

Lecture 1. Introduction to Formal Semantics and Compositionality

1. Formal Semantics: basic ideas	1
1.1. English as a Formal Language.....	1
1.2. Example. Syntax and semantics of the predicate calculus (PC).....	1
1.3. The Principle of Compositionality.....	3
1.4. Model-theoretic Semantics.....	4
2. Linguistic Examples.....	5
2.1. The structure of NPs with restrictive relative clauses.....	5
2.2. Phrasal and sentential conjunction.....	6
2.3. Adjective - noun combinations.....	6
2.4. Quantifier phrases and semantically relevant syntax.....	6
3. A first-order part of the lambda-calculus.....	7
4. Montague's intensional logic, with lambdas and types.....	8
4.1 Introduction.....	8
4.2. Intensional Logic (IL).....	9
4.2.1. Types and model structures.....	9
4.2.2. Atomic expressions ("lexicon"), notation, and interpretation.....	10
4.2.3. Syntactic rules and their model-theoretic semantic interpretation.....	11
5. A closer look at Lambdas.....	12
6. Montague's semantics for Noun Phrases.....	12
REFERENCES.....	13
APPENDIX. Syntax and semantics of the predicate calculus (PC).....	13
"HOMEWORK" No. 0: Participant Questionnaire.....	15
HOMEWORK No. 1.....	15
HELP FOR HOMEWORK #1.....	17

1. Formal Semantics: basic ideas

1.1. English as a Formal Language.

R. Montague 1970, "English as a Formal Language" begins: "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]."

This is the basic thesis of formal semantics. In the first two 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?

1.2. 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 propositions expressed by the sentences *John loves Mary* and *Everyone whom Mary loves is happy* can also be expressed by formulas of PC:

<i>John loves Mary</i>	love (John, Mary)
<i>Everyone whom Mary loves is happy</i>	$\forall x(\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x))$

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 of PC is very simple: individuals, relations of different arities (unary, binary, etc.), and truth-values.

In our examples we use the following expressions:

Expressions	Types
John, Mary	constants
x	variable
happy	unary predicate constant
love	binary predicate constant
love (John, Mary)	formulas
love(Mary, x)	
happy(x)	
$\forall x(\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x))$	
	individuals
	individuals
	unary relations
	binary relations
	truth-values

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* 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.

$$\mathbf{M} = \langle D, I \rangle$$

An *interpretation* M , built up recursively on the basis of the basic interpretation function *I*, assigns to every expression α its *semantic value* α^M in a given model **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 love^M , and to the predicate constant **happy**, a unary relation (property) happy^M . Formulas receive truth values. The formula **love (John, Mary)** is true in the model **M** if the pair of objects corresponding to the constants **John** and **Mary** belongs to the relation love^M .

The formula $\forall x(\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x))$ is true in **M** iff:
 for every object *d* in the domain,
 $d \in \text{happy}^M$ if $\langle \text{Mary}^M, d \rangle \in \text{love}^M$.

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
 $\exists y \forall x (\text{love}(y, x) \rightarrow \text{happy}(x))$.

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

$\forall x (\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x)) \stackrel{M, g}{=} 1$ iff for each d in D , if $\langle \text{Mary} \stackrel{M, g[d/x]}{}$, $d \rangle$
 $\text{love} \stackrel{M, g[d/x]}{}$, then $d \text{ happy} \stackrel{M, g[d/x]}{}$.

1.3. 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.

Generative grammar: there are infinitely many sentences in any natural language, and the brain is finite, so linguistic competence must involve some finitely describable means for specifying an infinite class of sentences. That is a central task of **syntax**.

Semantics: A speaker of a language knows the meanings of those infinitely many sentences, is able to understand a sentence he/she has never heard before or to express a meaning he/she has never expressed before. So for semantics also there must be a finite way to specify the meanings of the infinite set of sentences of any natural language.

A central principle of formal semantics is that the relation between syntax and semantics is compositional.

The Principle of Compositionality: The meaning of an expression is a function of the meanings of its parts and of the way they are syntactically combined.

Each of the key terms in the principle of compositionality is a “theory-dependent” term, and there are as many different versions of the principle as there are ways of specifying those terms. (*meaning, function, parts (syntax)*) See Partee (1984).

Some of the different kinds of things meanings could be in a compositional framework:

(a) (early Katz and Fodor) Representations in terms of semantic features. bachelor: [+HUMAN, +MALE, +ADULT, +NEVER-MARRIED (?!)]. Semantic composition: adding feature sets together. Problems: insufficient structure for the representations of transitive verbs, quantifiers, and many other expressions; unclear status of uninterpreted features.

(b) Representations in a “language of thought” or “conceptual representation”; if semantics is treated in terms of representations, then semantic composition becomes a matter of compositional translation from a syntactic representation to a semantic representation.

(c) The logic tradition: Frege, Tarski, Carnap, Montague. The basic meaning of a sentence is its truth-conditions: to know the meaning of a sentence is to know what the world must be like if the sentence is true. Knowing the meaning of a sentence does not require knowing whether the sentence is *in fact* true; it only requires being able to discriminate between situations in which the sentence is true and situations in which the sentence is false.

Starting from the idea that the meaning of a sentence consists of its truth-conditions, meanings of other kinds of expressions are analyzed in terms of their contribution to the truth-conditions of the sentences in which they occur.

