FONDAMENTI:
LOGICA

Prima Parte
logica del I ordine:
sintassi e semantica

We distinguish between *classes* and *sets*. Except in Chs. 10 and 11 (where the terms "class" and "set" are assigned a more precise technical meaning), a *class* is understood to be an arbitrary collection of objects, while a *set* is a class which can be a member of another class. (Another distinguishing feature of sets is that only they have cardinalities.)

Given $n \geqslant 1$ objects $x_1,\ldots,x_n$, we write $\langle x_1,\ldots,x_n\rangle$ for the *ordered n-tuple* of $x_1,\ldots,x_n$. Thus $\langle x,y\rangle$ is the *ordered pair* of x and y. *By convention*, we put $\langle x\rangle = x$ (the ordered singleton of x).

A *function (map, mapping)* is a class f of ordered pairs such that, whenever $\langle x,y\rangle \in f$ and $\langle x,z\rangle \in f$, we have $y=z$. The *domain* $\mathrm{dom}(f)$ of f is the class

$$\{x\colon \text{for some } y, \langle x,y\rangle \in f\}$$

and the *range* $\mathrm{ran}(f)$ of f is the class

$$\{y\colon \text{for some } x, \langle x,y\rangle \in f\}.$$

If A is a class and I is a set, we write A^I for the collection of all functions from I into A. (Notice that this definition implies $A^{\emptyset}=\{\emptyset\}=A^0$.) If $\{A_i\colon i\in I\}$ i an indexed family of sets, we write $\prod_{i\in I}A_i$ for the collection of all functions f with domain I such that $f(i)\in A_i$ for all $i\in I$. The *axiom of choice* asserts that, if each $A_i\neq\emptyset$, then $\prod_{i\in I}A_i\neq\emptyset$.

The Principle of Induction (IND)

In order to prove that $\forall n \in \omega . P(n)$

- basis: prove that *P*(0)
- induction step: prove that $\forall m \in \omega .\ P(m) \Rightarrow P(m+1)$

Course-of-values induction (C-IND)

$$(\forall m \in \omega . [(\forall k < m . Q(k)) \Rightarrow Q(m)]) \Rightarrow \forall n \in \omega . Q(n)$$

IND è equivalente a C-IND

I numeri naturali

Da un punto di vista ingenuo si può pensare ai naturali semplicemente come ad una sequenza infinita $\{0, 1, 2, 3, \ldots\}$.

I naturali sono una struttura $\mathsf{NAT} = \langle \mathbb{N}; 0; \mathsf{succ} \rangle$ t.c.

assioma 1: 0 è un elemento privilegiato di $\mathbb{N}$ detto *zero*;

assioma 2: $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$ è una operazione unaria iniettiva su A;

assioma 3: $0 \notin \mathrm{Im}(\mathsf{succ})$;

assioma 4: se $P \subseteq \mathbb{N}$ e valgono le seguenti proprietà:
i) $0 \in P$;
ii) $\forall n \in \mathbb{N}.(n \in P \Rightarrow \mathsf{succ}(n) \in P)$
allora $P = \mathbb{N}$.

3.1 Definizioni per induzione

Una prassi usuale in informatica è qualla di dare definizioni per induzione. Il caso più semplice è quallo della definizione per induzione di funzioni. Il teorema seguente ci garantisce che il procedimento di definizione induttivo è *buono*.

Teorema 3.1.1. *Sia* $h : \mathbb{N} \times A \to A$ *e* $c \in A$. *Esiste (ed è unica) una funzione* $f : \mathbb{N} \to A$ *t.c.:*
1. $f(0) = c$
2. per ogni $n \in \mathbb{N}$, $f(succ(n)) = h(n, f(n))$.

Esercizio 3.1.2. Sia $h : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ t.c. $h(x, y) = succ(x) \cdot y$. Per il teorema appena dimostrato esite un'unica funzione f t.c.:
$f(0) = 1$
$f(succ(n))) = h(n, f(n))$.
Che funzione è f ?

Dimostrazione

Esistenza di f: sia Ω la classe di tutti gli insiemi $Z \subseteq \mathbb{N} \times A$ t.c. $(0, c) \in Z$ e $(n, x) \in Z \Rightarrow (succ(n), h(n, x)) \in Z$; chiamiamo *soddisfacenti* tali insiemi.

E' immediato osservare che Ω non è vuoto.

Sia f l'intersezione di tutti gli elementi di Ω. L'insieme f è ovviamente soddisfacente ed è contenuto in ogni insieme soddisfacente.

