

Lecture Notes: Discrete Mathematics

GMU Math 125-001 Spring 2007

Alexei V Samsonovich

Any theoretical consideration, no matter how fundamental it is, inevitably relies on key primary notions that are accepted without definition. In our case, examples of notions that we shall accept without definitions are: an object, a value, a concept, a symbol, to name just a few. In fact, most English words used here are not defined yet expected to be understood. Given this agreement, our first step is to give useful definitions of the key terms, notations and concepts wisely selected among others. Below in this text, whenever a key term is introduced, it is typed in bold italic. Synonyms are listed in parentheses. An example follows immediately.

Discrete mathematics (sometimes understood as ***finite mathematics***) is the study of mathematical structures that do not support or require the notion of continuity. In particular, all computer implementations of continuous mathematical structures involve their finite discretization in a certain sense. Topics of discrete mathematics include logic, sets, number theory, combinatorics, graphs, algorithms, probability, information, complexity, computability, and more.

Lecture notes included here should be considered complementary rather than alternative to the textbook. The intent is to help with understanding of difficult concepts: sometimes by taking the reader to a higher level of abstraction that opens a better view of the main subject, and sometimes by squeezing the essence from long pages into just a few lines.

Lecture 1: Logic

It is generally understood that mathematics plays the role of a theoretical fundament for all modern natural sciences. In the same sense, logic can be viewed as a theoretical fundament for mathematics in general and for discrete mathematics in particular. It is not surprising therefore that we start our consideration with logic. In order to introduce concepts of logic “from first principles”, we need to introduce some other general notions first. By “introduce” I do not mean “rigorously define”. Many of these notions will be addressed in more detail and used extensively in subsequent lectures, so in a sense the following three paragraphs are just an outline.

A ***set*** is a collection of distinct objects called ***elements*** (or ***members***) of the set. In particular, the notion of a set does not allow for multiple instances (repetitions) of the same element in the set. In contrast, a ***sequence*** is an ordered collection that allows for repetition of its elements. An ***ordered n -tuple*** is a sequence of n elements. If X is a set, then X^n (the n -th power of X) is the set of all ordered n -tuples over X (“over X ” here means composed from elements of X). An n -place ***relation*** on X is a subset of X^n .

An **expression** is a finite sequence of elements-symbols (the entire set of symbols of a given language is called **alphabet** or **vocabulary**). A **well-formed expression** satisfies certain grammar rules (that need to be specified separately in each particular case of a formal theory). Below the term **expression** will refer to well-formed expressions only. A **language** is a set of expressions over an alphabet. A **formal axiomatic theory** is a language in which a subset of expressions are called **axioms**, and the rest are connected to the axioms via relations called **inference rules**. A sequence of these connections that starts from axioms is called a **proof**, and the expression at the end of this sequence is a **theorem**. A theory is **decidable**, if there is an **algorithm** to determine for any given expression whether it has a proof.

Here we compare two theories: propositional calculus and predicate calculus (both are addressed in more detail below). Elements of **propositional calculus** are truth-valued (Boolean) variables, constants and expressions constructed from them using **connectives** (AND, OR, etc.). Elements of **predicate calculus** are variables that may take values of any nature (e.g., numbers, people), plus arbitrary truth-valued functions of those variables called **predicates**, plus connectives and quantifiers used to construct expressions. A **variable** is a symbol that can be associated with (**bound** to) a value, an object, a set, or an expression. A variable can be **free** or **bound**. Variables may have types: restrictions on the nature of objects to which they can bind. On the other hand, a **quantifier** specifies a set of objects to which a variable is (to be) bound. While it is possible to design an unlimited number of quantifiers, two of them are commonly used in most cases: the **universal quantifier** \forall , specifying that something applies (the variable can be bound) to any element of a certain category, and the **existential quantifier** \exists , specifying that something applies to at least one element.

