Lecture 2: Formal Semantics in Logic and Linguistics

1. English as a Formal Language.

R. Montague 1970, “English as a Formal Language” argued that the syntax and semantics of natural languages could be treated by the same kinds of techniques used by logicians to specify the syntax and model theoretic semantics of formal languages such as the predicate calculus.

This is the basic thesis of formal semantics. In these lectures we will clarify its principal points. In the process, we will try to answer the following questions:

- What is a formal language?
- What features of formal languages are most important for formal semantics?
- What are the main differences between “artificial” formal languages and natural language?
- For what parts of “real” natural language semantics can the framework of (existing) formal semantics offer useful tools for linguistic research? For what parts are different tools needed?

2. Example. Syntax and semantics of the predicate calculus (PC).

Predicate Calculus is the most well known and in a sense the prototypical example of a formal language. We use it to demonstrate features of formal languages which are most important for us: the notions of model and model-theoretic semantics, and the Principle of Compositionality.

We limit ourselves here to some examples and remarks. More exact definitions are given in Appendix 1.

The sentences John loves Mary and Everyone whom Mary loves is happy can be represented as formulas of PC:

\[ \text{John loves Mary} \]
\[ \text{love (John, Mary)} \]
\[ \text{Everyone whom Mary loves is happy} \]
\[ \forall x (\text{love(Mary, } x) \rightarrow \text{happy}(x)) \]

Formulas and other expressions of PC are built from individual constants (or simply “constants”), (individual) variables, predicate constants (or predicate symbols), logical connectives and quantifiers. Each expression belongs to a certain type. The type structure

---

1 “I reject the contention that an important theoretical difference exists between formal and natural languages. ... In the present paper I shall accordingly present a precise treatment, culminating in a theory of truth, of a formal language that I believe may reasonably be regarded as a fragment of ordinary English. ... The treatment given here will be found to resemble the usual syntax and model theory (or semantics) [due to Tarski] of the predicate calculus, but leans rather heavily on the intuitive aspects of certain recent developments in intensional logic [due to Montague himself].” (Montague 1970b, p.188 in Montague 1974)
of PC is very simple: individuals, relations of different arities (unary, binary, etc.), and truth-values.

In our examples we use the following expressions:

<table>
<thead>
<tr>
<th>Expressions</th>
<th>Syntactic categories</th>
<th>Semantic Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>John, Mary</td>
<td>(individual) constant</td>
<td>individuals</td>
</tr>
<tr>
<td>x</td>
<td>variable</td>
<td>individuals</td>
</tr>
<tr>
<td>happy</td>
<td>unary predicate constant</td>
<td>unary relations</td>
</tr>
<tr>
<td>love</td>
<td>binary predicate constant</td>
<td>binary relations</td>
</tr>
<tr>
<td>love (John, Mary)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>love(Mary, x)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>happy(x)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>∀x(love(Mary, x) → happy(x))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Expressions are interpreted in models. The structure common to all of the models in which a given language is interpreted (the model structure for the model-theoretic interpretation of the given language) reflects certain basic presuppositions about the “structure of the world” that are implicit in the language. For PC, any given model structure consists of the set of truth-values \{0,1\}, a domain D which is some set of objects (or entities), and some n-ary relations on this set.

A model, or interpreted model, consists of a model structure plus a (“lexical”, or “basic”) interpretation function I which assigns semantic values to all constants.

\[ \text{M} = \langle D, I \rangle \]

An interpretation \( \| \cdot \|_M \), built up recursively on the basis of the basic interpretation function I, assigns to every expression \( \alpha \) its semantic value \( \| \alpha \|_M \) in a given model \( \text{M} \). (More precisely, \( \| \alpha \|_{M,g} \).) These semantic values must correspond to the types of the expressions. Thus, in our examples to the individual constants John and Mary are assigned certain objects, individual variables take their values in the set of objects (entities), to the predicate constant love is assigned a binary relation \( \| \text{love} \|_M \), and to the predicate constant happy, a unary relation (property) \( \| \text{happy} \|_M \). Formulas receive truth values. The formula love (John, Mary) is true in the model \( \text{M} \) if the pair of objects corresponding to the constants John and Mary belongs to the relation \( \| \text{love} \|_M \).

