MATHEMATICAL LOGIC FOR COMPUTER SCIENCE
(A.Y. 20/21)

HANDOUT N. 1

Abstract. Structures and First-Order Languages. Terms, formulas.

1. Structures

Before defining the language of first-order logic let's consider what kind of objects we want to talk about.

Graphs. A graph G is usually presented as an ordered pair (V, E) where V is a non-empty set of vertices and E is a binary relation on V; i.e. $E \subseteq V \times V$ (the edge relation).

Groups. In a standard Algebra course a group G is presented by specifying a non-empty set X, a binary operation (or function) on X, $\oplus : X \times X \to X$, and the neutral element (or identity) of the group, say $e \in X$.

Database. A relational database consists in a (non-empty) set of elements, A, and a number of subsets of A and relations on elements of A. For example, the following components specify a basic genealogical database. A finite set of individuals, a subsets U of A containing all and only the male elements of A, a binary relations on A, $M \subseteq A \times A$ containing the ordered pairs $(a, b) \in A \times A$ such that a is married to b, and a binary relation $P \subseteq A \times A$ containing the ordered pairs (a, b) such that a is a parent of b.

Arithmetic. The natural numbers: besides the underlying set $\mathbf{N}$ of natural numbers, it is common in Number Theory to consider: an ordering relation $<$ on $\mathbf{N}$, the addition and multiplication functions $+$ and $\times$, and to single-out two specific elements, 0 and 1, as the neutral elements of addition and, respectively, multiplication.

All the above examples are common examples of the general notion of mathematical structure.

Definition 1.1 (Structure)**.** A structure $\mathfrak{A}$ consist in the following components:

(1) A non-empty set A called the domain of the structure.
(2) Relations on A of any arity.
(3) Functions on A of any arity.
(4) Constants (i.e., elements of A).

We will denote structures in this sense by ordered tuples listing the components of the structure. For example, $\mathcal{G} = (\{0, 1, 2, 3, 4, 5\}, \{(0, 1), (0, 2), (2, 3), (4, 5)\})$ denotes a structure that is a graph; $\mathcal{N} = (\mathbf{N}, <, +, \times, 0, 1)$ denotes the usual structure of the natural numbers with the standard ordering, operations, and constants 0 and 1.

2. First-order language, terms, formulas

What kind of things do we want to express about structures? Generally, we want to express natural global properties of structures and local properties of elements of structures. For example, of a graph $G = (V, E)$, we might want to express global properties such as: G is undirected, G is complete, G is infinite, G is connected, G has diameter 2, etc.; as well as local properties of elements of G: there is a path of length 3 between vertes v and vertex w, v is isolated, etc.

Notes by Lorenzo Carlucci, please report typos or errors to lorenzo.carlucci@uniroma1.it.

Relative to the genealogical database, we naturally want to express concepts such as: a is the uncle of b, b is an ancestor of a, every person has exactly two parents, etc.

The concept of **predicate** or **first-order** language is designed in order to reflect the complexity of what we called structures. Basically we have one linguistic (syntactical) component corresponding to each (semantic, real-world) component of a structure. Besides, we have properly logical syntactical elements such as variables, logical connectives, quantifiers, and the identity symbol.

An important constraint that we insist on in First-Order Logic is that we want to express concepts by **quantifying only on elements of the structure** (rather than to subsets, relations, subsets of subsets etc) of the structure. This is why this logic is called **first-order**. Other logics exist in which quantification over subsets of a structure is allowed (higher-order logic).

For example, if we wish to express the fact that a node v in a graph G has at least two distinct neighbours, we would use the following approach: "There exists w, there exists z such that w and z are distinct, v is a neighbor of z and v is a neighbour of w"; rather than the equally legit formulation: "The set of neighbors of v has cardinality ≥ 2", which refers to subsets of V and to the notion of carinality (which is usually defined in terms of bijective functions).

Definition 2.1 (Predicate Language)**.** A *predicate or first-order language* is a collection $\mathcal{L}$ of symbols of the following types.

(1) An infinite countable set of **variables**.
(2) A finite or countably infinite set of **relation symbols**, each with its multiplicity/arity/dimenson.
(3) A finite or countably infinite set of **function symbols**, each with its multiplicity/arity/dimension.
(4) A finite or countably infinite set of **constant symbols**.

We usually denote the official variables of our predicate languages as $v_1, v_2, \ldots$ or $x_1, x_2, \ldots$. We will use x, y, z, w (with subscripts) as variables on variables (also called metavariables). Each relation symbol and each function symbol comes equipped with a fixed arity, i.e., a fixed number of arguments it can take. We denote relation symbols with uppercase Latin letters (if necessary with indices): e.g. R, S, T_1, U_3, etc. We denote function symbols with lowercase Latin letters (if necessary with indices): e.g., f, g, h, p_4, s_5, etc. We denote constant symbols with the first letters of the Latin alphabet in lowercase (if necessary with indices), e.g., a, b, c_7, d_{22}.

The definition above describes a predicative language in its maximum generality. In many cases we will consider specific languages, for example, without function or constant symbols. Depending on the mathematical theory or problem that we intend to formalize we will choose *a suitable* predicative language.

If the language does not contain function symbols we call it a *relational language*; if it does not contain function symbols and constant symbols we call it a *purely relational language*.

Example 2.2. A suitable language for Graph Theory is $\mathcal{L}_{\text{graphs}} = \{E\}$, where E is a binary relation symbol, denoting the edge relation.