1.4. Model-theoretic Semantics.

In formal semantics, truth-conditions are expressed in terms of truth relative to various parameters — a formula may be true at a given time, in a given possible world, relative to a certain context that fixes speaker, addressee, etc., and relative to a certain assignment of meanings to its atomic “lexical” expressions and of particular values to its variables. For simple formal languages, all of the relevant variation except for assignment of values to variables is incorporated in the notion of truth relative to a *model*. Semantics which is based on truth-conditions is called *model-theoretic*.

Compositionality in the Montague Grammar tradition:

The task of a **semantics** for language L is to provide truth conditions for every well-formed sentence of L, and to do so in a compositional way. This task requires providing appropriate model-theoretic interpretations for the *parts* of the sentence, including the lexical items.

The task of a **syntax** for language L is (a) to specify the set of well-formed expressions of L (of every category, not only sentences), and (b) to do so in a way which supports a compositional semantics. The syntactic part-whole structure must provide a basis for semantic rules which specify the meaning of a whole as a function of the meanings of its parts.

Basic structure:

(1) **Syntactic categories and semantic “types”:** For each syntactic category there must be a uniform semantic type. For example, one could hypothesize that sentences express propositions, nouns and adjectives express properties of entities, verbs express properties of events.

(2) **Basic (lexical) expressions and their interpretation.** Some syntactic categories include basic expressions; for each such expression, the semantics must assign an interpretation of the appropriate type. Within the tradition of formal semantics, most lexical meanings are left unanalyzed and treated as if primitive; Montague regarded most aspects of the analysis of lexical meaning as an empirical rather than formal matter; formal semantics is concerned with the *types* of lexical meanings and with certain aspects of lexical meaning that interact directly with compositional semantics, such as verbal aspect.

(3) **Syntactic and semantic rules.** Syntactic and semantic rules come in pairs: **<Syntactic Rule n, Semantic Rule n>**: in this sense compositional semantics concerns “the semantics of syntax”.

Syntactic Rule n: If α is an expression of category A and β is an expression of category B, then $F_i(\alpha, \beta)$ is an expression of category C. [where F_i is some syntactic operation on expressions]

Semantic Rule n: If α is interpreted as α' and β is interpreted as β' , then $F_i(\alpha, \beta)$ is interpreted as $G_k(\alpha', \beta')$. [where G_k is some semantic operation on semantic interpretations]

(More about Models, worlds, and axioms in Section 5 and in Lecture 3.)

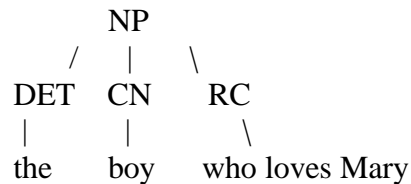
2. Linguistic Examples.

These are examples of some of the kinds of problems that we will be able to solve after we have developed some of the tools of formal semantics. These and other linguistic problems will be discussed in future lectures.

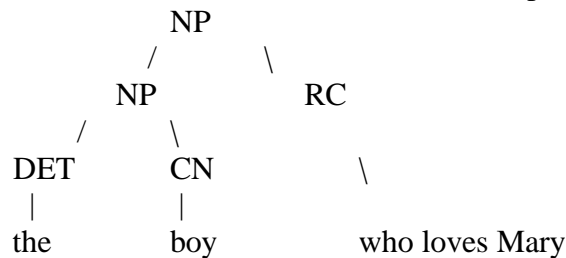
2.1. The structure of NPs with restrictive relative clauses.

Consider NPs such as “the boy who loves Mary”, “every student who dances”, “the doctor who treated Mary”, “no computer which uses Windows”. Each of these NPs has 3 parts: a determiner (DET), a common noun (CN), and a relative clause (RC). The question is: Are there semantic reasons for choosing among three different possible syntactic structures for these NPs?

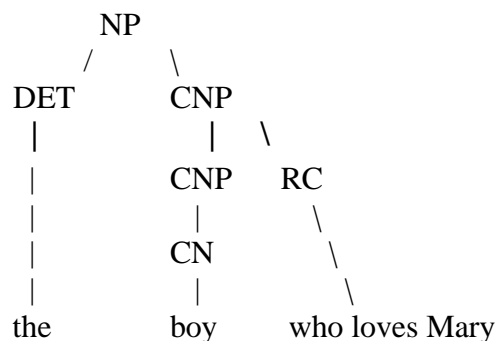
a. Flat structure:



b. “NP - RC” structure: The relative clause combines with a complete NP to form a new NP.



c. “CNP - RC” structure: (CNP: common noun phrase: common noun plus modifiers)



Argument: we will argue that compositionality requires the third structure: that “boy who loves Mary” forms a semantic constituent with which the meaning of the DET combines. We can show that the first structure does not allow for recursivity, and that the second structure cannot be interpreted compositionally.

2.2. Phrasal and sentential conjunction.

Consider the following equivalent and non-equivalent pairs, where the first sentence has phrasal conjunction (VP-conjunction, in particular) and the second has sentential conjunction (S-conjunction). The puzzle is to explain why some examples are semantically equivalent and some are not, although in each case the surface syntactic relation is the same.

<i>John sings and dances</i>	=	<i>John sings and John dances</i>
<i>One boy sings and dances</i>	≠	<i>One boy sings and one boy dances</i>
<i>Every boy sings and dances</i>	=	<i>Every boy sings and every boy dances</i>
<i>No boy sings and dances</i>	≠	<i>No boy sings and no boy dances</i>

We will need two parts to solve this puzzle: (i) the syntax and semantics of sentential and phrasal conjunction, particularly the question of how they are related; and (ii) the semantics of the Determiners *one, every, no* (and others), as well as of simple NPs like *John*.