The formula \( \forall x(\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x)) \) is true in \( \text{M} \) iff:

for every object \( d \) in the domain,
\[ d \in \| \text{happy} \|_M \text{ if } <\| \text{Mary} \|_M, d > \in \| \text{love} \|_M. \]

Restating the last statement more carefully and more generally requires talking about semantic values relative to a model and an assignment g of values to variables.

The notation g[d/x] means: The variable assignment which is identical to g except for the (possible) difference that g[d/x] assigns the individual d to the variable x.

The complication of needing to talk about g[d/x] comes from formulas with more than one variable, like:

\( \forall x\exists y(\text{love}(y, x) \rightarrow \text{happy}(x)) \) and
∃y ∀x (love(y, x) → happy(x)).

So let us restate more carefully, according to the semantics given in Appendix 1, the truth conditions for the formula: \( \forall x (\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x)) \):

\[
\| \forall x (\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x)) \|^{M,g} = 1 \text{ iff :}
\]

for each d in D,
if \(< \| \text{Mary} \|^{M,g[d/x]}, \| x \|^{M,g[d/x]} > 0 \| \text{love} \|^{M,g[d/x]}, \text{ then } d \in \| \text{happy} \|^{M,g[d/x]} \).

For each constant \( \alpha \), \( \| \alpha \|^{M,g[d/x]} = I(\alpha) \). And \( \| x \|^{M,g[d/x]} = g[d/x] (x) = d \). So the condition above is equivalent to:

iff: for each d in D,
if \(< I(\text{Mary}), d> \in I(\text{love}) \) \( \text{ then } d \in I(\text{happy}) \).

**Example**

Let us consider a very simple PC language which has (as in the formulas above) only two constants \textit{John} and \textit{Mary} and two predicate symbols \textit{love} (binary) and \textit{happy} (unary).

Let us consider two models, \( M_1 \) and \( M_2 \):

\[
M_1 = <D, I_1>, D = \{j,m\}, \\
I_1(\text{John}) = j, I_1(\text{Mary}) = m, \\
I_1(\text{love}) = \{<j,j>,<j,m>,<m,m>,<m,j>\}, I_1(\text{happy}) = \{j,m\},
\]

\[
M_2 = <D, I_2>, D = \{j,m\}, \\
I_2(\text{John}) = j, I_2(\text{Mary}) = m, \\
I_2(\text{love}) = \{<j,j>,<m,j>\}, I_2(\text{happy}) = \{m\}.
\]

It is easy to see that both formulas \textit{love (John, Mary)} and \textit{love (Mary, John)} are true in \( M_1 \) but only the second one is true in \( M_2 \).

The formula \( \forall x (\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x)) \) is true in \( M_1 \). But it is false in \( M_2 \), since for the evaluation \( g \) such that \( g(x) = j \) we have \( \| \text{love}(\text{Mary}, x) \|^{M_2,g} = 1 \) and \( \| \text{happy}(x) \|^{M_2,g} = 0 \).

The semantics of PC illustrates the Principle of Compositionality.

As we know the infinite set of formulas of PC are built from terms (individual variables and constants) and predicate symbols by recursive syntactic rules (rules R1—R8 in Appendix 1). The semantics of these formulas – their interpretation in every given model - - is defined by semantic rules S1 – S8, which correspond in a direct way to the syntactic rules. The semantics of the whole is based on the semantics of parts by means of this pairing of semantic interpretation rules with syntactic formation rules. This is a very important feature of every formal language -- The Principle of Compositionality – and it is natural to think that this principle holds also for natural language.

