1 Model-Theoretic Semantics

• The theory of semantics developed there is model-theoretic semantics.

• Model-theoretic semantics aims at explaining native speakers’ semantic intuitions about various types of sentences, in particular truth-conditional intuitions: When given a declarative sentence and a situation, they can tell whether the sentence is true or false.

• In order to account for the fact that native speakers have truth-conditional intuitions about infinitely many grammatical sentences, we assume that English and other natural languages obey the (Local) Compositionality Principle.

  The (Local) Compositionality Principle
  The meaning of a syntactically complex phrase $A$ is determined solely by the meaning of its immediate daughters and the syntactic structure.

• Every grammatical phrase in (a ‘fragment’ of) English is assigned a model-theoretic object as its denotation relative to a model. The meanings of syntactically complex phrases are derived by combining meanings of their parts, as the Compositionality Principle states.

2 Models

• A model $\mathcal{M}$ is meant to model a particular state of affairs. Meaning in model-theoretic semantics is always relative to some model, reflecting the intuition that truth/falsity is relative to a state of affairs.

• Formally, a model $\mathcal{M}$ is a pair $\langle \mathcal{D}, \mathcal{I} \rangle$, where $\mathcal{D}$ is a non-empty set of individuals/entities and $\mathcal{I}$ is a function that assigns meanings to constants.

• The denotation of a given (gramamtical) phrase $\alpha$ relative to assignment $a$ and model $\mathcal{M}$ is written as $\llbracket \alpha \rrbracket^a_{\mathcal{M}}$. The assignment is used to interpret traces and pronouns (see below).

• Model-theoretic objects are either of the following three kinds:
  - Individuals/Entities
  - Truth-values (0 or 1)
  - Functions of various kinds

• Model-theoretic objects are grouped according to their types.

  (l) Types
a. \( e \) is a type.
b. \( t \) is a type.
c. If \( \sigma \) and \( \tau \) are both types, \( \langle \sigma, \tau \rangle \) is also a type.
d. Nothing else is a type.

(2) Domains
a. \( D_e \) is the set of individuals, \( \mathcal{D} \).
b. \( D_t \) is the set of truth-values, \( \{0, 1\} \).
c. \( D_{\langle \sigma, \tau \rangle} \) is the set of functions whose domain is \( D_{\sigma} \) and whose range is \( D_{\tau} \).

• Notation: We sometimes write \( et \) for \( \langle e, t \rangle \).

3 The Syntax-Semantics Interface

• We assume that the syntax generates hierarchically organised syntactic objects called LF(s) (or ‘Logical Forms’). We assume that branching is at most binary.
• LF(s) are fully disambiguated with respect to lexical and structural ambiguities, including quantifier scope.
• In particular we assume that Quantifier Raising has resolved type-mismatches that arise with quantifiers in non-subject position.
• We also assume that a syntactic movement, overt or covert, creates a binding index node and a trace with the same index, where indices are simply natural numbers. This is depicted in the following schematic diagram where \( i \in \mathbb{N} \).

```
  XP
 /\  
\\ i  
 /\  
YP
```

• The semantic is composed of two major components.
  – The lexicon is a list of meanings of morphemes (it may contain other information necessary for morphology, syntax and phonology, but we will not be concerned with it).
  – The compositional rules are instructions as to how to combine different kinds of meanings.
• Our semantic theory is type-driven, meaning that the semantic types (as opposed to, say, syntactic labels) determine how the semantic composition proceeds.
• The task of a semanticist is to enrich the lexicon and/or compositional rules so as to cover more phrases and constructions of various natural languages.
In Term 1, we covered the following types of items.

- Proper names denote individuals/entities.
  
  (3) For any assignment $a$ and for any model $M$,
  
  a. $\llbracket \text{Nathan} \rrbracket ^{a,M} = I(\text{Nathan}) = \text{Nathan}$
  
  b. $\llbracket \text{Andrea} \rrbracket ^{a,M} = I(\text{Andrea}) = \text{Andrea}$

- Intransitive predicates denote functions of type $\langle e, t \rangle$.

  (4) For any assignment $a$ and for any model $M$,
  
  a. $\llbracket \text{smokes} \rrbracket ^{a,M} = \lambda x \in D_e. x \text{ smokes in } M$
  
  b. $\llbracket \text{linguist} \rrbracket ^{a,M} = \lambda x \in D_e. x \text{ is a linguist in } M$
  
  c. $\llbracket \text{British} \rrbracket ^{a,M} = \lambda x \in D_e. x \text{ is British in } M$

  (More formally, $\llbracket \text{smokes} \rrbracket ^{a,M} = \lambda x \in D_e. x \in I(\text{smokes})$)

- Transitive predicates denote functions of type $\langle e, \langle e, t \rangle \rangle$.