2.3. Adjective - noun combinations.

Different subclasses of adjectives show different semantic behavior when combining with nouns. For instance:

<i>a red book</i>	is red, and is a book
<i>a skillful doctor</i>	is a doctor; is skillful? – not in an absolute sense, but “as a doctor”
<i>a former teacher</i>	is not a teacher; once was a teacher; is not “former”.
<i>an alleged murderer</i>	may or may not be a murderer

This is a problem that connects lexical and compositional semantics. It is partly a question of the lexical semantics of adjectives, and partly a question of the semantics of the Adjective - Noun construction.

2.4. Quantifier phrases and semantically relevant syntax.

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.

<u>Categories of PC:</u>	<u>Categories of NL:</u>
Formula -	Sentence
Predicate -	Verb, Common Noun, Adjective
Term	
Constant -	Proper Noun
Variable -	Pronoun (he, she, it)
=====	
(no more) -	Verb Phrase, Noun Phrase, Common Noun Phrase, Adjective Phrase, Determiner, Preposition, Prepositional Phrase, Adverb,

3. A first-order part of the lambda-calculus.

To begin looking at the lambda calculus, we will start with just a “first-order” part of it, as if we were just adding a bit of the lambda calculus to the predicate calculus rules in the Appendix. Then in Lecture 2 we will look at the fully typed lambda calculus as included in Montague’s Intensional Logic.

Lambda-abstraction rule, first version.

λ -abstraction applies to *formulas* to make *predicates*. This extends PC in a way that allows us to represent more complex Common Noun Phrases, Adjective Phrases, some Verb Phrases. For some other categories we will need the full version of the λ -abstraction rule.

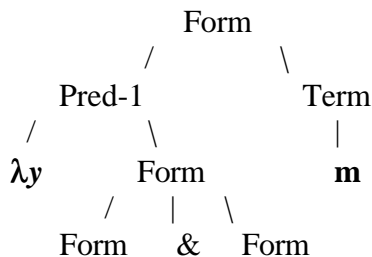
R9: If φ Form and v is a variable, then $\lambda v[\varphi]$ Pred-1.

S9: $\lambda v[\varphi]^{M,g}$ is the set S of all $d \in D$ such that $\varphi^{M,g[d/v]} = 1$.

Examples. For all examples, assume that we start with an assignment g such that $g(v) = \text{John}$ for all v . In most of the examples below, the choice of initial assignment makes no difference. And assume that $I(\mathbf{b}) = \text{Bill}$, $I(\mathbf{m}) = \text{Mary}$.

- (i) $\lambda x[\text{run}(x)]^{M,g} =$ the set of all individuals that run.
- (ii) $\lambda x[\text{love}(x, \mathbf{b})]^{M,g} =$ the set of all individuals that love $\mathbf{b}^{M,g}$, i.e. $I(\mathbf{b})$, i.e. Bill.
- (iii) $\lambda x[\text{love}(x, y)]^{M,g} =$ the set of all individuals that love $y^{M,g}$, i.e. $g(y)$, i.e. John.
- (iv) $\lambda x[\text{fish}(x) \ \& \ \text{love}(x, \mathbf{b})]^{M,g} =$ the set of all fish that love Bill.
- (v) to represent “walks and talks”: $\lambda y[(\text{walk}(y) \ \& \ \text{talk}(y))]$
- (vi) to represent “Mary walks and talks” with constituents that correspond to surface syntax:
 $\lambda y[(\text{walk}(y) \ \& \ \text{talk}(y))](\mathbf{m})$

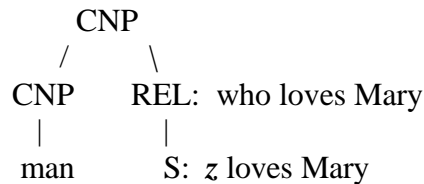
Syntactic Structure of the formula in (vi):



By λ -conversion (see below), the formula in (vi) is equivalent to $(\text{walk}(\mathbf{m}) \ \& \ \text{talk}(\mathbf{m}))$.

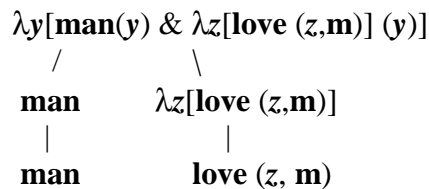
(vii) to represent the CNP “man who loves Mary”:

Syntactic structure:



Rule for combining CNP and REL: $\lambda y[\text{CNP}'(y) \ \& \ \text{REL}'(y)]$ (combining “translations”)

Compositional translation of the syntactic structure above into λ -calculus: (read bottom-to-top)



By λ -conversion (see below), the top line is equivalent to: $\lambda y[\text{man}(y) \ \& \ \text{love}(y, \mathbf{m})]$

Lambda-conversion: A principle concerning the application of λ -expressions to arguments.