3. **Ambiguity, “Logical form”, and semantically relevant syntax.**

**Lexical ambiguity:** \textit{bank}_1, \textit{bank}_2 : both CN (common noun), homonyms; \textit{open}_1 (ADJ), \textit{open}_2 (IV) (intransitive verb), \textit{open}_3 (TV) (transitive verb).
Structural ambiguity. Compositionality requires a “disambiguated language” (a “language without ambiguity”). So we interpret expressions with syntactic structure, not just strings.

(1) old men and women. Two meanings, two structures. “old” applies only to “men”, or to “men and women”.

(a) 
```
  NP
 /  |
NP and NP
 /   |
CNP CNP
     |  
ADJ CNP
   |   |
old men
```

(b) 
```
  NP
 /  |
CNP
 /  |
ADJ CNP
   |   |
old CNP and CNP
   |     |
men women
```

(2) Every student read a book. (Quantifier scope ambiguity)

Just one (surface) syntactic structure:

```
  S
 /  |
NP VP
 /  |
DET CNP V NP
   |     |
every student read DET CNP
                     a book
```

Predicate logic representations of the two readings:

(i) $\forall x \ (\text{Student}(x) \implies \exists y \ (\text{Book}(y) \& \text{Read}(x,y)))$

(ii) $\exists y \ (\text{Book}(y) \& \forall x \ (\text{Student}(x) \implies \text{Read}(x,y)))$

Compositional interpretation of the English sentence: ?? More next time.

A difficulty for compositionality if we try to use predicate calculus to represent “logical form”: What is the interpretation of “every student”? There is no appropriate syntactic category or semantic type in predicate logic. Inadequacy of 1st-order predicate logic for representing the semantic structure of natural language. We can solve this problem when we have the lambda-calculus and a richer type theory.
Compare the limited inventory of categories we find in PC with the rich range of categories found in natural language:

<table>
<thead>
<tr>
<th>Categories of PC</th>
<th>Categories of NL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula</td>
<td>Sentence</td>
</tr>
<tr>
<td>Predicate</td>
<td>Verb, Common Noun, Adjective</td>
</tr>
<tr>
<td>Term</td>
<td></td>
</tr>
<tr>
<td>Constant</td>
<td>Proper Noun</td>
</tr>
<tr>
<td>Variable</td>
<td>Pronoun (he, she, it)</td>
</tr>
<tr>
<td>(no more)</td>
<td>Verb Phrase, Noun Phrase, Common Noun Phrase, Adjective Phrase, Determiner, Preposition, Prepositional Phrase, Adverb, …</td>
</tr>
</tbody>
</table>

In the next lectures, we will see how a logic built on a richer type theory including the tools of the lambda-calculus can provide a richer formal semantics that can more adequately represent the structure of natural language semantics in a compositional way.

APPENDIX. Syntax and semantics of the predicate calculus (PC).

SYNTAX.
Syntactic Categories: terms (Term), 1-place predicates (Pred-1), 2-place predicates (Pred-2), ..., n-place predicates (Pred-n), formulas (Form).

Basic Expressions:
Basic Term(s): (i) (individual) variables: \(x, y, z, x_1, y_1, z_1, x_2, \ldots\)  
(ii) (individual) constants: \(a, b, c, a_1, John, Mary, \ldots\)  
Basic Pred-1: run, walk, happy, calm, ...  
Basic Pred-2: love, kiss, like, see, ...  
...  
Basic Form(ulas): — (none)

Syntactic Rules:
R1: If \(P \in \text{Pred-1}\) and \(T \in \text{Term}\), then \(P(T) \in \text{Form}\).  
R2: If \(R \in \text{Pred-2}\) and \(T_1, T_2 \in \text{Term}\), then \(R(T_1, T_2) \in \text{Form}\).  
More general rule: If \(R \in \text{Pred-n}\) and \(T_1, \ldots, T_n \in \text{Term}\), then \(R(T_1, \ldots, T_n) \in \text{Form}\).  
R3: If \(\varphi \in \text{Form}\), then \(\neg \varphi \in \text{Form}\).  
R4: If \(\varphi \in \text{Form}\) and \(\psi \in \text{Form}\), then \((\varphi \& \psi) \in \text{Form}\).  
R5: If \(\varphi \in \text{Form}\) and \(\psi \in \text{Form}\), then \((\varphi \lor \psi) \in \text{Form}\).  
R6: If \(\varphi \in \text{Form}\) and \(\psi \in \text{Form}\), then \((\varphi \rightarrow \psi) \in \text{Form}\).  
R7: If \(\nu\) is a variable and \(\varphi \in \text{Form}\), then \(\forall \nu \varphi \in \text{Form}\).  
R8: If \(\nu\) is a variable and \(\varphi \in \text{Form}\), then \(\exists \nu \varphi \in \text{Form}\).  