Given the above outline, we can define **propositional calculus** (propositional logic) as a formal axiomatic theory L with the alphabet, syntax, set of axioms and inference rules that follow below. The alphabet includes connectives ($\neg, \wedge, \vee, \rightarrow, \leftrightarrow$), parentheses: “(“, “)”, and variables denoted here by lowercase English letters that may appear with or without subscripts. Each variable in L is bound to a **proposition** (a **statement**), which is an expression that is either true or false (i.e., has a definite **truth value**). The set of expressions in L can be formally given as follows.

1. All letters that denote statements with definite truth values are expressions of L.
2. If B and G are expressions of L, then so are $(\neg B)$ and $(B \rightarrow G)$.
3. If B, G and D are placeholders for expressions of L, then the following axiom schemas define axioms of L:
 - a. $(B \rightarrow (G \rightarrow B))$
 - b. $((B \rightarrow (G \rightarrow D)) \rightarrow ((B \rightarrow G) \rightarrow (B \rightarrow D)))$
 - c. $((\neg G) \rightarrow (\neg B) \rightarrow (((\neg G) \rightarrow B) \rightarrow G))$
4. It suffices for L to have the only inference rule, **modus ponens**: $((a \rightarrow b), a) \Rightarrow b$.

Again, in summary, **propositional calculus** (propositional logic) can be defined as a formal axiomatic theory in which the alphabet includes variables denoted by letters (e.g.,

p, q_1), symbols **1** and **0** used for tautology and contradiction, connectives ($\neg, \wedge, \vee, \rightarrow, \leftrightarrow$) and parentheses: “(”, “)”, the set of axioms includes those given by the above schemas (a), (b) and (c), and the only inference rule is modus ponens. We can avoid using unnecessary parentheses, if we agree on a particular **order of precedence** for connectives, e.g.: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, and then \leftrightarrow and other symbols of meta-language.

We can reason about variables in L without knowing their values, yet we assume that they have definite values within the set $\{T, F\}$. **This feature separates propositional calculus from predicate calculus, in which predicates may have free variables.** In L, all expressions are propositions. A direct analogy with regular numerical expressions can be seen. Indeed, propositional logic is isomorphic to binary arithmetic (calculus of Boolean numbers: zeros and ones), as the following translation table shows. This translation provides an intuition into L and can be taken as a part of its definition.

Table 1. L is isomorphic to Boolean arithmetic.

Propositional logic	Boolean arithmetic
T	1
F	0
$p \wedge q$	pq
$p \vee q$	$p+q-pq$
$\neg p$	$1-p$
$p \rightarrow q$	$1-p+pq$
$p \leftrightarrow q$	$1-p-q+2pq$
$p \underline{\vee} q$ (XOR)	$p+q-2pq$
$p \wedge p \leftrightarrow p$	$p^2 \equiv p$
$p \wedge \neg p \leftrightarrow \mathbf{0}$	$p(1-p) \equiv 0$

Given this table, you can confidently solve your HW1 problems (or you can use the truth tables), yet there is a more convenient way of doing this: by using **equivalence relations** listed on page 24 in the textbook. At least, with this table you should have no problem understanding what L is about: it is a calculus in which all variables and expressions take only two values. In this sense it is much simpler than predicate calculus.

Therefore, for example, it is not surprising that $p \rightarrow q$ and $q \rightarrow p$ cannot be simultaneously false in propositional logic. It is also clear that there may be cases when $P \leftrightarrow Q$ holds and $P \Leftrightarrow Q$ does not (generally, whenever $P \leftrightarrow Q$ holds, $P \Leftrightarrow Q$ is a tautology, and vice versa).