Dimostriamo per induzione che : per ogni $n \in \mathbb{N}$ esiste esattamente un $a \in A$ t.c. $(n, a) \in f$ (ovvero f è una funzione).

caso base: sappiamo che $(0, c) \in f$. Supponiamo che $(0, d) \in f$ con $d \neq c$. Allora $f - \{(0, d)\}$ sarebbe sempre sodisfacente e sarebbe contenuto propriamente in f, impossibile.
passo induttivo: supponiamo che la proprietà valga per un generico naturale i, ovvero che esiste unico w t.c. $(i, w) \in f$. Per costruzione di f deve valere che: $(succ(i), h(i, w)) \in f$. Supponiamo ora che esista un elemento $e \neq h(i, w)$ t.c. $(succ(i), e) \in f$. L'insieme $f - \{(succ(i), e)\}$ è soddisfacente e sarebbe strettamente contenuto in f, impossibile.

Notiamo infine che la funzione f essendo soddisfacente soddisfa banalmente i punti (1.) e (2.) del teorema.

Unicità di f: supponiamo che esistano due funzioni f_1 e f_2 verificanti i punti (1.) e (2.) del teorema. Dimostriamo per induzione che $f_1 = f_2$.

caso base: $f_1(0) = f_2(0)$ per definizione. Supponiamo che per un generico naturale i, $f_1(i) = f_2(i)$.
passo induttivo: Applicando il punto due abbiamo che $f_1(succ(i)) = h(i, f_1(i)) = h(i, f_2(i)) = f_2(succ(i))$.

Definition 1.1.1 The language of propositional logic has an alphabet consisting of

proposition symbols : At={p_0, p_1, p_2, . . . } connectives : $\vee$, $\sim$, auxiliary symbols : (,).

Definition 1.1.2 The set PROP of propositions is the **smallest** set X with the properties
(i) $p_i \in X (i \in N)$, $\perp \in X$,
(ii) $\phi, \psi \in X \Rightarrow$, $(\phi \vee \psi) \in X$.

(iii) $\phi \in X \Rightarrow$ $(\sim\phi) \in X$.

Theorem 1.1.6 (Definition by Recursion) Let mappings
$H_\vee : A^2 \rightarrow A$;
$H_\sim : A \rightarrow A$
$H_{at} : At \rightarrow A$,
then there exists exactly one mapping
$F : PROP \rightarrow A$ such that
$F(\phi) = H_{at}(\phi)$ for ϕ atomic,
$F((\phi \vee \psi)) = H_\vee(F(\phi), F(\psi))$
$F((\sim\phi)) = H_\sim(F(\phi))$

BEGINNING MATHEMATICAL LOGIC

Let us start with an example. Consider the inference

(1) *Every tove is slithy*

(2) *Alice is not slithy*

---

(3) *Alice is not a tove*

1) $\forall x . tove(x) \supset slithy(x)$
2) $\neg\ slithy(Alice)$

---

3) $\neg\ tove(Alice)$

1) $\forall x . T(x) \supset S(x)$
2) $\neg\ S(a)$

---

3) $\neg\ T(a)$

(3) IS CONSEQUENCE OF (1) & (2)

We may say that (3) is a consequence of (1) and (2) by virtue of the *form* - as distinct from the *matter* - of these statements. In this connection the words "every" and "not" must be regarded as part of the form: if they are re-interpreted or replaced by other words, the inference may well become invalid

We need to construct artificial formal languages whose structure will be perfectly regular.

In dealing with a formal language $\mathcal{L}$ we must make a distinction between *syntax* and *semantics*.

First, there is the language that is *being discussed;* this is called the *object language*. Then there is the language *in which* the discussion takes place; this is called the *metalanguage*. The distinction between the two is extremely important and must be constantly borne in mind (even when the two languages happen to coincide)

Structure consists of the following ingredients:

(1) A non-empty class, called the *universe* or *domain* of the structure. The members of this universe are called the *individuals* of the structure.
(2) Various operations on the universe. These are called the *basic operations* of the structure. (**optional**) 0-ary operations are called its *designated individuals*.
(3) Various relations on the universe. These are called the *basic relations* of the structure.

running example: Elementary arithmetic

Elementary arithmetic may be defined as the study of one particular structure — *the elementary structure of natural numbers*. It has the set of natural numbers as universe, two designated individuals (viz. 0 and 1) and two basic operations (viz. addition and multiplication). Here the only basic relation is the identity relation.

Suppose we are given a structure $\mathfrak{U}$ and we want to set up a formal language $\mathscr{L}$ in which statements about $\mathfrak{U}$ are to be expressed. What symbols should $\mathscr{L}$ have?