SEMANTICS.
Model structure:  
Domain \(D\) of entities (individuals)  
Truth values \(\{\text{True, False}\}\) or \(\{1,0\}\)  
I: Interpretation function which assigns semantic values to all constants (in Term and in  
\(\text{Pred-1, Pred-2, ... Pred-n}\))  
\(M = \langle D, I \rangle\)  
Set \(G\) of assignment functions \(g\), functions from variables to \(D\).
Semantic Types assigned to Syntactic Categories:
Term: entities, individuals. The semantic values of this type are the members of D.
Pred-1: sets (of entities). Semantic values of this type are members of \( \wp(D) \).
(\( \wp(D) \) is the power set (the set of all subsets) of D).
Pred-2: relations between entities (sets of pairs). Values: members of \( \wp(D \times D) \).
Pred-n: n-place relations; sets of n-tuples of entities. Values: members of \( \wp(D \times \ldots \times D) \).
Form: Truth values. Values: members of \{0,1\}.

Semantic interpretation relative to \( M, g \):
We use the notation \( \| \phi \|^{M,g} \) for the semantic value of an expression \( \phi \) relative to \( M, g \).

Basic Expressions ("lexical semantics"):
A. If \( \alpha \) is a variable, then \( \| \alpha \|^{M,g} = g(\alpha) \).
B. If \( \alpha \) is a constant, then \( \| \alpha \|^{M,g} = I(\alpha) \).

Semantic Rules ("semantics of syntax"):
S1: If \( P \in \text{Pred-1} \) and \( T \in \text{Term} \), then \( \| P(T) \|^{M,g} = 1 \) iff \( \| T \|^{M,g} \in \| P \|^{M,g} \).
S2: More general rule: If \( R \in \text{Pred-n} \) and \( T_1, \ldots, T_n \in \text{Term} \), then \( \| R(T_1, \ldots, T_n) \|^{M,g} = 1 \) iff
\[ <\| T_1 \|^{M,g}, \ldots, \| T_n \|^{M,g}> \in \| R \|^{M,g} \].
S3: If \( \phi \in \text{Form} \), then \( \| \neg \phi \|^{M,g} = 1 \) iff \( \| \phi \|^{M,g} = 0 \).
S4: If \( \phi, \psi \in \text{Form} \), then \( \| (\phi \& \psi) \|^{M,g} = 1 \) iff \( \| \phi \|^{M,g} = 1 \) and \( \| \psi \|^{M,g} = 1 \).
S5: If \( \phi, \psi \in \text{Form} \), then \( \| (\phi \lor \psi) \|^{M,g} = 1 \) iff \( \| \phi \|^{M,g} = 1 \) or \( \| \psi \|^{M,g} = 1 \).
S6: If \( \phi, \psi \in \text{Form} \), then \( \| (\phi \rightarrow \psi) \|^{M,g} = 1 \) iff \( \| \phi \|^{M,g} = 0 \) or \( \| \psi \|^{M,g} = 1 \).
S7: If \( v \) is a variable and \( \phi \in \text{Form} \), then \( \| \forall v \phi \|^{M,g} = 1 \) iff
there is a \( d \in D \) such that \( \| \phi \|^{M,g[d/\alpha]} = 1 \).
S8: If \( v \) is a variable and \( \phi \in \text{Form} \), then \( \| \exists v \phi \|^{M,g} = 1 \) iff
for all \( d \in D \), \( \| \phi \|^{M,g[d/\alpha]} = 1 \).