Example 2.3. A suitable language for Group Theory is the following $\mathcal{L}_{\text{groups}} = \{\cdot, e\}$; where $\cdot$ is a symbol of binary function for the operation of the group and e a constant for the neutral element. Note that it might make sense to also include a function symbol for the inverse operation, which is defined in every group; so a different language for Graph Theory is also the following $\mathcal{L}' = \{\cdot, inv, e\}$, where inv is function symbol with one argument.

Example 2.4. A suitable, succint, language for arithmetic is the following $\mathcal{L}_A = \{<, +, \times, 0, 1\}$, where $, \times$ are symbols of functions to two arguments, $<$ a symbol of binary relation and $0, 1$ are constant. Note that we are using $<, +, \times, 0, 1$ as symbols here, not as the set-theoretic objects that they usually denote!

Example 2.5. A language for Set Theory is the following $\mathcal{L}_{\text{sets}} = \{\in\}$ where $\in$ is a symbol of binary relation.

Terms. Symbols for constants and functions can be combined to obtain *proper names* for elements of the domain. These names are called terms. The official definition extends the idea to include variables and terms obtained by applying function symbols to variables.

Definition 2.6 (Terms)**.** The *terms* are obtained starting from the variables and the constants and closing under application of function symbols. A term that does not contain variables is called a *closed* term.

According to the official definition, terms are built using prefix notation. Thus, for example, in a language for arithmetic $\mathcal{L} = \{<, +, \times, 0, 1\}$ the following are official terms: $+(0, v_{12})$, $\times(+(1, +(1, 1)), \times(+(1, 1), +(v_3, 0)))$, etc. For readability we will use standard infix notation: $0 + v_{12}$, $(1 + (1 + 1)) \times ((1 + 1) \times (v_3 + 0))$, etc.

Example 2.7. $((v_9 + 1) + 1) \times 0$ is a term in the language $\mathcal{L}_A$. $e \cdot e$, $e \cdot (e \cdot e)$, $(e \cdot e) \cdot e$ are closed terms in the language $\mathcal{L}_{\text{groups}}$. The terms of the language $\mathcal{L}_{\text{sets}}$ are only the free variables.

Formulas. We want to use $\mathcal{L}$ to express properties of structures, i.e., properties that can be true or false of all or some elements of an intended structure. To create formulas in the language $\mathcal{L}$ we use the **logical symbols**

- propositional (Boolean) connectives $\wedge$ (and), $\vee$ (or), $\rightarrow$ (if...then), $\neg$ (not), $\leftrightarrow$ (if and only if).
- quantifiers $\exists$ (exists), $\forall$ (for all).
- the symbol of identity $=$.

Remark 2.8. Here we do not care about economy of symbols. In fact, the connectives $\neg, \vee$ are sufficient to express all the truth functions, and $\rightarrow$ and $\leftrightarrow$ are definable from them. This economic approach is convenient when we must prove results by induction on the structure of a formula because it allows us to consider less cases. Even $\forall$ and $\exists$ are inter-definable.

Remark 2.9. We declared the identity symbol as a logical symbol. Another approach to identity in first-oder logic is possible: identity can be treated as a binary relation symbol, say $I(x, y)$. In this case one should add special propositions that force the interpretation of the symbol to be the identity relation in the intended model. We will discuss below some difficulties that arise in this respect.

Definition 2.10 (Atomic Formulas)**.** An *atomic formula* is a string of the type $(t = s)$, where t, s are terms, or a string of the type $R(t_1, \ldots, t_k)$, where R is a relation symbol of arity k and $t_1, \ldots, t_k$ are terms.

Example 2.11. $0 < (v_1 + 1) \times ((v_4 + 1) + v_2)$ is an atomic formula in $\mathcal{L}_A$. The unique atomic formulas in $\mathcal{L}_{\text{sets}}$ are of the form $x \in y$ and $x = y$ with x, y variables.

Definition 2.12 (Formulas)**.** The *formulas* are obtained starting from the atomic formulas and closing under propositional connectives and universal and existential quantifiers. The (non-atomic) formulas are therefore of the type

$$(F \wedge G), (F \vee G), (\neg F), (F \rightarrow G), (F \leftrightarrow G), ((\forall v)F), ((\exists v)F),$$

where F and G are formulas and v is a variable. The above definition should be read as an inductive definition of a set of finite strings in the alphabet of a given predicate language.

In formulas $((\forall v)F)$ and $((\exists v)F)$, F is called the *domain* (or *scope*) of the quantifier.

A *subformula* of a formula F is any substring of F that appears in some step of the inductive decomposition of F according to the definition of formula, including F itself.

Free and bound variables An *occurrence* of a variable v in a formula F is any appearance of the symbol v in the string of symbols that make up F. An occurrence of a variable v in a formula F is *bound* if v occurs in a (sub)formula G in a context of the form $((\forall v)G)$ or $((\exists v)G)$ within F; that is, if it is in the scope of a quantifier quantifying over that variable.

Note: the same variable can have in the same formula free and bound occurrences. Yet, as we will see it is always possible to assume that no variable has free and bound occurrences in a formula.

Notation: If F is a formula and $x_1, \ldots, x_n$ are *distinct* variables, we write $F(x_1, \ldots, x_n)$ to indicate that the free variables of F are *contained* in the set $\{x_1, \ldots, x_n\}$. Similarly we can associate to a term the set of variables contained in it.

Definition 2.13 (Sentence)**.** A *sentence* is a formula without free variables.

We will see in a moment that sentences are of central interest.