Examples: $\lambda x[\text{run}(x)](\mathbf{b})$ $\text{run}(\mathbf{b})$
 $\lambda y[(\text{walk}(y) \ \& \ \text{talk}(y))](\mathbf{m})$ $(\text{walk}(\mathbf{m}) \ \& \ \text{talk}(\mathbf{m}))$
 $\lambda z[\text{love}(z, \mathbf{m})](y)$ $\text{love}(y, \mathbf{m})$

Lambda-conversion Rule: $\lambda v[\alpha](\beta)$ α , where α is like α but with every free occurrence of v replaced by β .

(Note: Occurrences of v that are free in α are bound by λv in $\lambda v[\alpha]$.)

4. Montague’s intensional logic, with lambdas and types.

4.1 Introduction

In this handout we present a small sample English grammar (a “fragment”, in MG terminology), that is, an explicit description of the syntax and semantics of a small part of English.

This fragment is intended to serve several purposes: making certain aspects of formal semantics more explicit, including (and illustrating) more of the basics of the lambda-calculus. The fragment is of interest in its own right and will also serve as background for the next lecture. The fragment, with its very minimal lexicon, also illustrates the typically minimal treatment of the lexicon in classical Montague grammar.

The semantics of the fragment will be given via translation into Montague’s Intensional Logic (IL) (the alternatives would be to give a direct model-theoretic interpretation, or an interpretation via translation into some other model-theoretically interpreted intermediate

language). So we begin by presenting Montague's IL: Its type structure and the model structures in which it is interpreted, and its syntax and model-theoretic semantics.

Then we introduce the fragment of English: first the syntactic categories and the category-type correspondence, then the syntactic rules and the principles of semantic interpretation, and then a small lexicon and some meaning postulates. We conclude with some examples.

Tools like Montague's Intensional Logic are important in making a more satisfactory compositional analysis of natural language semantics possible. What are the differences between Montague's IL and PC? Here are some of the most important:

- (i) The rich type structure of IL.
- (ii) The central role played by function-denoting expressions. All of the types except the basic types *e* and *t* are *functional types*, and all of the expressions of IL except those of types *e* and *t* are expressions which denote functions. Functions may serve as the arguments and as the values of other functions. In particular, all relations are also represented as functions.
- (iii) The inclusion of the operation of "functional application" or "function-argument application", the application of a function to its argument.
- (iv) The use of lambda-expressions. The lambda-operator is the basic tool for building expressions which denote functions.
- (v) In place of the one "world" of PC (where there is in effect no distinction between a "world" and a model), the models of IL include a set of *possible worlds*. Possible worlds are crucially connected with intension/extension distinction and with intensional types. Possible worlds, in particular, underlie the interpretation of modal operators and referential opacity.
- (vi) The models of IL also include, in one way or another, a structure of time, used among other things in the interpretation of tense operators like **PAST** in the fragment below.

4.2. Intensional Logic (IL).

4.2.1. Types and model structures.

4.2.1.1. Types

Montague's IL is a *typed* intensional language; unlike the predicate calculus, which has variables of only one type (the type of entities or individuals), and expressions only of the types of individuals, truth-values, and n-ary relations over individuals, IL has a rich system of types which makes it much easier to achieve a (relatively) close fit between expressions of various categories of a natural language and expressions of IL. The types serve as syntactic categories for the expressions of IL; because of the role of IL as an intermediate language in the semantic interpretation of natural language, the same types are referred to as *semantic types* for expressions of natural language.

The types of Montague's IL are as follows:

Basic types: *e* (entities), *t* (truth values)

Functional types: If *a, b* are types, then $\langle a, b \rangle$ is a type (the type of functions from *a*-type things to *b*-type things.) **Note:** We use interchangeably the two notations $\langle a, b \rangle$ and $a \rightarrow b$, both of which are common in the literature.

Intensional types: If *a* is a type, then $\langle s, a \rangle$ is a type (the type of functions from possible worlds to things (extensions) of type *a*.)

(In some systems, the basic type *t* is taken as intensional, interpreted as the type of propositions rather than of truth-values. In general, we will mostly ignore intensionality in these lectures, working most of the time with extensional versions of our fragments and

mentioning intensionality only where directly relevant. But that is only for simplicity of exposition; in general, a thoroughly intensional semantics is presupposed.)

4.2.1.2. Model structures.

In the first lecture, we introduced a simple model $\mathbf{M} = \langle D, I \rangle$ for interpreting the predicate calculus. A model \mathbf{M} for the typed intensional logic IL has much more structure, but that structure is built up recursively from a small set of primitives.

Model for IL: $\mathbf{M} = \langle D, W, \leq, I \rangle$. Each model must contain:

A domain D of entities (individuals)

A set W of possible worlds (or possible world-time pairs, or possible situations)

\leq : an ordering (understood as temporal order) on W

I : Interpretation function which assigns semantic values to all constants.

[The part $\langle D, W, \leq \rangle$ is the *model structure* and $\langle D, W, \leq, I \rangle$ is an *interpreted model*.]

The domains of possible denotations for expressions of type a (relative to D, W) are defined recursively as follows:

$\mathbf{D}_e = D$

$\mathbf{D}_t = \{0, 1\}$

$\mathbf{D}_{\langle a, b \rangle} = \{f \mid f: \mathbf{D}_a \rightarrow \mathbf{D}_b\}$ (i.e. the set of all functions f from \mathbf{D}_a to \mathbf{D}_b .)

[$\mathbf{D}_{\langle s, a \rangle} = \{f \mid f: W \rightarrow \mathbf{D}_a\}$ (i.e. the set of all functions f from W to \mathbf{D}_a .)] *[not to be discussed here]*

The semantic interpretation of IL also makes use of a set G of assignment functions g , functions from variables of all types to values in the corresponding domains.