[The notation \( g[d/\alpha] \) means: The variable assignment which is identical to \( g \) except for the (possible) difference that \( g[d/\alpha] \) assigns the individual \( d \) to the variable \( \alpha \).]

Truth: Some formulas are true independent of the choice of assignment; those can be called true relative to just \( M \), i.e. simply true on the given interpretation.

If \( \phi \in \text{Form} \), then: \( \| \phi \|^{M} = 1 \) iff for all assignments \( g \), \( \| \phi \|^{M,g} = 1 \).
\( \| \phi \|^{M} = 0 \) iff for all assignments \( g \), \( \| \phi \|^{M,g} = 0 \).
Otherwise \( \| \phi \|^{M} \) is undefined.
Homework #1. Due Feb 27.

This homework has two parts. The first part is directly related to the material in today’s handout. For the first part, I provide background with a fully worked-out answer to one question and then a very similar question for you to do. The second part is designed to review your knowledge of first-order predicate logic and its use to represent (part of) the interpretation of English sentences. The second part has two purposes – for you to get some practice using predicate logic, and for me to get an impression of your background. If some of you have never studied logic before, and this is new to you, (i) don’t panic – we won’t use it every day, and I can help you get additional background for the amount we will use, in group tutorials or individually; and (ii) try the exercises if you can at all, with the assurance that this has no effect on your grade, and will help me know what kind of further help is needed. If you have some specific questions about the problems, feel free to e-mail me or phone or come see me.

Part One.

Background:
Below you will find a syntactic “derivation” tree for the formula \( \forall x (\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x)) \), which expresses the same proposition as the English sentence Everyone who Mary loves is happy. That is followed by a derivation of the truth-conditions of the formula according to the compositional semantic rules of the predicate calculus. Each line is annotated to identify what semantic rule was applied in the derivation of that line, and what node of the syntactic derivation tree it corresponds to. (The problem you are asked to solve is stated after all of that.)

Tree 1.
\[
\forall x (\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x)), \text{ Form, R7}
\]
\[
\text{love}(\text{Mary}, x), \text{ Form, R2}
\]
\[
\text{happy}(x), \text{ Form, R1}
\]
\[
x, \text{ T, Basic}
\]
\[
\text{Mary}, \text{T,Basic}
\]

Annotated semantic derivation of truth conditions:

1. \( \| \forall x (\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x)) \|^{M,g} = 1 \) iff for each \( d \) in \( D \),
   \( \| \text{love}(\text{Mary}, x) \rightarrow \text{happy}(x) \|^{M,g[d/x]} = 1 \). By rule S7 at the “R7” node.

2. That will hold iff for each \( d \) in \( D \),
   \( \| \text{love}(\text{Mary}, x) \|^{M,g[d/x]} = 0 \) or \( \| \text{happy}(x) \|^{M,g[d/x]} = 1 \). By rule S6 at the “R6” node.
3. That will hold iff for each d in D,

\[\text{if } \langle \text{Mary} \rangle^M, \text{d} \rangle^d \in \langle \text{love} \rangle^M, \text{d} \rangle^d, \text{then } \langle x \rangle^M, \text{d} \rangle^d \in \langle \text{happy} \rangle^M, \text{d} \rangle^d.\]

By rule S2 at the R2 node and by S1 at the R1 node.

4. And that will hold iff for each d in D,

\[\text{if } \langle \text{Mary} \rangle^M, \text{d} \rangle^d \in \langle \text{love} \rangle^M, \text{d} \rangle^d, \text{then } \text{d} \in \langle \text{happy} \rangle^M, \text{d} \rangle^d.\]

By rule A (for variables) at the two x nodes.

5. I.e., if \( \text{I(Mary), d} \rangle^d \in \text{I(happy)}.\)

By rule B (for constants) at the nodes for Mary, love, happy.