The analogy between \wedge and multiplication, \vee and addition with regard to **commutativity**, **associativity** and **distributivity** is straightforward. In particular, it implies that equivalence relations allow us to reduce any logical expression to its **disjunctive normal form** (dns): i.e., a “logical sum” (disjunction) of **disjuncts**, each of which is either a “logical product” (conjunction) of **literals** (i.e., variable letters or their negations) or a single literal. The counterpart of dns is a **conjunctive normal form** (cns) defined similarly, with swapping \wedge

and \vee . These forms can be useful when you want to compare two expressions. A dns / cns is called **full**, if each of its disjuncts / conjuncts is a **minterm** (contains each variable used in the form exactly once). Every non-contradiction / non-tautology can be reduced to a full dns / cns. An expression is **satisfiable** if it is true for some values of its variables.

A logical **argument** is a finite set of propositions called **premises** or **hypotheses** followed by a proposition called **conclusion**. In a **valid argument**, the conclusion is true whenever premises are true. A **sound argument** is a valid argument in which premises are true. An arguments can be written in a column form (below), or as a compound statement with an implication connecting premises to the conclusion. As a compound statement, a valid argument is a tautology. Moreover, its variables can be treated as placeholders for expressions (**substitution theorem**). All this is useful for problem solving. A table of useful arguments – some inference rules derived from modus ponens – is given below.

Table 2. Inference rules.

Modus ponens	$\begin{array}{l} p \rightarrow q \\ \underline{p} \\ q \end{array}$	Modus tollens	$\begin{array}{l} p \rightarrow q \\ \underline{\neg q} \\ \neg p \end{array}$
\leftrightarrow introduction	$\begin{array}{l} p \rightarrow q \\ \underline{q \rightarrow p} \\ p \leftrightarrow q \end{array}$	Contrapositive	$\begin{array}{l} p \rightarrow q \\ \underline{\neg q \rightarrow \neg p} \end{array}$
Case analysis	$\begin{array}{l} p \vee q \\ p \rightarrow r \\ \underline{q \rightarrow r} \\ r \end{array}$	Vacuous proof:	$\begin{array}{l} \underline{\neg p} \\ p \rightarrow q \end{array}$
\wedge introduction	$\begin{array}{l} p \\ \underline{q} \\ p \wedge q \end{array}$	\wedge elimination	$\begin{array}{l} \underline{p \wedge q} \\ p \end{array}$
\vee introduction	$\begin{array}{l} \underline{p} \\ p \vee q \end{array}$	\vee elimination	$\begin{array}{l} p \vee q \\ \underline{\neg p} \\ q \end{array}$
Disjunctive syllogism	$\begin{array}{l} p \vee q \\ \underline{\neg p} \\ q \end{array}$	Resolution	$\begin{array}{l} p \vee r \\ \underline{q \vee \neg r} \\ p \vee q \end{array}$
Chain rule	$\begin{array}{l} p \rightarrow q \\ \underline{q \rightarrow r} \\ p \rightarrow r \end{array}$	Contradiction	$\begin{array}{l} p \\ \underline{\neg p} \\ F \end{array}$

First-order logic (FOL, first-order predicate calculus) is an extension of propositional logic L that includes quantifiers, predicates (see above) and variables with arbitrary values beyond $\{T, F\}$, with the exception for predicates themselves, which makes this theory “first-order” (without this restriction we run into logical paradoxes). The set of axioms and the set of rules of inference in FOL, as compared to L , is augmented with (intuitively obvious) axioms and arguments involving quantifiers. A **higher-order logic** differs from FOL in that its predicates are allowed to have predicates as their arguments.

The notion of logic, especially in computer science, is rapidly expanding in many dimensions, including fuzzy logic, many-valued logic, quantum logic, modal logics (temporal, dynamic, deontic, alethic, epistemic, etc.). Related calculi include situation calculus, fluent calculus, event calculus, etc.

Further Reading:

Hamburger, H., and Richards, D. (2002) *Logic and Language Models for Computer Science*. Upper Saddle River, NJ: Prentice Hall.

Mendelson, E. (1997) *Introduction to Mathematical Logic*. New York: Chapman & Hall.