Each expression of IL has an *intension* and, at each w in W , an *extension*. The intension is relative to \mathbf{M} and g ; the extension is relative to \mathbf{M} , w , and g . But we will not discuss intensions and extensions in this lecture.

4.2.2. Atomic expressions (“lexicon”), notation, and interpretation.

The atomic expressions of IL are constants and variables; there are infinitely many constants and infinitely many variables in each type. Montague introduced a general nomenclature for constants and variables of a given type, using c and v with complex subscripts indicating type and an index. In practice, including Montague’s, more mnemonic names are used. Our conventions will be as follows:

Constants of IL will be written in **non-italic boldface**, and their names will usually reflect the English expressions of which they are translations: **man, love**, etc. Their types will be specified. Variables of IL will be written in **italic boldface**, usually observing the following conventions as to types:

Type e : ***w, x, y, z***, with and without subscripts or primes (this modifier holds for all types.)

Type $\langle e, t \rangle$: ***P, Q***

Various relational types such as $\langle e, \langle e, t \rangle \rangle$: ***R***

The type of generalized quantifiers: ***T***

The interpretation of constants is given by the interpretation function I of the model, and the interpretation of variables by an assignment g , as specified in Rule 1 below.

4.2.3. Syntactic rules and their model-theoretic semantic interpretation

The syntax of IL takes the form of a recursive definition of the set of “meaningful expressions of type a ”, ME_a , for all types a . The semantics gives an interpretation rule for each syntactic rule.

Note: when giving syntactic and semantic rules for IL, as for predicate logic, we use a metalanguage which is very similar to IL; but we are not boldfacing the constants and variables of the metalanguage. The metalanguage variables over variables are most often chosen as u or v .

The first rule is a rule for atomic expressions, and the first semantic rule is its interpretation:

Syntactic Rule 1: Every constant and variable of type a is in ME_a .

Semantic Rule 1: (a) If α is a constant, then $5\alpha 5^{M,w,g} = I(\alpha)(w)$.
(b) If α is a variable, then $5\alpha 5^{M,w,g} = g(\alpha)$.

Note: The recursive semantic rules give *extensions* relative to model, world, and assignment. Read “ $5\alpha 5^{M,w,g}$ ” as “the semantic value (extension) of alpha relative to M , w , and g .” The interpretation function I assigns to each constant an *intension*, i.e. a function from possible worlds to extensions; applying that function to a given world w gives the extension.

Syntactic Rule 2. (logical connectives and operators that apply to formulas, mostly from propositional and predicate logic, plus some modal and tense operators.) If $\phi, \psi \in ME_t$, and u is a variable of any type, then $\neg\phi$, $\phi \& \psi$, $\phi \vee \psi$, $\phi \rightarrow \psi$, $\phi \leftrightarrow \psi$ (also written as $\phi \equiv \psi$), $\exists u\phi$, $\forall u\phi$, ϕ , **PAST** $\phi \in ME_t$. Note: “ ϕ ” is read as “Necessarily phi”.

Semantic Rule 2:

(a) $\neg\phi$, $\phi \& \psi$, $\phi \vee \psi$, $\phi \rightarrow \psi$, $\phi \leftrightarrow \psi$ (also written as $\phi \equiv \psi$), $\exists u\phi$, $\forall u\phi$ as in predicate logic.
(b) $5\phi 5^{M,w,g} = 1$ iff $5\phi 5^{M,w',g} = 1$ for all w' in W .
(c) **5PAST** $\phi 5^{M,w,g} = 1$ iff $5\phi 5^{M,w',g} = 1$ for some $w' \prec w$. (This is a simplification; here we are treating each w as a combined “world/time index”, possibly a situation index; $w' \prec w$ if w' is a temporally earlier slice of the same world as w .)

Syntactic Rule 3: (=): If $\alpha, \beta \in ME_a$, then $\alpha = \beta \in ME_t$.

Semantic Rule 3: $5\alpha = \beta 5^{M,w,g} = 1$ iff $5\alpha 5^{M,w,g} = 5\beta 5^{M,w,g}$.

(The next two pairs of rules, concerning the “up” and “down” operators, are crucial for intensionality, but we will not discuss them and will not use them.)

Syntactic Rule 4: (“up”-operator.) If $\alpha \in ME_a$, then $[\alpha] \in ME_{\langle s,a \rangle}$.

Semantic Rule 4: $5 [\alpha] 5^{M,w,g}$ is that function h of type $\langle s,a \rangle$ such that for any w' in W ,
 $h(w') = 5\alpha 5^{M,w',g}$.

Syntactic Rule 5: (“down”-operator.) If $\alpha \in ME_{\langle s,a \rangle}$, then $[\alpha] \in ME_a$.

Semantic Rule 5: $5 [\alpha] 5^{M,w,g}$ is $5\alpha 5^{M,w,g}(w)$

The next two pairs of rules, function-argument application and lambda-abstraction, are among the most important devices of IL, and we will make repeated use of them.

Function-argument application:

Syntactic Rule 6: If $\alpha \in ME_{\langle a,b \rangle}$ and $\beta \in ME_a$, then $\alpha(\beta) \in ME_b$.