First, we would like $\mathscr{L}$ to have symbols that may be used as *variables* ranging over the universe of $\mathfrak{U}$. The need for variables is obvious to anyone acquainted with mathematics. Variables ranging over the universe of $\mathfrak{U}$ are used, e.g., in expressing conditions which individuals of $\mathfrak{U}$ may or may not satisfy, and in making general statements about $\mathfrak{U}$.

Next, we expect $\mathscr{L}$ to have symbols that may be used to denote the various basic operations of $\mathfrak{U}$. Such symbols are called *function symbols*. More specifically, a symbol designed to denote an n-ary operation is called an *n-ary function symbol*. In particular, if $\mathfrak{U}$ has designated individuals then $\mathscr{L}$ should have symbols for denoting them. Such symbols are called *individual constants* or, more briefly, just *constants*. Since designated individuals are regarded as 0-ary operations, constants are to be regarded as 0-ary function symbols.

Using variables and function symbols, we can construct expressions called *terms*. Roughly speaking, terms are the nounlike expressions of $\mathscr{L}$.

For example, in a formal language suitable for elementary arithmetic we should have variables, say **x**,**y**, etc., intended to range over the set N of natural numbers; and function symbols, say **0**,**1**,**+**,**×**, intended to denote the numbers zero and one and the operations of addition and multiplication, respectively. Then **x**, **1**, **1+x**, **0×y**, **((1+x)×(y+1))+0** are some of the terms we can form.

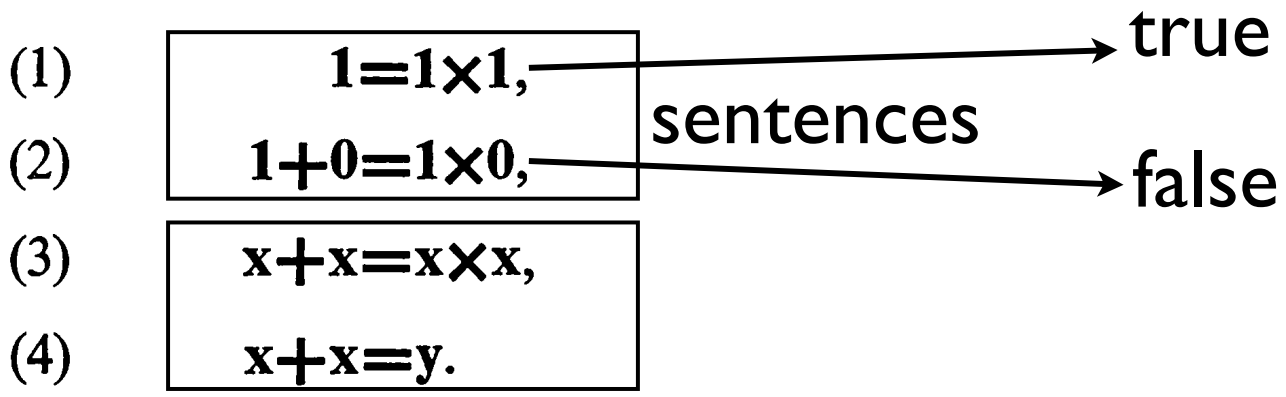
Of course, different interpretations can be applied to one and the same language. For example, the language described in the preceding paragraph can be re-interpreted by letting its variables range over some arbitrary non-empty class and letting $\mathbf{0},\mathbf{1},+,\times$ denote two arbitrarily chosen members of that class and two arbitrarily chosen binary operations on it.

But suppose we have fixed one particular interpretation for a formal language $\mathscr{L}$, by means of a structure $\mathfrak{U}$. Then those terms of $\mathscr{L}$ that do not contain variables will denote individuals of $\mathfrak{U}$. A term containing variables will not denote any particular individual, but will assume various individuals as values, depending on which individuals are assigned as values to the variables.

For example, in the particular language described above (taken with its originally intended interpretation) the term $\mathbf{1}+\mathbf{1}$ denotes the natural number two, while the term $(\mathbf{1}+\mathbf{1})\times\mathbf{x}+\mathbf{y}$ has as value the number obtained by adding whatever number is assigned as value to $\mathbf{y}$, to twice the number that happens to be assigned as value to $\mathbf{x}$.

