1 Logic, language and meaning

- A **formal system** is a set of primitives, some statements about the primitives (axioms), and some method of deriving further statements about the primitives from the axioms.
- Predicate logic (calculus) is a formal system consisting of:
  1. A syntax defining the expressions of a language.
  2. A set of axioms (formulae of the language assumed to be true)
  3. A set of rules of inference for deriving further formulas from the axioms.
- We use this formal system as a tool for analyzing relevant aspects of the meanings of natural languages.
- A formal system is a syntactic object, a set of expressions and rules of combination and derivation. However, we can talk about the relation between this system and the models that can be used to interpret it, i.e. to assign extra-linguistic entities as the meanings of expressions.
- Logic is useful when it is possible to translate natural languages into a logical language, thereby learning about the properties of natural language meaning from the properties of the things that can act as meanings for a logical language.

2 Syntax and semantics of Predicate Logic

2.1 The vocabulary of Predicate Logic

1. Individual constants: $\{d, n, j, ...\}$
2. Individual variables: $\{x, y, z, ...\}$ The individual variables and constants are the *terms.*
3. Predicate constants: $\{P, Q, R, ...\}$ Each predicate has a fixed and finite number of arguments called its *arity* or *valence.* As we will see, this corresponds closely to argument positions for natural language predicates.
4. Connectives: $\{\neg, \vee, \wedge, \rightarrow, \leftrightarrow\}$
5. Quantifiers: $\{\forall, \exists\}$
6. Constituency labels: parentheses, square brackets and commas.

2.2 Example: A toy language L_x

Let us consider a very simple Predicate Logic language with basic expressions of three categories: names, one-place predicates, and two-place predicates.

(1) **Basic expressions of L_x**

Category	**Basic expressions**	**NL counterpart**
Names	d, n, j, m	David, Nancy, John, Mary
One place (unary) predicates	H, C	happy, cries
Two place (binary) predicates	D, L	dislike, love

(2) **Syntactic rules of L_x**

a. If δ is a one place predicate and α is a name, then $\delta(\alpha)$ is a formula.

b. If γ is a two place predicate and α and β are names then $\gamma(\alpha,\ \beta)$ is a formula.

c. If ϕ is a formula. then $\neg\phi$ is a formula.

d. If ϕ and ψ are formulas then $[\phi \wedge \psi]$ is a formula.

e. If ϕ and ψ are formulas then $[\phi \vee \psi]$ is a formula.

f. If ϕ and ψ are formulas then $[\phi \rightarrow \psi]$ is a formula.

g. If ϕ and ψ are formulas then $[\phi \leftrightarrow \psi]$ is a formula.

We may also stipulate that square brackets can be omitted from "matrix" formulas. Exx:

(3) a. $C(j)$

b. $D(n, d)$

c. $L(n, j) \vee C(j)$

d. $*\, C(j) \vee L(n, j) \rightarrow [H(j) \rightarrow [C(j) \vee L(n, j)]]$

e. $*\, C(j \wedge d)$

2.3 A semantics for L_x

Expressions are interpreted in models. A model M is a pair $\langle D, I\rangle$, where D is the domain, a set of individuals, and I is an interpretation function: an assignment of semantic values to every basic expression (constant) in the language.

Models are distinguished both by the objects in their domains and by the values assigned to the expressions of the language by I — by the particular way that the words of the language are "linked" to the things in the world. For example:

(4) $M_1 = \langle D_1, I_1\rangle$, where:

a. $D_1 = \{$David, Nancy, John, Mary$\}$

b. I_1 determines the following mapping mapping between names and predicate terms in L_x and objects in D_1:

name	**value**	**predicate**	**value**
d	David	H	{Nancy, Mary}
n	Nancy	C	{Nancy, Mary, David}
j	John	D	{⟨ Mary, David ⟩, ⟨ Nancy, David ⟩ }
m	Mary	L	{ ⟨ Nancy, John⟩, ⟨ David, John⟩, ⟨ Mary, John⟩ }

The interpretation of an arbitrary expression α relative to M, $[\![\alpha]\!]^M$, is built up recursively on the basis of the basic interpretation function I and a set of COMPOSITION RULES: the "engine" of the semantics.

For example, the composition rules in (5) provide TRUTH CONDITIONAL meanings for the complex expressions of L_x: the interpretation of a formula (sentence) of the language is a truth value (1 = "true" and 0 = "false"), where truth values of particular sentences are ultimately determined by the model (via the predicate rules in (5a-b)).