Semantic Rule 6: $5\alpha(\beta) 5^{M,w,g} = 5\alpha 5^{M,w,g}(5\beta 5^{M,w,g})$

Lambda-abstraction:

Syntactic Rule 7: If $\alpha \in ME_a$ and u is a variable of type b , then $\lambda u[\alpha] \in ME_{\langle b,a \rangle}$.

Semantic Rule 7: $5\lambda u [\alpha] 5^{M,w,g}$ is that function f of type $b \rightarrow a$ such that for any object d of type b , $f(d) = 5\phi 5^{M,w,g[d/u]}$.

5. A closer look at Lambdas

Lambda-abstraction, full version.

In general: λ -expressions denote *functions*.

$\lambda v[\alpha]$ denotes a function whose *argument* is represented by the variable v and whose *value* for any given value of v is specified by the expression α .

Example: $\lambda x[x^2 + 1]$ denotes the function $x \rightarrow x^2 + 1$.

Function-argument application: $\lambda x[x^2 + 1](5) = 26$

λ -expressions provide explicit specification of the functions they name, unlike arbitrary names like f, g . (The λ -calculus was invented by the logician Alonzo Church. The programming language LISP, invented by John McCarthy, was modelled on the λ -calculus.)

Syntactic and Semantic Rule: (a restatement of Syntactic and Semantic Rules 7 of IL)

R7': If α is an expression of any type a and v is a variable of type b , then $\lambda v[\alpha]$ is an expression of type $b \rightarrow a$ (the type of functions from b -type things to a -type things.)

S7': $\lambda v[\alpha]$ ^{M, g} is that function f of type $b \rightarrow a$ such that for any object d of type b , $f(d) = \alpha$ ^{M, g[d/v]}.

Lambda-conversion: A principle concerning the application of λ -expressions to arguments.

Examples: $\lambda x[x^2 + 1](5) = 5^2 + 1 = 26$

$\lambda x[\text{run}(x)](\mathbf{b})$ **run(b)**

$\lambda y[(\text{walk}(y) \ \& \ \text{talk}(y))](\mathbf{m})$ **(walk(m) & talk(m))**

$\lambda z[\text{love}(z, \mathbf{m})](\mathbf{y})$ **love(y, m)**

Lambda-conversion Rule: $\lambda v[\alpha](\beta)$ α , where α is like α but with every free occurrence of v replaced by β .

(Note: Occurrences of v that are free in α are bound by λv in $\lambda v[\alpha]$.)

Note: There are different punctuation conventions for lambda-expressions; in case of confusion, draw a parse tree to be clear about scopes and function-argument relations. The Heim & Kratzer textbook uses the notation $[\lambda v.\alpha]$ instead of $\lambda v[\alpha]$; that is better in showing that the whole λ -expression is a constituent. But the notation of the handouts in this seminar is $\lambda v[\alpha]$. (We may be inconsistent when we write on the blackboard; feel free to ask or to correct us.)

6. Montague's semantics for Noun Phrases.

Indicated by translations of English expressions into the λ -calculus. P is a variable ranging over sets, i.e. a predicate variable. (In full Montague grammar with intensionality, the analysis uses variables over properties, so sometimes in discussion in class, we talk as though P were a variable over properties.)

John $\lambda P[P(\mathbf{j})]$

John walks $\lambda P[P(\mathbf{j})](\text{walk})$ **walk(j)**

every student $\lambda P[\forall x (\text{student}(x) \rightarrow P(x))]$

every student walks $\lambda P[\forall x (\text{student}(x) \rightarrow P(x))](\text{walk})$ $\forall x (\text{student}(x) \rightarrow \text{walk}(x))$

a student $\lambda P[\exists x (\text{student}(x) \ \& \ P(x))]$

the king $\lambda P[\exists x (\text{king}(x) \ \& \ \forall y (\text{king}(y) \rightarrow y = x) \ \& \ P(x))]$

(the set of properties which the one and only king has)

...

REFERENCES.

- Frege, Gottlob (1892) "Über Sinn und Bedeutung", *Zeitschrift für Philosophie und philosophische Kritik* 100, 25-50. Translated as "On sense and reference", in P.T. Geach and M.Black, eds., *Translations from the Philosophical Writings of Gottlob Frege*. Oxford: Blackwell (1952), 56-78.
- Gamut, L.T.F. (1991), *Logic, Language, and Meaning. Vol I: Introduction to Logic; Vol II: Intensional Logic and Logical Grammar*. University of Chicago Press, Chicago and London.
- Lewis, David (1970) "General semantics" *Synthese* 22, 18-67; reprinted in D.Davidson and G.Harman (eds.), *Semantics of Natural Language*. Dordrecht: Reidel (1972), 169-218.
- Montague, Richard (1970b) "English as a Formal Language", in B. Visentini et al, eds. *Linguaggi nella Società e nella Tecnica*. Milan: Edizioni di Comunita; reprinted in Montague (1974) 188-221.
- Montague, Richard (1970c) "Universal Grammar", *Theoria* 36, 373-398; reprinted in Montague (1974) 222-246.
- Montague, R. (1973) "The Proper Treatment of Quantification in Ordinary English," in K.J.J. Hintikka, J.M.E. Moravcsik, and P. Suppes, eds., *Approaches to Natural Language*, Reidel, Dordrecht, 221-242, reprinted in Montague (1974) 247-270.
- Montague, Richard (1974) *Formal Philosophy: Selected Papers of Richard Montague*. Edited and with an introduction by Richmond Thomason, New Haven: Yale Univ. Press.
- Partee, Barbara H. (1984) "Compositionality", in F. Landman and F. Veltman, eds., *Varieties of Formal Semantics: Proceedings of the 4th Amsterdam Colloquium, Sept. 1982*. Foris Pubs., Dordrecht, 281-311.
- Partee, Barbara (1996) "The development of formal semantics in linguistic theory", in Shalom Lappin, ed., *The Handbook of Contemporary Semantic Theory*, Blackwell Handbooks in Linguistics Series, Oxford: Blackwell, 11-38.
- Partee, B., A. ter Meulen, and R.E. Wall (1990) *Mathematical Methods in Linguistics*, Dordrecht: Kluwer Academic Publishers.