Variables and function symbols alone do not suffice for formulating in $\mathscr{L}$ statements about a structure $\mathfrak{U}$. For this, $\mathscr{L}$ must have symbols that can be used to denote the basic relations of $\mathfrak{U}$. A symbol designed to denote an *n*-ary relation is called an *n-ary predicate symbol.*

If — as is usually the case — the identity relation is one of the basic relations of $\mathfrak{U}$, then $\mathscr{L}$ needs to have a predicate symbol to serve as a name for it. It is convenient to earmark one particular symbol, $=$, for this role.[1]



Formulas (3) and (4) are not sentences; they do not express propositions, but *conditions* regarding the values which may be assigned to the variables

We would like *negation* to be expressible in $\mathscr{L}$. Thus, for any formula α of $\mathscr{L}$ we want $\mathscr{L}$ to have a formula $\neg\alpha$ (read: "*not* α" or "*it is not the case that* α") which will be true whenever α is false, and false whenever α is true. (Thus α and $\neg\alpha$ always have opposite truth values.)

Next, we want the conjunction *and* to be expressible in $\mathscr{L}$. Thus, for any two formulas α, β we need a formula $\alpha \wedge \beta$ (read: "α and β") which will be true iff both α and β are true.

Similarly, we want $\mathscr{L}$ to have, for any formulas α and β, a formula $\alpha \vee \beta$ (read: "α *or* β") which is false iff both α and β are false.[1]

Further, we want $\mathscr{L}$ to be capable of expressing conditional statements. Therefore, for any formulas α, β of $\mathscr{L}$ there should be in $\mathscr{L}$ a formula $\alpha \rightarrow \beta$ (read: "α *implies* β" or "*if* α, *then* β"). This formula will be false iff α is true but β is false.[2]

Finally, $\mathscr{L}$ should have, for any formulas α and β a formula $\alpha \leftrightarrow \beta$ (read: "α *iff* β") which is true whenever α and β have the same truth values and false whenever their truth values are different.

We could satisfy all these demands by requiring $\mathscr{L}$ to have five logical symbols, called *connectives*, viz. $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$.

$$\alpha \leftrightarrow \beta \qquad (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$

$$\alpha \wedge \beta \qquad \neg(\alpha \rightarrow \neg\beta)$$

The last demand we shall make on $\mathscr{L}$ is that for any formula α and any variable x, $\mathscr{L}$ should have a formula $\forall x\alpha$ (read: "*for every* [*value of*] x, α") and a formula $\exists x\alpha$ (read: "*for some* [*value of*] x, α").

$\forall x(x=x\times x)$ false

$\exists x(x=x\times x)$ true

$\forall x(x\times y=z)$ satisfied if we assign to y and z the value 0

$\exists x\alpha \qquad \neg\forall x\neg\alpha$

BASIC SYNTAX

first-order language $\mathscr{L}$.

The *symbols* of $\mathscr{L}$ are:

(a) An infinite sequence of *(individual) variables,* namely

$$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \ldots .$$

(b) For each natural number n, a set of *n-ary function symbols.*

(c) For each positive natural number n, a set of *n-ary predicate symbols.*

For at least one n this set must be non-empty.

(d) The *connectives* $\neg$ *(negation)* and $\rightarrow$ *(implication).*

(e) The *universal quantifier* $\forall$.

The 0-ary function symbols (if any) are called *(individual) constants.*

If $\mathscr{L}$ has the special binary predicate $=$, we say that $\mathscr{L}$ is a language *with equality. We stipulate that if* $\mathscr{L}$ *has at least one function symbol that is not an individual constant, then* $\mathscr{L}$ *must be a language with equality.*[1]

The variables, the connectives, the universal quantifier and $=$ are called *logical* symbols. They are assumed to be the same in all first-order languages (or, in the case of $=$, in all first-order languages with equality). The function symbols and the predicate symbols other than $=$ are called *extralogical* symbols and may differ from one language to another.

A finite (possibly empty) sequence of $\mathscr{L}$-symbols is called an *$\mathscr{L}$-string*.

We shall only be interested in two kinds of strings, called *terms* and *formulas*.

$\mathscr{L}$-terms

$\mathscr{L}$-terms are $\mathscr{L}$-strings formed according to the following two rules:

(1) Any $\mathscr{L}$-string consisting of (a single occurrence of) a variable is an *$\mathscr{L}$-term*.

(2) If $\mathbf{f}$ is an n-ary function symbol of $\mathscr{L}$ and $\mathbf{t}_1,\ldots,\mathbf{t}_n$ are *$\mathscr{L}$-terms*, then $\mathbf{ft}_1\ldots\mathbf{t}_n$ is an *$\mathscr{L}$-term*.