(5) **Composition rules of L_x**

a. If δ is a one place predicate and α is a name, then $[\![\delta(\alpha)]\!]^M = 1$ iff $[\![\alpha]\!]^M \in [\![\delta]\!]^M$.

b. If γ is a two place predicate and α and β are names, then $[\![\gamma(\alpha,\beta)]\!]^M = 1$ iff $\langle [\![\alpha]\!]^M, [\![\beta]\!]^M \rangle \in [\![\gamma]\!]^M$.

c. If ϕ is a formula, then $[\![\neg\phi]\!]^M = 1$ iff $[\![\phi]\!]^M = 0$.

d. If ϕ and ψ are formulas then $[\![[\phi \wedge \psi]]\!]^M = 1$ iff both $[\![\phi]\!]^M = 1$ and $[\![\psi]\!]^M = 1$.

e. If ϕ and ψ are formulas then $[\![[\phi \vee \psi]]\!]^M = 1$ iff at least one of $[\![\phi]\!]^M$, $[\![\psi]\!]^M = 1$.

f. If ϕ and ψ are formulas then $[\![[\phi \rightarrow \psi]]\!]^M = 1$ iff either $[\![\phi]\!]^M = 0$ or $[\![\psi]\!]^M = 1$.

g. If ϕ and ψ are formulas then $[\![[\phi \leftrightarrow \psi]]\!]^M = 1$ iff both $[\![\phi]\!]^M$ and $[\![\psi]\!]^M = 1$ or if both $[\![\phi]\!]^M$ and $[\![\psi]\!]^M = 0$.

Exercise: determine the semantic values of the (well-formed) formulas in (3).

2.4 Quantifiers and variables

Predicate logic, in addition to the individual and predicate constants, and connectives, includes individual variables and the existential and universal quantifiers. x, y, z are variables that range over individuals. Variables enable us to refer to individuals without referring to any particular individual.

(6) **The syntax of quantifiers in L_x**
If ϕ is a formula and x is an individual variable, then $\forall x\phi$ and $\exists x\phi$ are formulas.

For example:

(7) a. $\exists x Q(x)$

b. $\forall x[P(x) \rightarrow \neg Q(x)]$

c. $\forall x \exists y[P(j) \rightarrow R(n,j)]$

Some important terminology:

(8) a. If x is a variable and ϕ a formula to which a quantifier is attached, than ϕ is the **scope** of the quantifier.
b. A variable x is **bound** if it occurs in the scope of $\forall x$ or $\exists x$.
c. A variable x that does not occur in the scope of $\forall x$ or $\exists x$ is **free**.
d. Every variable is either free or bound and if bound, it is bound exactly once.
e. A **closed formula** is one that does not contain any free variables.
f. Any well-formed formula of predicate logic which contains a free individual variable is an **open formula**.

So far we have assumed that interpretations of basic expressions are given by I, which assigns values in D to names and predicates. The interpretation of individual variables requires a further semantic component, called an ASSIGNMENT FUNCTION, notated g. The assignment function assigns individuals in D to individual variables in formulas. For example, (9) shows three different assignment functions for the variables in L_x and the domain D_1.

(9)

	variable	value		variable	value		variable	value
$\mathbf{g}_1$:	x	David	$\mathbf{g}_2$:	x	Mary	$\mathbf{g}_3$:	x	David
	y	Nancy		y	Nancy		y	David
	z	John		z	John		z	David

The interpretation of complex expressions of L_x (and Predicate Logic in general) are thus relative to both a model and an assignment function: $[\![\alpha]\!]^{M,g}$ is the semantic value of α relative to model M and assignment g. Other than relativization to assignment functions, the composition rules defined in (5) remain the same.

Assignment functions also play a role in the interpretation of formulas with quantifiers, though in a somewhat special way. To see how, we first need some notation:

(10) For any assignment function g, variable x and object $d \in D$, $g[d/x]$ is an assignment function that is just like g except that the value it assigns to x is d.

For example, $g_1[Mary/x]$ looks just like g_2 in (9). With this notation in hand, we can add the following new composition rules to handle formulas with quantifiers:

(11) **Composition rules for quantifiers in L_x**
a. If ϕ is a formula then $[\![\forall x\phi]\!]^{M,g} = 1$ iff $[[\phi]]^{M,g[d/x]} = 1$ for all $d \in$ D.
b. If ϕ is a formula then $[\![\exists x\phi]\!]^{M,g} = 1$ iff $[[\phi]]^{M,g[d/x]} = 1$ for some $d \in$ D.

The goal here is to provide a semantics in which quantified formulas can be used to express generalizations about objects in the domain, instead of claims about particular objects. We do this by inserting a variable in the appropriate argument position, and then saying what must obtain given different ways of mapping that variable to objects in the domain. For $\exists$, there must be *some way* of assigning a value to the variable that makes the formula true; for $\forall$, *every way* of assigning a value to the variable must make the formula true.