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: $x, y, z, x_1, y_1, z_1, x_2, \dots$
 (ii) (individual) constants: **a,b,c, a₁, .John, Mary, ...**
- 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 $T_1, \dots, T_n \in \text{Term}$, then $R(T_1, \dots, T_n) \in \text{Form}$

R3: If $\phi \in \text{Form}$, then $\neg\phi \in \text{Form}$.

R4: If $\phi \in \text{Form}$ and $\psi \in \text{Form}$, then $(\phi \ \& \ \psi) \in \text{Form}$.

R5: If $\phi \in \text{Form}$ and $\psi \in \text{Form}$, then $(\phi \ \vee \ \psi) \in \text{Form}$.

R6: If $\phi \in \text{Form}$ and $\psi \in \text{Form}$, then $(\phi \rightarrow \psi) \in \text{Form}$.

R7: If v is a variable and $\phi \in \text{Form}$, then $\forall v\phi \in \text{Form}$.

R8: If v is a variable and $\phi \in \text{Form}$, then $\exists v\phi \in \text{Form}$.

SEMANTICS.

Model structure: Domain D of entities (individuals)
 Truth values {True, False} or {1,0}
 I: Interpretation function which assigns semantic values to all constants
 (in Term and in Pred-1, Pred-2, ... Pred-n)
 $\mathbf{M1} = \langle D, I \rangle$
 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 $\wp(D \times \dots \times D)$.
 Form: Truth values. Values: members of {0,1}.

Semantic interpretation relative to $\mathbf{M1}, g$:

We use the notation $\varphi^{M1,g}$ for the semantic value of an expression φ relative to $\mathbf{M1}, g$.

Basic Expressions (“lexical semantics”):

- A. If α is a variable, then $\alpha^{M1,g} = g(\alpha)$.
- B. If α is a constant, then $\alpha^{M1,g} = I(\alpha)$.

Semantic Rules (“semantics of syntax”):

- S1:** If $P \in \text{Pred-1}$ and $T \in \text{Term}$, then $P(T)^{M1,g} = 1$ iff $T^{M1,g} \in P^{M1,g}$.
- S2:** More general rule: If $R \in \text{Pred-n}$ and $T_1, \dots, T_n \in \text{Term}$, then $R(T_1, \dots, T_n)^{M1,g}$ iff $\langle T_1^{M1,g}, \dots, T_n^{M1,g} \rangle \in R^{M1,g}$.
- S3:** If $\varphi \in \text{Form}$, then $\neg\varphi^{M1,g} = 1$ iff $\varphi^{M1,g} = 0$.
- S4:** If $\varphi, \psi \in \text{Form}$, then $(\varphi \ \& \ \psi)^{M1,g} = 1$ iff $\varphi^{M1,g} = 1$ and $\psi^{M1,g} = 1$.
- S5:** If $\varphi, \psi \in \text{Form}$, then $(\varphi \ \vee \ \psi)^{M1,g} = 1$ iff $\varphi^{M1,g} = 1$ or $\psi^{M1,g} = 1$.
- S6:** If $\varphi, \psi \in \text{Form}$, then $(\varphi \ \rightarrow \ \psi)^{M1,g} = 1$ iff $\varphi^{M1,g} = 0$ or $\psi^{M1,g} = 1$.
- S7:** If v is a variable and $\varphi \in \text{Form}$, then $\forall v\varphi^{M1,g} = 1$ iff for all $d \in D$, $\varphi^{M1,g[d/v]} = 1$.
- S8:** If v is a variable and $\varphi \in \text{Form}$, then $\exists v\varphi^{M1,g} = 1$ iff there is a $d \in D$ such that $\varphi^{M1,g[d/v]} = 1$.

[**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 .]

Truth: Some formulas are true independent of the choice of assignment; those can be called true relative to just $\mathbf{M1}$, i.e. simply true on the given interpretation.

If $\varphi \in \text{Form}$, then: $\varphi^{M1} = 1$ iff for all assignments g , $\varphi^{M1,g} = 1$.
 $\varphi^{M1} = 0$ iff for all assignments g , $\varphi^{M1,g} = 0$.

Otherwise φ^{M1} is undefined.

“HOMEWORK” No. 0: Participant Questionnaire.

Please answer the following questions for us very briefly -- no more than 2 pages total.

1. Your name.
2. “Who and where” – Ph.D. student in linguistics? or? –and what is your home institution?
3. How much / what kind of the following are part of your background?
 - a. Semantics
 - b. Syntax
 - c. Logic
 - d. Mathematics
4. What languages do you know and/or have you done linguistic work on?
5. In one or two sentences, your specialty and main interests.
6. In one or two sentences, why you are taking this course and what you hope to learn from it.