If we then annotate the syntactic tree above to also show the semantic rule applied at each step, we can see a perfect match between syntactic and semantic rules in the derivation of the form and meaning of the formula.

Tree 2. 

\[\forall x (\text{love(Mary, x}) \rightarrow \text{happy(x)}) , \text{ Form, R7, S7}\]

\[x (\text{love(Mary, x}) \rightarrow \text{happy(x)}) , \text{ Form, R6, S6}\]

\[\text{love(Mary, x)}, \text{ Form, R2, S2} \quad \text{happy(x)}, \text{ Form, R1, S1}\]

\[\text{love, Pred-2, Basic, B} \quad \text{x, T, Basic, A} \quad \text{happy, Pred-1, Basic, B} \quad \text{x, T, Basic, A}\]

\[\text{Mary, T, Basic, B}\]

**Homework problem for you to do:**

The predicate logic formula \(\forall x (\exists y \text{love(x, y}) \rightarrow \text{happy(x)})\) is equivalent to the English sentence *Everyone who loves someone is happy.*

(b) **Draw a syntactic tree** (analogous to Tree 1 above) which shows how that formula is built up from its parts according to the syntactic rules of the predicate calculus (in the Appendix above).

(c) **Give each node a label** that identifies both the syntactic category of the expression it dominates and the number of the syntactic rule by which its immediate constituents were combined (or “Basic”, if that node dominates a basic expression.)

(d) **Work out the truth-conditions** of the formula according to the semantic rules of the predicate calculus, analogous to the step-by-step derivation of truth conditions for the example above. **Annotate each line** by identifying the semantic rule that was applied anywhere within that line (show where), and the node of the tree to
which it corresponds. (According to the principle of compositionality, there should be a perfect match between syntactic rule and semantic rule applied at each node.)

(e) In addition, **further annotate the syntactic tree** by adding to the label of each non-terminal node the number of the *semantic* rule which was used to combine the meanings of the daughter-node expressions to get the meaning of the whole expression dominated by that node. For nodes dominating basic expressions, indicate whether the semantic rule to use is Rule A or Rule B. (If you’ve done it right, there should be a perfect correspondence between syntactic rules and semantic rules applied at a given node, as in Tree 2 above.)

**Part Two.**

See the general notes about these questions in the introduction to the homework.

Use the notation and the predicates and constants given in the handout. Add additional constants as needed. Translate the following English sentences into first-order predicate calculus. If any of the English sentences are ambiguous with respect to their ‘logical structure’, give two different translations that capture their two interpretations. (You are free to assume that D is a domain of humans, so that quantifier expressions like “No one”, “everyone”, etc., can be represented without the need for an additional predicate `person`.)

Feel free to add notes, comments, and questions. There is not always a single right answer, and the meanings of the English sentences are not always clear-cut, e.g. when to interpret “if” as “if and only if”. When in doubt about the correctness of an answer, state your assumptions as clearly as you can, so I can see whether what you wrote in your formula corresponds to how you in fact interpreted the sentence.

1. Everyone loves Mary.
2. John does not love anyone. (Not ambiguous, but there are two equivalent and equally good formulas for it, one involving negation and the existential quantifier, the other involving negation and the universal quantifier. Give both.)
3. Everyone who sees Mary loves Mary.
4. Everyone loves someone. (Ambiguous)
5. Someone loves everyone. (Ambiguous)
7. Someone walks and someone talks.
8. Everyone who walks is calm.
9. No one who runs walks. (Not ambiguous, but same note as for number 2.)
10. Everyone who Mary loves loves someone who is happy.
11. If anyone cheats, he suffers.
12. If anyone cheats, everyone suffers.
13. Anyone who loves everyone loves himself.
14. Mary loves everyone except John. (For this one, you need to add the two-place predicate of identity, “=”. Think of “everyone except John” as “everyone who is not identical to John”.)
15. Redo the translations of sentences 1, 4, 6, and 7, making use of the predicate `person`, as we would have to do if the domain D contains not only humans but cats, robots, and other entities.