  (5) For any assignment $a$ and for any model $M$,
  
  a. $\llbracket \text{likes} \rrbracket ^{a,M} = \lambda x \in D_e. \lambda y \in D_e. y \text{ likes } x \text{ in } M$
  
  b. $\llbracket \text{part} \rrbracket ^{a,M} = \lambda x \in D_e. \lambda y \in D_e. y \text{ is part of } x \text{ in } M$
  
  c. $\llbracket \text{fond} \rrbracket ^{a,M} = \lambda x \in D_e. \lambda y \in D_e. y \text{ is fond of } x \text{ in } M$
  
  d. $\llbracket \text{from} \rrbracket ^{a,M} = \lambda x \in D_e. \lambda y \in D_e. y \text{ is from } x \text{ in } M$

  (More formally, $\llbracket \text{likes} \rrbracket ^{a,M} = \lambda x \in D_e. \lambda y \in D_e. \langle x, y \rangle \in I(\text{likes})$)

- Semantically vacuous items denote identity functions.

  (6) For any assignment $a$ and for any model $M$,
  
  a. $\llbracket \text{apredicative} \rrbracket ^{a,M} = \lambda P \in D_{\langle e, t \rangle}. P$
  
  b. $\llbracket \text{is} \rrbracket ^{a,M} = \lambda P \in D_{\langle e, t \rangle}. P$
  
  c. $\llbracket \text{of} \rrbracket ^{a,M} = \lambda x \in D_e. x$
  
  d. $\llbracket \text{who} \rrbracket ^{a,M} = \lambda P \in D_{\langle e, t \rangle}. P$
  
  e. $\llbracket \text{such} \rrbracket ^{a,M} = \lambda P \in D_{\langle e, t \rangle}. P$

- Quantificational determiners denote functions of type $\langle et, \langle et, t \rangle \rangle$.

  (7) For any assignment $a$ and for any model $M$,
  
  a. $\llbracket \text{every} \rrbracket ^{a,M} = \lambda P \in D_{\langle e, t \rangle}. \lambda Q \in D_{\langle e, t \rangle}.$
     \begin{align*}
     & \text{for every } x \in D_e \text{ such that } P(x) = 1, Q(x) = 1 \\
     & \text{for every } x \in D_e \text{ such that } P(x) = 1, Q(x) = 1 \\
     & \text{for some } x \in D_e \text{ such that } P(x) = 1, Q(x) = 1 \\
    \end{align*}

  b. $\llbracket \text{no} \rrbracket ^{a,M} = \lambda P \in D_{\langle e, t \rangle}. \lambda Q \in D_{\langle e, t \rangle}.$
     \begin{align*}
     & \text{for no } x \in D_e \text{ such that } P(x) = 1, Q(x) = 1 \\
     & \text{for no } x \in D_e \text{ such that } P(x) = 1, Q(x) = 1 \\
     & \text{for some } x \in D_e \text{ such that } P(x) = 1, Q(x) = 1 \\
    \end{align*}

- These meanings combine via various compositional rules.

- For branching structures, we have three compositional rules:
Functional Application (FA)
For any assignment \(a\) and for any model \(\mathcal{M}\), if \(A\) is a branching node with children \(B\) and \(C\) such that \([C]^{a,\mathcal{M}} \in \text{dom}([B]^{a,\mathcal{M}})\), then \([A]^{a,\mathcal{M}} = [B]^{a,\mathcal{M}}([C]^{a,\mathcal{M}})\).

Predicate Modification (PM)
For any assignment \(a\) and for any model \(\mathcal{M}\), if \(A\) is a branching node with children \(B\) and \(C\) such that \([B]^{a,\mathcal{M}}\) and \([C]^{a,\mathcal{M}}\) are both of type \(\langle e, t \rangle\), then \([A]^{a,\mathcal{M}} = \lambda x \in D_e. [B]^{a,\mathcal{M}}(x) = [C]^{a,\mathcal{M}}(x) = 1\) for some variable \(x\) of type \(e\).

Predicate Abstraction (PA)
For any assignment \(a\), for any model \(\mathcal{M}\), and for any index \(i \in \mathbb{N}\),
\[
\left[ \begin{array}{c}
\lambda x \in D_e. [B]^{a,\mathcal{M}}(x) = [C]^{a,\mathcal{M}}(x) = 1 \\
\end{array} \right]^{a,\mathcal{M}} 
\]
for some variable \(x\) of type \(e\).

- Predicate Modification accounts for NP-modification by adjectives, PPs, and relative clauses.
- Predicate Abstraction accounts for structures with movement.
- Assignments
  - Assignments are functions from indices (natural numbers) to individuals.
  - Assignment modification \(a[i \rightarrow x]\) is the assignment that is just like \(a\) except that \(a(i) = x\).
- For non-branching structures, we have the following three compositional rules.

  Non-Branching Node Rule (NB)
  For any assignment \(a\) and for any model \(\mathcal{M}\),
  \[
  \left[ \begin{array}{c}
  A \\
  B \\
  \end{array} \right]^{a,\mathcal{M}} = [B]^{a,\mathcal{M}}. 
  \]

  Lexical Terminal Node Rule (TN)
  For any assignment \(a\) and for any model \(\mathcal{M}\), if \(A\) is a terminal node and is not a trace or pronoun, then \([A]^{a,\mathcal{M}}\) is in the lexicon.

  Traces and Pronouns Rule (TP)
  For any assignment \(a\) and for any model \(\mathcal{M}\), if \(A\) is a trace or a pronoun with an index \(i \in \mathbb{N}\), then \([A]^{a,\mathcal{M}} = a(i)\).