HOMEWORK No. 1.

Do all three problems. 2 pages should be enough; 3 pages maximum. There is an extra page of “homework help” for this homework. First try to do it without looking at the help, then look at the help, and then try it again if you did it wrong the first time. Please come see us during consultation sessions, especially if you are unsure of anything, but also with any comments or questions related to either the homework or the lecture.

1. The predicate logic formula $\forall x(\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x))$ is equivalent to the English sentence *Everyone who Mary loves is happy*. Draw a syntactic tree (analogous to the tree-fragment on page 7 for formula (vi)), which shows how that formula (*not* the English sentence!) is built up from its parts according to the syntactic rules of the predicate calculus (in the Appendix above).
 - (a) 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.)
 - (b) Below is a derivation of the truth-conditions of the formula according to the semantic rules of the predicate calculus. 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.)
 - (c) 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.)

Semantic derivation of truth conditions:

1. $\forall x(\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x))$ $M1.g = 1$ iff for each d in D ,

$$\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x) \quad M1.g[d/x] = 1.$$

2. That will hold iff for each d in D ,

$$\text{love}(\text{Mary}, x) \quad M1.g[d/x] = 0 \quad \text{or} \quad \text{happy}(x) \quad M1.g[d/x] = 1 .$$

3. That will hold iff for each d in D ,

if $\langle \text{Mary}^{M1,g[d/x]}, x^{M1,g[d/x]} \rangle \text{love}^{M1,g[d/x]}$, then $x^{M1,g[d/x]} \text{happy}^{M1,g[d/x]}$.

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

if $\langle \text{Mary}^{M1,g[d/x]}, d \rangle \text{love}^{M1,g[d/x]}$, then $d \text{happy}^{M1,g[d/x]}$.

5. I.e., if $\langle I(\text{Mary}), d \rangle I(\text{love})$, then $d I(\text{happy})$.

2. (a) Write down the translation into the λ -calculus of “A student walks and talks”. This handout already shows the translations of “a student” and “walks and talks”. Put them together by “function-argument application”.

(b) Apply λ -conversion to simplify the formula. There will be two applications, and the resulting formula should have no λ 's.

(c) Write down the translation of “A student walks and a student talks”; simplify by λ -conversion.

(d) The two formulas (if you did parts (a-c) correctly) are not equivalent. Describe a situation (a model) in which one of them is true and the other one is false.

3. In the predicate calculus, the sentence “No student talks” can be represented as follows:

$\neg \exists x [\text{student}(x) \ \& \ \text{talk}(x)]$ or equivalently as $\forall x [\text{student}(x) \rightarrow \neg \text{talk}(x)]$

But in the predicate calculus, there is no way to represent the meaning of the NP “no student”. Using the λ -calculus in the way illustrated above for the NPs “every student”, “a student”, “the king”, write down a translation for the NP “no student”. (There are two logically equivalent correct answers; write down either or both.)

When you do homework, feel free to write questions on your homework paper.

Help with Problem 2. Instead of giving an answer, here is an answer to a similar problem, which we'll call Problem 2*.

Problem 2*. (a) Write down the translation into the λ -calculus of "Every student walks and talks". The handout already shows the translations of *every student* and *walks and talks*. Just put them together by function-argument application.

(b) Apply λ -conversion to simplify the formula. There will be two applications, and the resulting formula should have no λ 's.

(c) Write down the translation of "Every student walks and every student talks"; simplify by λ -conversion.

(d) Are the two formulas equivalent? Give an argument.

Answer.

(2*) (a) *every student* : $\lambda P[\forall x (\text{student}(x) \rightarrow P(x))]$

walks and talks: $\lambda y[(\text{walk}(y) \ \& \ \text{talk}(y))]$

every student walks and talks: $\lambda P[\forall x (\text{student}(x) \rightarrow P(x))] (\lambda y[(\text{walk}(y) \ \& \ \text{talk}(y))])$

(b) Simplify the expression by two applications of λ -conversion.

Step 1: $\lambda P[\forall x (\text{student}(x) \rightarrow P(x))] (\lambda y[(\text{walk}(y) \ \& \ \text{talk}(y))]) \equiv$

$\forall x (\text{student}(x) \rightarrow \lambda y[(\text{walk}(y) \ \& \ \text{talk}(y))] (x))$

Step 2: $\forall x (\text{student}(x) \rightarrow \lambda y[(\text{walk}(y) \ \& \ \text{talk}(y))] (x)) \equiv$

$\forall x (\text{student}(x) \rightarrow (\text{walk}(x) \ \& \ \text{talk}(x)))$

(c) *Every student walks and every student talks*:

$\lambda P[\forall x (\text{student}(x) \rightarrow P(x))] (\text{walk}) \ \& \ \lambda P[\forall x (\text{student}(x) \rightarrow P(x))] (\text{talk})$

$\equiv \forall x (\text{student}(x) \rightarrow \text{walk}(x)) \ \& \ \forall x (\text{student}(x) \rightarrow \text{talk}(x))$

It doesn't matter whether the same variable x is used in both formulas or not; this is equivalent to: $\forall x (\text{student}(x) \rightarrow \text{walk}(x)) \ \& \ \forall y (\text{student}(y) \rightarrow \text{talk}(y))$

(d) Use first-order predicate logic to argue that the last formula in (b) and the last formulas in (c) are equivalent. Both formulas require that every student have both properties.