Notice that, for $n=0$, (2) says that any constant is a term. In a term $\mathbf{ft}_1\ldots\mathbf{t}_n$ formed according to (2), $\mathbf{t}_1,\ldots,\mathbf{t}_n$ are called *arguments* of $\mathbf{f}$.

By the *degree of complexity* of a term $\mathbf{t}$ (briefly deg $\mathbf{t}$) we mean the total number of occurrences of function symbols in $\mathbf{t}$.

$\mathscr{L}$-formulas

$\mathscr{L}$-formulas are $\mathscr{L}$-strings formed according to the following four rules:

(1) If **P** is an n-ary predicate symbol of $\mathscr{L}$ and $\mathbf{t}_1,\ldots,\mathbf{t}_n$ are $\mathscr{L}$-terms, then $\mathbf{P}\mathbf{t}_1\ldots\mathbf{t}_n$ is an *$\mathscr{L}$-formula.*

(2) If α is an *$\mathscr{L}$-formula,* then so is $\neg\alpha$.

(3) If α and β are *$\mathscr{L}$-formulas*, then so is $\rightarrow\alpha\beta$.

(4) If α is an *$\mathscr{L}$-formula* and **x** is a variable, then $\forall\mathbf{x}\alpha$ is an *$\mathscr{L}$-formula.*

A formula $\mathbf{P}\mathbf{t}_1\ldots\mathbf{t}_n$ formed according to (1) is called an *atomic* formula; the terms $\mathbf{t}_1,\ldots,\mathbf{t}_n$ here are the *arguments* of **P**. An atomic formula whose predicate symbol is $=$ is called an *equation* and its first and second arguments are called its *left-hand side* and *right-hand side,* respectively.

The *degree of complexity* of a formula α (briefly deg α) is the sum obtained by adding up 2 for each occurrence of $\rightarrow$ and 1 for each occurrence of $\neg$ and $\forall$ in α.

If S is a term (or formula) and R,T are strings such that RST is again a term (or formula, respectively), then S is said to be a *subterm* (or *subformula*, respectively) of RST. If moreover R is non-empty[1] then S is a *proper* subterm (or subformula, respectively) of RST.

Notational conventions

(a) Boldface lower-case Roman letters from the end of the alphabet (especially "**x**", "**y**", "**z**") are used as syntactic variables ranging over the variables of a first-order language.

(b) As syntactic variables ranging over function symbols we use "**f**", "**g**" and "**h**".

(c) As syntactic variables ranging over constants we use "**a**", "**b**" and "**c**".

(d) As syntactic variables ranging over predicate symbols we use "**P**", "**Q**" and "**R**".

(e) As syntactic variables ranging over terms we use "**r**", "**s**" and "**t**".

(f) Boldface lower-case and upper-case Greek letters are used as syntactic variables ranging over formulas and sets of formulas, respectively.

5.1. DEFINITION.

(a) $(\mathbf{r}=\mathbf{s}) =_{df} =\mathbf{rs}$.

(b) $(\mathbf{r}\neq\mathbf{s}) =_{df} \neg(\mathbf{r}=\mathbf{s})$.

(c) $(\alpha \to \beta) =_{df} \to\alpha\beta$.

(d) $(\alpha \wedge \beta) =_{df} \neg(\alpha \to \neg\beta)$.

(e) $(\alpha \vee \beta) =_{df} (\neg\alpha \to \beta)$.

(f) $(\alpha \leftrightarrow \beta) =_{df} ((\alpha \to \beta) \wedge (\beta \to \alpha))$.

(g) $\exists x\alpha =_{df} \neg\forall x\neg\alpha$.

priority order “$\leftrightarrow$”, “$\rightarrow$”, “$\vee$”, “$\wedge$”

The ranges of “$\neg$”, “$\forall$” and “$\exists$” are to be as short as possible.

$$\alpha \wedge \beta \rightarrow \gamma \leftrightarrow \neg \alpha \rightarrow \beta \vee \gamma \quad \text{is} \quad \{[(\alpha \wedge \beta) \rightarrow \gamma] \leftrightarrow [\neg \alpha \rightarrow (\beta \vee \gamma)]\}$$

$$\exists x \alpha \wedge \beta \vee \gamma \quad \text{is} \quad [(\exists x \alpha \wedge \beta) \vee \gamma].$$

association to the right

$$\alpha \rightarrow \beta \wedge \gamma \rightarrow \beta \rightarrow \gamma \quad \text{is} \quad \alpha \rightarrow \{(\beta \wedge \gamma) \rightarrow (\beta \rightarrow \gamma)\}$$