Higher-order variables

- *Bill has the same hair colour as John.*
- *Santa Claus has all the attributes of a sadist.*

Type-theoretic 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-theoretic 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 $\alpha \in WE_{<\sigma, \tau>}$, $\beta \in WE_\sigma$, then $\alpha(\beta) \in WE_\tau$.
  - If A, B are in WE_t, then so are $\neg$ A, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$,$(A \leftrightarrow B)$
  - If A is in WE_t, then so are $\forall v A$ and $\exists v A$, where v is a variable of arbitrary type.
  - If α, β are well-formed expressions of the same type, then $\alpha = \beta \in WE_{t.}$

Building well-formed expressions

Bill drives fast.

drive: <e,t>   fast: <<e,t>,<e,t>>

Bill: e   fast(drive): <e,t>

fast(drive)(bill): t

Mary works in Saarbrücken

mary: e   work: <e,t>   in: <e,<t,t>>   sb: e

work(mary): t   in(sb): <t,t>

in(sb)(work(mary)): t

More examples

- *Blond is a hair colour.*
- *Santa Claus has all the attributes of a sadist.*

Type-theoretic 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:
  - $D_e = U$
  - $D_t = \{0,1\}$
  - $D_{<\sigma, \tau>}$ is the set of all functions from D_σ to D_τ

Type-theoretic semantics [2]

- A model structure for a type theoretic language:

  M = <U, V>, where
  - U (or U_M) is a non-empty domain of individuals
  - V (or 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-theoretic 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

$[[\alpha(\beta)]]^{M,g} = [[\alpha]]^{M,g}([[\beta]]^{M,g})$

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

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

If $v \in Var_\tau$, $[[\exists v\varphi]]^{M,g} = 1$ iff there is $a \in D_\tau$ such that $[[\varphi]]^{M,g[v/a]} = 1$

If $v \in Var_\tau$, $[[\forall v\varphi]]^{M,g} = 1$ iff for all $a \in D_\tau$: $[[\varphi]]^{M,g[v/a]} = 1$

$[[\alpha=\beta]]^{M,g} = 1$ iff $[[\alpha]]^{M,g} = [[\beta]]^{M,g}$

Type theory

- The definition of the syntax and semantics of type theory is a straightforward extension of FOL.
- Words 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 $<\sigma, \tau>$ where σ and τ are atomic
  - .....

Currying

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

Summary

- First-order logic is nice, but its expressive power has limits that are not acceptable in NL semantics:
  - 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.